Homebrew-file Documentation

Release 3.12.3

rcmdnk

Jul 31, 2017

Contents

Brewfile manager for Homebrew at OS X.

Contents:

CHAPTER 1

Installation

Install Homebrew-file with Homebrew:

```
$ brew install rcmdnk/file/brew-file
```

or you can use install script:

```
$ curl -fsSL https://raw.github.com/rcmdnk/homebrew-file/install/install.sh |sh
```

which installs Homebrew itself, too, if it is not installed.

Then, add following lines in you **.bashrc** or **.zshrc** to wrap `brew` command:

```
if [ -f $(brew --prefix)/etc/brew-wrap ];then
  source $(brew --prefix)/etc/brew-wrap
fi
```

brew-wrap wraps the original `brew` command for an automatic update of **Brewfile** when you execute such a `brew install` or `brew uninstall`.

Note: 17/Dec/2015 update

The default place of Brewfile has been changed from:

```
/usr/local/Library/Brewfile
```

to:

```
~/.brewfile/Brewfile
```

because Homebrew deletes files under **/usr/local** other than Homebrew's one when such `brew update` is executed. (Homebrew checkout its repository as **/usr/local**.)

If you used an old default setting (**/usr/local/Library/Brewfile**), you might lose Brewfile.

In such case, please try `brew file init` and chose local Brewfile, which makes new file **~/.brewfile/Brewfile**.

If you used git repository, you might see a output when you executed `brew update`:

```
$ brew update
Ignoring path Library/rcmdnk_Brewfile/
To restore the stashed changes to /usr/local run:
  `cd /usr/local && git stash pop`
  Already up-to-date.
```

In this case, please delete **/usr/local/Library/<your_git_account>_Brewfile**, then do `brew file set_repo`.

New repository will be checked out to **~/.brewfile/<your_git_account>_Brewfile**.

CHAPTER 2

Requirements

- Homebrew (Can be installed by the install script of brew-file, too).
- Python 2.7.7 or later (optional).
- Requests module (optional)

Although it is not mandatory, the latest Python 2.7.x or 3.x is recommended, to use `brew file brew` command (and `brew-wrap`).

You can install the latest python2.x by Homebrew:

```
$ brew install python
```

or python3.x:

```
$ brew install python3
```

If you set your `PYTHONPATH` for python3, you may need to execute `brew-file` directly:

```
$ python3 /usr/local/bin/brew-file
```

instead of `brew file`.

- Can't ignore unknown argument in subparser of ArgumentParser of Python even if parse_known_args is given
- PythonArgumentParsersubparserparse_known_argsunknown

Requests module of python is needed to create Brewfile repository with `setup_repo`. If you make the repository by yourself, it is not needed.

To install:

```
$ easy_install pip # # in case you've not installed pip
$ pip install requests
```

CHAPTER 3

Getting Started

Use local Brewfile

By default, **Brewfile** is **~/.brewfile/Brewfile**.

If you don't have **Brewfile**, first, do:

```
$ brew init
```

`brew init` is same as `brew file init`, if you setup `brew-wrap` as above.

Note: In below, `set_repo` command can be used directly after `brew`, but `install` or `update` need to use with `brew file` because `brew` command has own `install` or `update` commands.

You can check your package list by:

```
$ brew file cat
```

If you already have **Brewfile**, then copy it to **~/.brewfile/Brewfile** and install packages listed in **Brewfile** by:

```
$ brew file install
```

After that, you need to do only normal `brew` commands, like `brew install` or `brew uninstall`. After each command, **Brewfile** is updated automatically if you set `brew-wrap` as above.

When you get new Mac, copy **~/.brewfile** to new Mac and just do:

```
$ brew file install
```

Use Dropbox (or any online storages) for Brewfile management

Set Brewfile place

You can set the place of Brewfile by using the environment variable like:

```
export HOMEBREW_BREWFILE=~/Dropbox/Brewfile
```

Then, you can use Brewfile as same as the original Brewfile place.

In this case, when you have new Mac, set `HOMEBREW_BREWFILE` and synchronize the file with a online storage service, then do:

```
$ brew file install
```

If you are using multiple Mac in the same time, it is good to have a cron job like:

```
30 12 * * * brew file update
```

This command installs new packages which were installed in another Mac at a lunch time (12:30) every day.

