

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

February 17th 2016



Nicola Paolucci in [Tips](#), [Git](#)

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](#) on people's solutions to store their [dotfiles](#). User [StreakyCobra](#) [showed his elegant setup](#) 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](#).

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](#) 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:

```
1 git init --bare $HOME/.cfg
2 alias config='/usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME '
3 config config --local status.showUntrackedFiles no
```

```
4 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](#) 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](#) up on Bitbucket and linked it from a short-url. So that you can set things up with:

```
1 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:

```
1 config status
2 config add .vimrc
3 config commit -m "Add vimrc"
4 config add .bashrc
5 config commit -m "Add bashrc"
6 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](#), 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`:

```
1 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:

```
1 echo ".cfg" >> .gitignore
```

- Now clone your dotfiles into a **bare** repository in a *"dot"* folder of your `$HOME`:

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

- Define the alias in the current shell scope:

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

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

```
1 config checkout
```

- The step above might fail with a message like:

```
1 error: The following untracked working tree files would be overwritten by
  checkout:
2     .bashrc
3     .gitignore
4 Please move or remove them before you can switch branches.
5 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:

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

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

```
1 config checkout
```

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

```
1 config config --local status.showUntrackedFiles no
```

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

```
1 config status
2 config add .vimrc
3 config commit -m "Add vimrc"
4 config add .bashrc
5 config commit -m "Add bashrc"
```

```
6 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](#) like I did, [create a short url](#) for it and call it like this:

```
1 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](#) containers to test it out):

```
1 git clone --bare https://bitbucket.org/durdn/cfg.git $HOME/.cfg
2 function config {
3     /usr/bin/git --git-dir=$HOME/.cfg/ --work-tree=$HOME $@
4 }
5 mkdir -p .config-backup
6 config checkout
7 if [ $? = 0 ]; then
8     echo "Checked out config.";
9 else
10    echo "Backing up pre-existing dot files.";
11    config checkout 2>&1 | egrep "\s+\." | awk {'print $1'} | xargs -I{} mv {}
    .config-backup/{}
12 fi;
13 config checkout
14 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](#). Also please do stay connected by following [@durdn](#) or my awesome team at [@atlassiandev](#).

Latest developer blog posts



Coming soon: Jira 8.0

Tomek Bartyzel - August 3rd 2018 in [Jira](#), [Server](#), [Data Center](#), [APIs](#), [Early Access Program](#)

[Upcoming breaking changes across Server](#)



Change notice for Jira Cloud - Get more permissions resource will require a query parameter

Krzysztof Kercz - July 25th 2018 in [Jira](#), [Cloud](#), [API Experience](#)



Products

Peter Scobie - August 2nd 2018 in [Server](#), [Jira](#), [Confluence](#), [Bitbucket](#), [AUI](#), [Developer Experience](#)



Important Stride and Hipchat update: Atlassian and Slack announce a new partnership

Brendan Foley - July 26th 2018 in [Stride](#), [Hipchat](#), [Partnership](#), [Developer Experience](#)



Atlas Camp is back and the agenda is ever

Ralph Whitbeck - July 12th 2018 in [Atlas Camp](#), [Developer Experience](#)



Gear up for building for Stride with the Toolkit

Trevor Thompson - July 11th 2018 in [Stride](#), [Developer Experience](#)

[All developer blog posts](#) | [feed](#)

To stay up-to-date on developer topics, subscribe to our feed and join our mailing list!

[Help us improve](#) - [Report a problem with this page](#)

[System status](#)

[Privacy](#)

[Developer Terms](#)

[Trademark](#)

© 2018 Atlas