Homework 12
Towards a Development Environment

Due: Wednesday, April 4th, 11:59PM (Hard Deadline)

Submission Instructions

When you are done, submit a link to your GitHub repository here: https://goo.gl/forms/2PMqSRZ8IYb4gQCj2

1 A dotfiles repository

As we've seen in class, many tools store their configuration in "dotfiles" in your home directory. While this is convenient for configuring things on one machine, often you may have multiple machines that you use (desktop at work, laptop for travel, etc). It would be really nice if all of these things stayed in sync.

Fortunately, we've also learned a great tool for keeping things synchronized in this class – git! Keeping your configuration files in git has the added benefit that you can try changes that your aren't sure if you will like (e.g trying out vi mode in bash) knowing that you can easily revert them later.

The trick is that it's not a good idea to make your home directory a git repository (because then every file in your home directory would be in that repository!). Instead, create a folder named `dotfiles`, make that a repository, and then use symlinks to point from where tools expect your dotfiles (in your home directory) to where they actually are (the dotfiles folder):

```
$ ls -l ~/.bashrc
lrwxr-xr-x  1 ppannuto  staff  40 Mar 31 23:43 /Users/ppannuto/.bashrc ->
/Users/ppannuto/Dropbox/dotfiles/.bashrc
```

2 Remote work and configuring connections

Previously, we have directed you to look at `ssh` and `scp` for doing work on and moving files to and from remote machines. Repeated typing of anything in a Linux environment should be leading you to asking, "Is there any way I can do this with less typing?" The answer lies with the *OpenSSH SSH client configuration files*, whose reference can be found with `man ssh_config`. Full documentation can be found on the man page, of course, but some more concrete examples and sample files are right at the end of a google search.

This investigation will show you that hosts can be configured and referred to using simple (short) nicknames defined in ~/.ssh/config. An example from the instructor's config file looks like:

```
# UMich GitLab
Host gitlab
  HostName gitlab.com
  User git
  IdentityFile ~/.ssh/mmdarden_rsa

# CAEN
Host mmd
  HostName login.engin.umich.edu
  User mmdarden
  Compression yes
```

```
# Home media server
Host nas
  HostName 10.180.239.64
  User darden
  IdentityFile ~/.ssh/id_rsa
```

These hosts can be specified by their shorter nicknames anywhere you would use the full User@HostName. This leads to simplified logins, shorter `git clone` commmands, and easier file management with `scp`, among other things:

```
$ ssh mmd  # login to CAEN as mmdarden at login.engin.umich.edu
$ git clone gitlab:mmdarden/vimconfig.git
$ scp nas:music/artist/album/song.mp3 .  # copy a song to my local machine
```

The Assignment

Create a dotfiles repository. Move your existing `.bashrc` file into this repository, commit it, and set up a symlink from your home directory for bashrc. Create an `ssh_config` file and add it to your dotfiles repository. Finally, pick at least one other dotfile to move under version control and add it to this repository. For ideas, here are the files in my dotfiles folder:

```
.bash_aliases .bash_profile .bashfuncs .bashrc .gitconfig .gitignore .gsdesktop-helper[1]
.hgrc .inputrc .pypirc .screenrc .vim[2]  .vimrc
setup_dotfiles.sh[3]
ssh-config
```

You can also find *lots* of other dotfiles repositories on the web with tons of cool ideas for configurations. Some of the staff even have pretty good dotfiles repositories...

Use any repository hosting service (Umich Gitlab, GitHub, personal server, etc) to make your dotfiles repo publically accessible. Then submit a link to your repository at the link given on the first page of this homework

[1] Pour one out... (dead config file from this for Grooveshark)
[2] This is actually a directory for Pathogen
[3] This is a script that deletes any existing dotfiles (e.g. things usually ship with a template `.bashrc`) and replaces them with symlinks to the dotfile repo. It also handles ssh-config, which lives at ~/.ssh/config