This command also does `brew update && brew upgrade`, and removes packages not listed in `Brewfile`.

If you want to do only installing new packages, then set as:

```
30 12 * * * brew file install
```

Use GitHub (or any git repository) for Brewfile management

Set up a repository

First, create a repository with a file named **Brewfile**.

If you use GitHub, you can make it with brew-file:

```
$ brew set_repo

Set repository, "non" for local Brewfile.
<user>/<repo> for GitHub repository,
or full path for the repository:
```

Give a name like `rcmdnk/Brewfile` (will be recognized as a GitHub repository), or such `git@github.com:rcmdnk/Brewfile`.

Then, initialize **Brewfile**:

```
$ brew init
```

Set up new Mac with your Brewfile in the repository

Do:

```
$ brew set_repo
```

and give your repository name.

And install packages listed in **Brewfile** like:

```
$ brew file install
```

Brewfile management

To update the repository, do:

```
$ brew file update
```

If you have set the repository, this command does `git pull` and `git push` in addition to such brew's `install`, `clean`, `update`, `updgrade` and removing packages described in online storages section above.

It is good if you have such a cron job like:

```
30 12 * * * brew file update
```

The repository is updated at lunch time every day.

CHAPTER 4

Usage

Brew-file manages packages installed by Homebrew. It also supports brew-cask and other Homebrew subcommand installers.

It uses input file. By default, the file is **~/.brewfile/Brewfile**. You can reuse `Brewfile` for Brewdler, too.

If you want to specify input file, use `-f` option.

If you want to change default `Brewfile`, set environmental variable: `HOMEBREW_BREWFILE` in your setup file (e.g. **.bashrc**), like:

```
export HOMEBREW_BREWFILE=~/.brewfile
```

You can also modify the default installation locations of Cask packages. To make this settings, it is the same as issuing How to Use Homebrew-cask#Options. You might want to add the following line to your **.bashrc** or **.zshenv**:

```
export HOMEBREW_CASK_OPTS="--appdir=$HOME/MyApplications"
```

Similarly, you can specify the environment for brew-gem. The following will tell brew-gem to use the Ruby installed by Homebrew itself:

```
export HOMEBREW_GEM_OPTS="--homebrew-ruby"
```

If there is no `Brewfile`, Brew-file will ask you if you want to initialize `Brewfile` with installed packages or not. You can also make it with `install` (`-i`) subcommand.

With `install` subcommand, Brew-file tries to install packages listed in `Brewfile`. If any packages managed with Homebrew Cask are listed, brew-cask is also installed automatically.

`Brewfile` convention is similar as Brewdler. Normally, you don't need to modify anything on Brewdler's `Brewfile` for Brew-file

Example:

```
# Tap repositories and their packages
tap caskroom/cask
brew 'brew-cask'
```

```
# install brew-cask # install is same as "brew". Quotes are not mandatory.

tapall rcmdnk/file # This will trigger `brew install brew-file`, too

# Cask packages
cask firefox
#cask install firefox # "cask install" is same as "cask"

# Other Homebrew packages
brew mercurial
brew macvim --with-lua

# Additional files
file ~/.Brewfile
```

First column is command. Second to the last columns are package name and options. They are used as arguments for such `brew install`, therefore any options of Homebrew can be used.

Command	For what? (X is package+options)
brew	`brew install X`
install	Same as `brew`
tap	`brew tap X`
tapall	`brew tap X`, and installs all packages of Formulae in the tap.
cask	`brew cask install X`. Require caskroom/homebrew-cask. (It will be installed automatically.)
pip	*brew pip X*. Require hanxue/brew-pip. (It will be installed automatically.)
gem	`brew gem install X`. Require sportngin/brew-gem. (It will be installed automatically.)
appstore	Apps installed from AppStore. The line is like: *appstore <identifier> <App Name>*. Identifier can be obtained by argon/mas. (It will be installed automatically.) For older OS X, it might be not available. For such a case, only App names are listed by `init`, and `install` command just warns like `Please install <App Name> from App Store!`.
file	Additional files. A path is a absolute path, or a relative path, relative to the file which calls it.
brewfile	Same as `file`.
before	Execute `X` at the beginning of the install.
after	Execute `X` after all install commands.
Anything others	Execute the line (first and other columns as one line) before `after` is executed.

