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 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:

```
git init --bare $HOME/.cfg
alias config='/usr/bin/git --git-dir=$HOME
config config --local status.showUntracked
echo "alias config='/usr/bin/git --git-dir
```

- 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:

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

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, 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
```

- 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 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
```

- 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 tre
    .bashrc
    .gitignore
Please move or remove them before you can
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
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.showUntracked
```

- 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 like I did, create a short url for it and call it like this:

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

For completeness this is what I ended up with (tested on many freshly minted Alpine Linux containers to test it out):

```
git clone --bare https://bitbucket.org/dur
function config {
   /usr/bin/git --git-dir=$HOME/.cfg/ --wo
}
```

```
    mkdir -p .config-backup
    config checkout
    if [ $? = 0 ]; then
      echo "Checked out config.";
      else
        echo "Backing up pre-existing dot file
        config checkout 2>&1 | egrep "\s+\." |
    fi;
    config checkout
    config config status.showUntrackedFiles no
```