**The best way to store your dotfiles: A bare Git repository**

*Disclaimer: the title is slightly hyperbolic, there are other proven solutions to the problem. I do think the technique below is very elegant though.*

Recently I read about this amazing technique in an [Hacker News thread](https://news.ycombinator.com/item?id=11070797) on people's solutions to store their [dotfiles](https://en.wikipedia.org/wiki/Dot-file). User StreakyCobra [showed his elegant setup](https://news.ycombinator.com/item?id=11071754) and ... It made so much sense! I am in the process of switching my own system to the same technique. The only pre-requisite is to install [Git](https://www.atlassian.com/git).

In his words the technique below requires:

No extra tooling, no symlinks, files are tracked on a version control system, you can use different branches for different computers, you can replicate you configuration easily on new installation.

The technique consists in storing a [Git bare repository](http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/) in a "*side*" folder (like $HOME/.cfg or $HOME/.myconfig) using a specially crafted alias so that commands are run against that repository and not the usual .git local folder, which would interfere with any other Git repositories around.

**Starting from scratch**

If you haven't been tracking your configurations in a Git repository before, you can start using this technique easily with these lines:

git init --bare $HOME/.cfg  
alias config='/usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME'  
config config --local status.showUntrackedFiles no  
echo "alias config='/usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME'" >> $HOME/.bashrc

* The first line creates a folder ~/.cfg which is a [Git bare repository](http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/) that will track our files.
* Then we create an alias config which we will use instead of the regular git when we want to interact with our configuration repository.
* We set a flag - local to the repository - to hide files we are not explicitly tracking yet. This is so that when you type config status and other commands later, files you are not interested in tracking will not show up as untracked.
* Also you can add the alias definition by hand to your .bashrc or use the the fourth line provided for convenience.

I packaged the above lines into a [snippet](https://bitbucket.org/snippets/nicolapaolucci/ergX9) up on Bitbucket and linked it from a short-url. So that you can set things up with:

curl -Lks http://bit.do/cfg-init | /bin/bash

After you've executed the setup any file within the $HOME folder can be versioned with normal commands, replacing git with your newly created config alias, like:

config status  
config add .vimrc  
config commit -m "Add vimrc"  
config add .bashrc  
config commit -m "Add bashrc"  
config push

**Install your dotfiles onto a new system (or migrate to this setup)**

If you already store your configuration/dotfiles in a [Git repository](https://www.atlassian.com/git), on a new system you can migrate to this setup with the following steps:

* Prior to the installation make sure you have committed the alias to your .bashrc or .zsh:

alias config='/usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME'

* And that your source repository ignores the folder where you'll clone it, so that you don't create weird recursion problems:

echo ".cfg" >> .gitignore

* Now clone your dotfiles into a [bare](http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/) repository in a "*dot*" folder of your $HOME:

git clone --bare <git-repo-url> $HOME/.cfg

* Define the alias in the current shell scope:

alias config='/usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME'

* Checkout the actual content from the bare repository to your $HOME:

config checkout

* The step above might fail with a message like:

error: The following untracked working tree files would be overwritten by checkout:  
    .bashrc  
    .gitignore  
Please move or remove them before you can switch branches.  
Aborting

This is because your $HOME folder might already have some stock configuration files which would be overwritten by Git. The solution is simple: back up the files if you care about them, remove them if you don't care. I provide you with a possible rough shortcut to move all the offending files automatically to a backup folder:

mkdir -p .config-backup && \  
config checkout 2>&1 | egrep "\s+\." | awk {'print $1'} | \  
xargs -I{} mv {} .config-backup/{}

* Re-run the check out if you had problems:

config checkout

* Set the flag showUntrackedFiles to no on this specific (local) repository:

config config --local status.showUntrackedFiles no

* You're done, from now on you can now type config commands to add and update your dotfiles:

config status  
config add .vimrc  
config commit -m "Add vimrc"  
config add .bashrc  
config commit -m "Add bashrc"  
config push

Again as a shortcut not to have to remember all these steps on any new machine you want to setup, you can create a simple script, [store it as Bitbucket snippet](https://bitbucket.org/snippets/nicolapaolucci/7rE9K) like I did, [create a short url](http://bit.do/) for it and call it like this:

curl -Lks http://bit.do/cfg-install | /bin/bash

For completeness this is what I ended up with (tested on many freshly minted [Alpine Linux](http://www.alpinelinux.org/) containers to test it out):

git clone --bare https://bitbucket.org/durdn/cfg.git $HOME/.cfg  
function config {  
   /usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME $@  
}  
mkdir -p .config-backup  
config checkout  
if [ $? = 0 ]; then  
  echo "Checked out config.";  
  else  
    echo "Backing up pre-existing dot files.";  
    config checkout 2>&1 | egrep "\s+\." | awk {'print $1'} | xargs -I{} mv {} .config-backup/{}  
fi;  
config checkout  
config config status.showUntrackedFiles no

**Wrapping up**

I hope you find this technique useful to track your configuration. If you're curious, [my dotfiles live here](https://bitbucket.org/durdn/cfg.git). Also please do stay connected by following [@durdn](https://www.twitter.com/durdn) or my awesome team at [@atlassiandev](https://www.twitter.com/atlassiandev).