If the second column is `install`, it will be ignored.

i.e. `brew install package` is same as `brew package`.

If you want to build macvim with lua option, you can write as above example `Brewfile`.

If you use `tap`, Brew-file only does `tap` the repository.

If you use `tapall`, Brew-file does `brew install` for all Formulae in the repository in addition to do `tap` the repository.

If you want to divide the list into several files. If the main `Brewfile` has `file` or `brewfile` commands, then corresponding argument is used as an external file. The path can be an absolute or a relative. If you use a relative path, like .''/Brewfile2'', then the start directory is the directory where the main `Brewfile` is.

You can use a nest of `file`, too. The relative path starts from the parent file's directory.

For the path, such ~ is translated into `$HOME`, and any environmental variables can be used.

e.g.

If you have:

```
file ./${HOST}.Brewfile
```

in main `Brewfile`, and prepare files like:

```
Brewfile Host1.Brewfile Host2.Brewfile Host3.Brewfile
```

in the same directory, then `brew-file` picks up **Host1.Brewfile** for Host1, and **Host2.Brewfile** for Host2, etc...

Or if you just have:

```
file ~/.Brewfile
```

then you can put Host specific packages in **~/.Brewfile**. (If the file doesn't exist, `brew-file` just ignores it.)

Other example: Add an option to ignore appstore apps · Issue #22 · rcmdnk/homebrew-file

You don't need to `brew install` by hand. As written above, `tap 'caskroom/cask'` is can be dropped because `cask 'firefox'` triggers it.

Some packages such macvim has Application (MacVim.app). If you want to install them to Applications area, please use `-l` (for `~/Applications/`) or `-g` (for `/Applications/`).

With `clean` option, Brew-file runs cleanup. By default, it just does dry run (no actual cleanup). To run cleanup in non dry-run mode, use `-C`.

If you want edit `Brewfile`, use `edit` option.

> **Warning:** If you do `brew file edit` before installing `Brewfile` and save w/o any modification, you may make empty `Brewfile` (Be careful, `brew -c -C` remove all packages :scream:). Therefore I recommend you to do `brew file -i` at first if you don't have `Brewfile`.

You can maintain your `Brewfile` at the git repository. First, make new repository at GitHub (or other git server).

Then, set the repository by:

```
$ brew file set_repo -r <repository>
```

It will clone the repository. If the repository has a file named `Brewfile`, the file will be used instead of `~/.brewfile/Brewfile`. (then `~/.brewfile/Brewfile` will have this repository informatoin.)

`repository` should be like rcmdnk/Brewfile in GitHub, which should have `Brewfile` (different file name can be used by `-f`).

If you want to use other hosts than github, use full path for the repository, like:

```
$ brew file set_repo -r git@bitbucket.org:rcmdnk/my_brewfile
```

If the repository doesn't have `Brewfile` (or specified by `-f`, `brew file init` initialize the file. Then, you can push it by `brew file push`.

With this procedure, you can synchronize all your Mac easily :thumbsup:

To install new package, use:

```
$ brew file brew install <package>
```

instead of `brew install <package>`, because above command automatically update `Brewfile`.

This is useful especially if you are using the repository for the `Brewfile`, and want to use `brew file update`.

Otherwise, please be careful to use `brew file update`, because it deletes what you installed, but you have not registered in `Brewfile`.

If you want to check your Apps for Cask, use:

```
$ brew file casklist
```

This command makes `Caskfile.txt`, which is like:

```
### Cask applications
### Please copy these lines to your Brewfile and use with `brew bundle`.

### tap and install Cask (remove comment if necessary).
#tap caskroom/cask
#install brew-cask

### Apps installed by Cask in /Applications
cask install adobe-reader # /Applications/Adobe Reader.app
cask install xtrafinder # /Applications/XtraFinder.app

### Apps installed by Cask in /Applications/Utilities:
cask install xquartz # /Applications/Utilities/XQuartz.app

### Apps installed by Cask in ~/Applications.
cask install bettertouchtool.rb # ~/Applications/BetterTouchTool.app

##############################

### Apps not installed by Cask, but installed in /Applications.
### If you want to install them with Cask, remove comments.
#cask install keyremap4macbook # /Applications/KeyRemap4MacBook.app

### Apps not installed by Cask, but installed in /Applications/Utilities:
### If you want to install them with Cask, remove comments.

### Apps not installed by Cask, but installed in ~/Applications.
### If you want to install them with Cask, remove comments.
#cask install copy.rb # ~/Applications/Copy.app


##############################

### Apps not registered in Cask, but installed in /Applications.
# /Applications/App Store.app
# /Applications/Calendar.app
...

### Apps not registered in Cask, but installed in /Applications/Utilities:
...

### Apps not registered in Cask, but installed in ~/Applications.
```

You can find applications which were installed manually, but can be managed by Cask under “Apps not installed by Cask, but installed in...”.

