

## Contents

<b>1</b>	<b>git-dotfiles: Store \$HOME dot-files in Git</b>	<b>1</b>
1.1	Activating git-dotfiles To Manage Your \$HOME . . . . .	1
1.2	Use \$HOME/.bash_personal for Private Configurations . . . .	2
<b>2</b>	<b>Nix Enviroment via nix-env</b>	<b>2</b>
<b>3</b>	<b>Rust Development</b>	<b>3</b>
<b>4</b>	<b>References</b>	<b>3</b>
4.1	Dot-file Configuration . . . . .	3
4.1.1	<a href="https://atlassian.com/git/tutorials/dotfiles">https://atlassian.com/git/tutorials/dotfiles</a> .	3
4.2	Nix . . . . .	3
4.2.1	<a href="https://nixcademy.com/posts/nix-on-macos/">https://nixcademy.com/posts/nix-on-macos/</a> . . .	3
4.2.2	<a href="https://checkoway.net/musings/nix">https://checkoway.net/musings/nix</a> . . . . .	3
4.2.3	<a href="https://nixos.org/manual/nixos/stable/#module-services-emacs">https://nixos.org/manual/nixos/stable/#module-services-emacs</a>	3
4.2.4	<a href="https://nixos.wiki/wiki/TexLive">https://nixos.wiki/wiki/TexLive</a> . . . . .	3

## 1 git-dotfiles: Store \$HOME dot-files in Git

### 1.1 Activating git-dotfiles To Manage Your \$HOME

Check out the "bare" (.git directory) of `pjkundert/git-dotfiles` Git repo to `~/git-dotfiles`:

```
sudo apt install git
git clone --bare git@github.com:pjkundert/git-dotfiles.git .git-dotfiles
```

To activate, first create the `git-dotfiles` alias, and configure it to not show untracked files (the rest of the contents of your `$HOME` directory!):

```
alias git-dotfiles='git --git-dir=$HOME/.git-dotfiles --work-tree=$HOME'
git-dotfiles config --local status.showUntrackedFiles no
```

See how your current home directory's dot-files configuration compares:

```
git-dotfiles status
```

If this is a freshly created, you'll probably see that your home directory is probably missing a bunch of `.alias`, `.bashrc`, ... files:

```
$ git-dotfiles status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    .alias
    deleted:    .bash_profile
...
```

So, if you're satisfied, just reset your home directory's dotfiles to the contents of the master branch's HEAD, to install all these files:

```
git-dotfiles reset --hard HEAD
```

Otherwise, if you have some custom content in some eg. `.bashrc` files that you want to integrate, you could `stash save` them, reset to the recommended baseline, and then `stash apply` your custom stuff onto the baseline:

```
git-dotfiles stash save
git-dotfiles reset --hard HEAD
git-dotfiles stash apply
```

## 1.2 Use `$HOME/.bash_personal` for Private Configurations

For API tokens, etc., put them in `.bash_personal`. This file will not be stored in the `git-dotfiles` Git repo.

## 2 Nix Enviroment via `nix-env`

First: do NOT install nix from the Apt repositories! It is broken in significant ways, as of Q1 2025.

Install Nix in multi-user mode, load its environment variables, and update its channels according to the supplied `.nix-channels`:

```
$ sudo apt install curl
$ sh <(curl -L https://nixos.org/nix/install) --daemon # on macOS, skip --daemon
$ . .bash_profile
$ nix-shell -p nix-info --run "nix-info -m"
- system: ...
$ nix-channel --update
unpacking 2 channels...
$ nix-env-update # to install the env.nix packages; uncomment tex for emacs PDF export
```

Use the `nix-env-update` alias to update your system to the current set of targets in `env.nix`.

A (partial) replacement for `homebrew` is provided by the `$HOME/env.nix` file. List any desired Nix targets here to have them added to your `$HOME/.nix-profile/bin`.

### 3 Rust Development

You must configure `$HOME/.rustup/settings.toml` to specify `default_host_triple` and `default_toolchain` as appropriate for your local environment.

### 4 References

#### 4.1 Dot-file Configuration

This implementation was derived from:

##### 4.1.1 <https://atlassian.com/git/tutorials/dotfiles>

Provides a way to have a Git managed `$HOME` directory.

- Changed the `git` alias for accessing the Git repo to `git-dotfiles`

#### 4.2 Nix

##### 4.2.1 <https://nixcademy.com/posts/nix-on-macos/>

General Nix functionality on macOS.

##### 4.2.2 <https://checkoway.net/musings/nix>

Sensibly manage the Nix defaults used by `nix-env`, etc.

##### 4.2.3 <https://nixos.org/manual/nixos/stable/#module-services-emacs>

Provision a specific version of Emacs and any modules required.

##### 4.2.4 <https://nixos.wiki/wiki/TeXLive>

Integrated this approach to provisioning texlive into `emacs.nix`.