# Placing Config Files Under Version Control with Git and GitHub

Posted At : October 30, 2009 1:39 PM | Posted By : Bob Silverberg
Related Categories: Git, OS X

Working with Git, I've become aware of the fact that there are certain config files on my machine which require customization and therefore would be nice to have under version control. These files are often referred to as dot files, or dotfiles, as their names all start with a dot. The three files that I currently have under version control are .bash_profile, .gitconfig and .gitignore. The first two of those files expect to reside in my home directory, but the way Git works, in order to place them under version control they need to reside in a folder that is also a Git repository.

For obvious reasons I don't want to make my home directory a Git repo, but there's a simple solution to this problem. Using symbolic links, a topic that I discussed in an earlier blog post, I can keep my dotfiles in a Git repo, and also continue to use them as live config files. Here is a step-by-step guide to getting your dotfiles under version control with Git:

1. Open a Terminal window.

2. Change to a directory in which you want to put your Git repo. E.g.,

```
cd ~/gitRepos
```

3. Create a directory called 'dotfiles':

```
mkdir dotfiles
```

4. Change back to your home directory:

```
cd ~/
```

5. Move your dotfiles from your home directory into the dotfiles directory, and lose the dot. E.g.,

```
mv .bash_profile ~/gitRepos/dotfiles/bash_profile
```

6. Create a symbolic link from the newly moved file to the location it is expected to be. E.g.,

```
ln -s -i -v ~/gitRepos/dotfiles/bash_profile .bash_profile
```

7. Change to the dotfiles directory. E.g.,

```
cd ~/gitRepos/dotfiles
```

8. Create a new Git repository:

```
git init
```

9. Add all the files in the current directory to the repo:

```
git add .
```

10. Commit the files to the Git repo:

```
git commit -m 'storing initial dotfiles in repo'
```

That's it. Your dotfiles are now under Git version control. It would be nice to have these files backed up somewhere though, don't you think? And also nice to be able to keep these files in sync if you have multiple machines on which you work. Enter GitHub.

I discussed getting set up with Git and GitHub in a previous post, so let's assume that you've got a GitHub account. Here are the steps to getting your local dotfiles repo linked with one on GitHub:

1. Create a new repository on GitHub called 'dotfiles'.

2. In a terminal window, change to your repo's directory. E.g.,

```
cd ~/gitRepos/dotfiles
```

3. Specify GitHub as a remote repository. E.g.,

```
git remote add origin git@github.com:bobsilverberg/dotfiles.git
```

Note that the last part of that command, the one that says "git@github.com:bobsilverberg/dotfiles.git" is the same as the one listed as *Your Clone URL* on the GitHub home page for your project.

4. Push your files to GitHub:

```
git push origin master
```

That's really it. Your local Git repo is now linked with your remote Git repo on GitHub. Any time you make new commits to your local repo, just push those changes to GitHub and you'll be backed up. You can also take advantage of the fact that your config files are now available on GitHub and keep multiple machines in sync. I'll discuss how to do that in a future post.