If you want to manage them with `Brewfile`, just copy above lines w/o “#” for these Apps.

CHAPTER 5

Settings

Following environmental variables can be used.

Name	Description	Default
HOMEBREW_BREWFILE	Set place of Brewfile.	"~/brewfile/Brewfile"
HOMEBREW_BREWFILE_BACKUP	If it is set to not empty, Brewfile's back up is made to HOMEBREW_BREWFILE_BACKUP when Brewfile is updated.	""
HOMEBREW_BREWFILE_LEAVES	Set 1 if you want to list up only leaves (formulae which don't have any dependencies, taken by *brew leaves*).	0
HOMEBREW_BREWFILE_ON_REQUEST	Set 1 if you want to list up only packages installed on request. If it is set 1, it is given priority over *LEAVES* option. Note: This list can be changed if packages installed by brew-file in new machine. (some "on_request" package could be installed as "as_dependencies" of others before being installed on request.)	0
HOMEBREW_BREWFILE_TOP_PACKAGES	Packages which are listed in Brewfile even if *leaves* is used and they are under dependencies. (Useful for such *go*, which is used by itself, but some packages depend on it, too.)	""
HOMEBREW_BREWFILE_VERBOSE	Set verbose level.	1
HOMEBREW_BREWFILE_APPSTORE	Set 0 you don't want to list up AppStore applications Brewfile.	1
HOMEBREW_CASK_OPTS	This is Cask's option to set cask environment. If appdir or fontdir is set with these options, Brew-file uses these values in it.	""
HOMEBREW_GEM_OPTS	This is brew-gem's option to set Ruby environment.	""

CHAPTER 6

brew-wrap

If you want to automatically update Brewfile after `brew install/uninstall`, please use `brew-wrap`.

homebrew-file/etc/brew-wrap has a wrapper function `brew`.

Features:

- It executes `brew file init` after such `brew install` automatically.
- `file` can be skipped for non-conflicted commands with `brew`.
  - e.g.) `init` command is not in `brew`. Then, you can replace `brew file init` with:

    ```
    $ brew init
    ```

  - Such `edit` command is also in `brew`. In this case, `brew edit` executes original `brew edit`.
    - But you can use `brew -e` or `brew --edit` to edit **Brewfile**.

To enable it, just read this file in your `.bashrc` or any of your setup file:

```
if [ -f $(brew --prefix)/etc/brew-wrap ];then
  source $(brew --prefix)/etc/brew-wrap
fi
```

`brew` function in `brew-wrap` executes original `brew` if `brew-file` is not included.

Therefore, you can safely uninstall/re-install brew-file even if you have already sourced it.

Some subcommands of `brew-file` can be used as a subcommand of `brew`, if the command is not in original brew subcommands.

Such `init` or `casklist` commands can be used like:

```
$ brew init # = brew file init

$ brew casklist # brew file casklist
```

With completion settings below, `file` is included in the completion list of `brew`.

In addition, the completion for `brew file` is also enabled, as same as `brew-file` command.

> **Warning:** Previously, `brew-wrap` was in `bin/brew-wrap`, and it was used like `alias brew="brew-wrap"`.
>
> If you have this obsolete setting, please delete and renew as above.

CHAPTER 7

Completion

Functions for Bash/Zsh completions are also installed.

For Bash, please install Bash-Completion by:

```
$ brew install bash-completion
```

then, add following settings to your **.bashrc**:

```
brew_completion=$(brew --prefix 2>/dev/null)/etc/bash_completion
if [ $? -eq 0 ] && [ -f "$brew_completion" ];then
  source $brew_completion
fi
```

For Zsh, add following settings in your **.zshrc**:

```
brew_completion=$(brew --prefix 2>/dev/null)/share/zsh/zsh-site-functions
if [ $? -eq 0 ] && [ -d "$brew_completion" ];then
  fpath=($brew_completion $fpath)
fi
autoload -U compinit
compinit
```

In case you have installed zsh-completions (can be installed by brew: `$ brew install zsh-completions`), settings can be like:

```
for d in "/share/zsh-completions" "/share/zsh/zsh-site-functions";do
  brew_completion=$(brew --prefix 2>/dev/null)$d
  if [ $? -eq 0 ] && [ -d "$brew_completion" ];then
    fpath=($brew_completion $fpath)
  fi
done
autoload -U compinit
compinit
```

If you are using `brew-wrap`, please write these completion settings **BEFORE** `brew-wrap` reading.

CHAPTER 8

Help

Help message of brew-file:

```
usage: brew-file [-f INPUT] [-b BACKUP] [-F FORM] [--leaves] [--on_request]
                 [--top_packages TOP_PACKAGES] [-U] [--preupdate] [-r REPO]
                 [-n] [--caskonly] [--no_appstore] [-C] [-y] [-V VERBOSE] [-h]
                 [command] ...

Brew-file: Manager for packages of Homebrew
https://github.com/rcmdnk/homebrew-file

requirement: Python 2.7 or later

optional arguments:
  -f INPUT, --file INPUT
                        Set input file (default: /Users/<USER>/.brewfile/Brewfile).
                        You can set input file by environmental variable,
                        HOMEBREW_BREWFILE, like:
                            export HOMEBREW_BREWFILE=~/.brewfile
  -b BACKUP, --backup BACKUP
                        Set backup file (default: ).
                        If it is empty, no backup is made.
                        You can set backup file by environmental variable, HOMEBREW_
↪BREWFILE_BACKUP, like:
                        .    export HOMEBREW_BREWFILE_BACKUP=~/brewfile.backup
  -F FORM, --format FORM, --form FORM
                        Set input file format (default: none).
                        file (or none)    : brew vim --HEAD --with-lua
                        brewdler or bundle: brew 'vim', args: ['with-lua', 'HEAD']
                          Compatible with [homebrew-bundle](https://github.com/
↪Homebrew/homebrew-bundle).
                        command or cmd    : brew install vim --HEAD --with-lua
                          Can be used as a shell script.
  --leaves              Make list only for leaves (taken by `brew leaves`).
                        You can set this by environmental variable, HOMEBREW_BREWFILE_
↪LEAVES, like:
```

```
                        .    export HOMEBREW_BREWFILE_LEAVES=1
  --on_request          Make list only for packages installed on request.
                        This option is given priority over 'leaves'.
                        You can set this by environmental variable, HOMEBREW_BREWFILE_
↪ON_REQUEST, like:
                        .    export HOMEBREW_BREWFILE_ON_REQUEST=1
  --top_packages TOP_PACKAGES
                        Packages to be listed even if they are under dependencies and␣
↪`leaves`/'on_request' option is used.
                        You can set this by environmental variable, HOMEBREW_BREWFILE_
↪TOP_PACKAGES (',' separated), like:
                        .    export HOMEBREW_BREWFILE_TOP_PACKAGES=go,coreutils
  -U, --noupdate        Do not execute `brew update/brew upgrade` at `brew file␣
↪update`.
  --preupdate           Execute `brew update` before install or other commands.
  -r REPO, --repo REPO  Set repository name. Use with set_repo.
  -n, --nolink          Don't make links for Apps.
  --caskonly            Write out only cask related packages
  --no_appstore         Don't check AppStore applications.
                        (For other than casklist command.)
                        You can set input file by environmental variable:
                            export HOMEBREW_BRWEFILE_APPSTORE=0
  -C                    Run clean as non dry-run mode.
                        Use this option to run clean at update command, too.
  -y, --yes             Answer yes to all yes/no questions.
  -V VERBOSE, --verbose VERBOSE
                        Verbose level 0/1/2
  -h, --help            Print Help (this message) and exit.

subcommands:
  [command]
    install             Install packages in BREWFILE.
                        Use `--preupdate` to execute `brew update` before install.
    brew                Execute brew command, and update BREWFILE.
    init                or dump/-i/--init
                        Initialize/Update BREWFILE with installed packages.
    set_repo            or -s/--set_repo
                        Set BREWFILE repository (e.g. rcmdnk/Brewfile).
    pull                Update BREWFILE from the repository.
    push                Push your BREWFILE to the repository.
    clean               or -c/--clean
                        Cleanup.
                        Uninstall packages not in the list.
                        Untap packages not in the list.
                        Cleanup cache (brew cleanup)
                        By drault, cleanup runs as dry-run.
                        If you want to enforce cleanup, use '-C' option.
    update              or -u/--update
                        Do brew update/upgrade, pull, install,
                        init and push.
                        In addition, pull and push
                        will be done if the repository is assigned.
                        'clean' is also executed after install if you give -C option.
    edit                or -e/--edit
                        Edit input files.
    cat                 or --cat
                        Show contents of input files.
    casklist            Check applications for Cask.
```

```
    cask_upgrade        Check updates of cask applications.
                        With -C, upgrade is enforced (old versions will be removed).
    test                or --test. Used for test.
    get_files           Get Brewfile's full path, including additional files.
    commands            or --commands
                        Show commands.
    version             or -v/--version
                        Show version.
    help                or -h/--help
                        Print Help (this message) and exit.

                        Check https://homebrew-file.readthedocs.io for more details.
```

CHAPTER 9

Homebrew-file module index

Brew-file: Manager for packages of Homebrew https://github.com/rcmdnk/homebrew-file

requirement: Python 2.7 or later

class BrewFile.**BrewFile**
Main class of Brew-file.

ask_yn(*question*)
Helper for yes/no.

cask_upgrade()
Upgrade cask applications

cat_brewfile()
Cat brewfiles

check_brew_cmd()
Check Homebrew

check_cask()
Check applications for Cask

check_cask_cmd(*force=False*)
Check cask is installed or not

check_gem_cmd(*force=False*)
Check gem is installed or not

check_github_repo()
helper to check and create GitHub repository.

check_input_file()
Check input file

check_mas_cmd(*force=False*)
Check mas is installed or not

check_pip_cmd(*force=False*)
Check pip is installed or not

check_repo()
Check input file for GitHub repository.

clean_list()
Remove duplications between brewinfo.list to extra files' input

cleanup()
Clean up.

edit_brewfile()
Edit brewfiles

execute()
Main execute function

find_app(*app*, *taps*, *casks*, *nonapp_casks*, *casks_noinst*, *nonapp_casks_noinst*)
Helper function for Cask

find_brew_app(*name*, *tap*)
Helper function for Cask to find app installed by brew install

get_appstore_list()
Get AppStore Application List

get_cask_list(*force=False*)
Get Cask List

get_files(*is_print=False*)
Get Brewfiles

get_list()
Get List

initialize(*check=True*)
Initialize Brewfile

install()
Install

make_pack_deps()
Make package dependencies

parse_env_opts(*env_var*, *base_opts=None*)
Returns a dictionary parsed from an environment variable

remove(*path*)
Helper to remove file/directory.

repo_file()
Helper to build Brewfile path for the repository.

repomgr(*cmd=''*)
Helper of repository management.

set_args(***kw*)
Set arguments.

set_brewfile_repo()
Set Brewfile repository

class BrewFile.**BrewHelper**(*opt*)
Helper functions for BrewFile.

proc(*cmd*, *print_cmd=True*, *print_out=True*, *exit_on_error=True*, *separate_err_msg=False*, *verbose_level=1*, *env={}*)
Get process output.

class BrewFile.**BrewInfo**(*helper*, *filename=''*)
Homebrew information storage.

get_option(*package*, *package_info=''*)
get install options from brew info

get_tap(*tap*)
Helper for tap configuration file

get_tap_path(*tap*)
Get tap path

class BrewFile.**Tee**(*out1*, *out2=<open file '<stdout>', mode 'w'>*, *use2=True*)
Module to write out in two ways at once.

close()
Close output files.

flush()
Flush the output

write(*text*)
Write w/o line break.

writeln(*text*)
Write w/ line break.

BrewFile.**my_decode**(*word*)
Decode when python3 is used.

BrewFile.**my_input**(*word*)
Input method compatibility.

BrewFile.**open_output_file**(*name*, *mode='w'*)
Helper function to open a file even if it doesn't exist.

Python Module Index

b

Index