16 - Working Remotely & Closing Git Branches

CS 2043: Unix Tools and Scripting, Spring 2016 [1]

Stephen McDowell

March 4th, 2016

Cornell University

Table of contents

Some Logistics

- Homework:
  - How long should it be taking me?
  - THEY ARE SO LONG MAN WHY?
    - Are they really? 3 vs 5...more time...lots of fluff.
    - (poll) It's supposed to be fun. Want me to remove it?
- Evaluations: please fill them out.
  - "Stephen has a stupid face and I don't like it."
  - Criticism is welcome; please provide input on how you think it could change to be better.
  - Please fill them out, *especially* for the TAs. Feedback helps us all develop, as well as gives you the opportunity to have an impact on future students.

Working Remotely

Some Terminology

- The server you are logging into is called the `remote` (host).
- The user (you) are referred to as the `client`.
- If you obtain access to a *cluster* (many individual nodes presented together), you may encounter terms such as:
  - The `master` node (sometimes called `head`).
  - The `slave` nodes (the workers).
  - You often are only allowed to log into the `master` node.
  - There is usually a queuing system (e.g. `qsub`) that submits `jobs` that get farmed out to the slaves.
  - In most scenarios, you get charged by the number of cores / resources you are using.

Using `ssh`

Secure Shell

`ssh [opts] <username@remote.host`

- `username` is the username on the *remote* host.
- `remote.host` is the url of the server you want to log into.
  - IP Address: `128.253.141.42`
  - Symbolic name: `csug11.csuglab.cornell.edu`
- Use `-l` to specify username (no need for `@` anymore).
- `-p <port>`: connect to a specific port (may be necessary depending on the server).
- Can forward graphical *programs* (NOT the entire screen):
  - Enable `X11` forwarding with `-X`.
  - Enable "trusted" `X11` forwarding with `-Y` (actually less secure, only use if needed).

`ssh` by Example

- On `csug` (CS Undergraduate) I am `sjm324`:
  - `ssh sjm324@128.253.141.42`
  - `ssh sjm324@csug11.csuglab.cornell.edu`
  - `ssh -l sjm324 csug11.csuglab.cornell.edu`
- Sweet! Hey `csug` has MATLAB, can I use it?

```
>>> /usr/local/MATLAB/R2012a/bin/matlab
Warning: No display specified.  You will not be able to
         display graphics on the screen.
exit()
# exit() left Matlab
>>> exit # close the ssh connection
```

- `ssh -X sjm324@csug11.csuglab.cornell.edu`

```
>>> /usr/local/MATLAB/R2012a/bin/matlab
```

Connecting to Servers

- Warning: you are being *heavily* monitored. Always.
  - Think before you try to do something even *remotely* dubious.
- Cornell `csug` has 15 redundant servers:
  - `{csug01..csug15}.csuglab.cornell.edu`
  - Files you make on `csug01` will appear on `csug10`!
  - If one is particularly slow, try another one.
- On campus, you do not need to log into the `vpn`.
- Off campus, you do (`ssh` will just hang).
  - Install: http://www.it.cornell.edu/services/vpn/howto/index.cfm
  - After installing, run `Cisco AnyConnect`, then `ssh` in.
  - The `vpn` can be pretty laggy sometimes, oddly usually between 2am and 4am.
- Your login: `NetID`. Password: same as `CMS` / `studentcenter`.
- More info: http://www.it.cornell.edu/support/coecis/cis/linux.cfm

Lets Have Some Fun!

- Remember those permissions I keep droning on about?
- They actually *do* mean something!
  - Now that we can `ssh`, you are in a system with *many* users and groups, and don't have access to everything like you do on your personal computer.
- Go ahead and `ssh` into `csug`.
- Our course playground is `/courses/cs2043`.
  - Your personal folder: `/courses/cs2043/<your_netID>`
  - The party: `/courses/cs2043/zzz_COLOR_PARTY`

Transferring Files

Secure Copy

`scp [flags] <from> <to>`

- It's exactly like **`cp`**, only you are transferring over the web.
- Transfer *from* the **`client`** *to* the **`remote`** host.
- Transfer *from* the **`remote`** host *to* the **`client`**.
- Copy directories just like before using the **`-r`** flag.
- Must specify **`user`** on the **`remote`**.
- **`Remote`** syntax:
  `user@host.name:/path/to/file/or/folder`
  - You need the **`:`** to start the **`path`**.
- If you don't have permission...you can't get it!
- More modern systems let you **`TAB`** complete across the **`remote`** directories :)

`scp` by Example

- Transfer from `remote` to local computer:

```
>>> scp sjm324@blargh.ru:/absolute/path/colorize.sh ~/Desktop/
colorize.sh                          100% 3299     3.2KB/s   00:00
```

- Transfer from `remote` to local:

```
>>> scp sjm324@blargh.ru:~/Desktop/colorize.sh /usr/share/
colorize.sh                          100% 3299     3.2KB/s   00:00
```

- Transfer from the `client` to the `remote`: just reverse it.

```
>>> scp /usr/share/colorize.sh sjm324@blargh.ru:~/Desktop/
colorize.sh                          100% 3299     3.2KB/s   00:00
```

- As with regular `cp`, can give a new name at the same time:

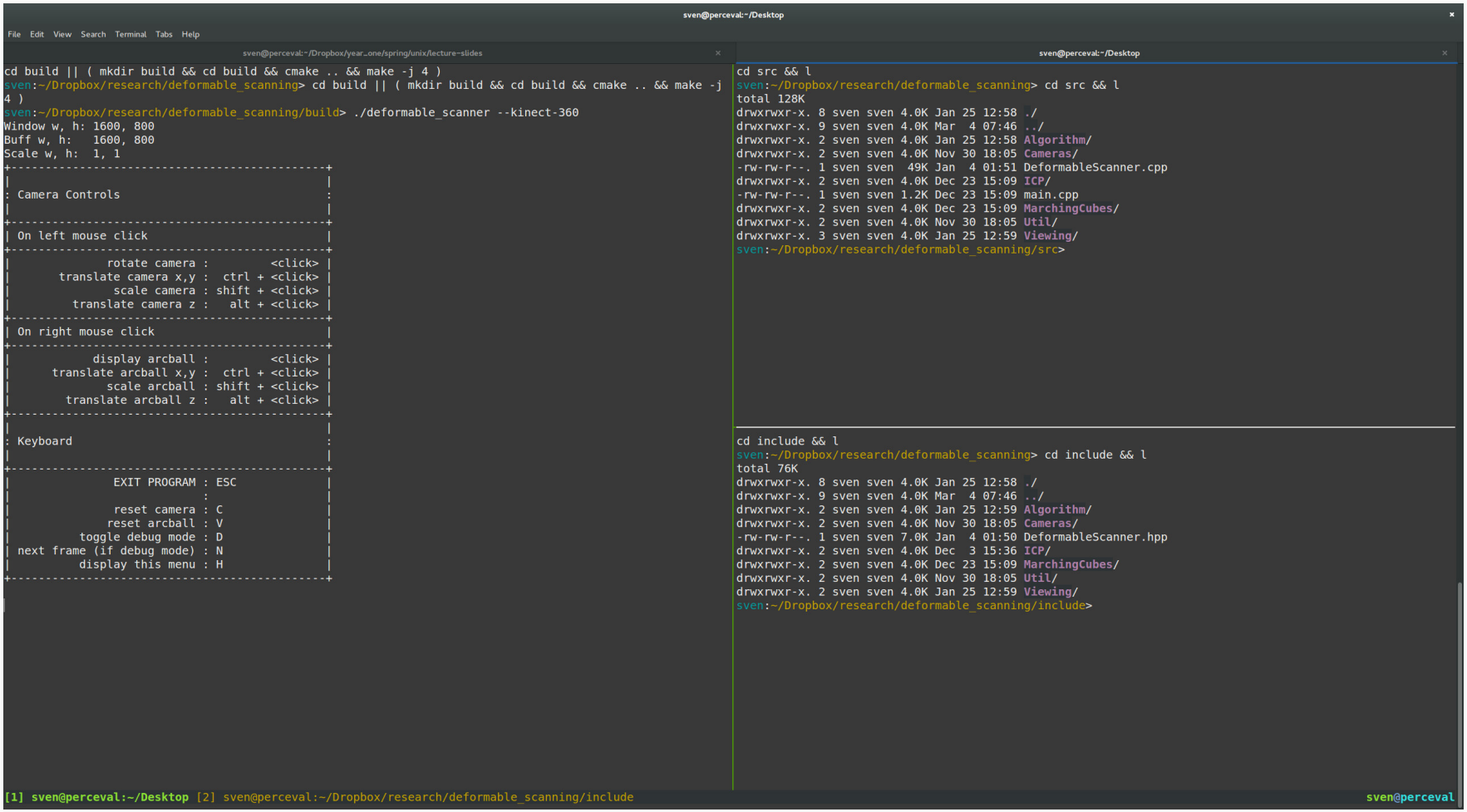
```
>>> scp /usr/share/colorize.sh sjm324@blargh.ru:~/new_name.sh
colorize.sh                          100% 3299     3.2KB/s   00:00
```

Multiplexing

What is Multiplexing

- Complex combinatorial logic meant to be studied with rigor and painful effort.
- But not in this class!
- Terminal multiplexing is just the ability to:
  - Split your terminal into multiple *panes*.
  - The ability to *detach* and re-*attach* to a `shell` without having to close it.
  - Also a whole lot more, but we will focus on these.
- You can leave your multiplexed terminal running on the `remote`, and connect to it with any `client` you want whenever you want.
- Extremely convenient if you want to be able to work effectively with `ssh`.
- Unfortunately, not available to you on `csug` (for good reason).

What does it Look Like?



Suggestion: `tmux`

Terminal Multiplexer

`tmux`

- Vanilla (no options) starts a new multiplexed instance.
- Can split into *panes* horizontally and vertically.
- Can `detach` (sort of put in background, but it is still running).
- Can re-`attach`.
- Can open new windows, sessions, panes, and more.
- `tmux list-{buffers,clients,commands,keys,panes,sessions,windows}`
- `ctrl+D` to close current *in-focus* pane / window.

Notes on Multiplexing

- Learn the hotkeys: http://tmuxcheatsheet.com/
- After you `ssh` in, just `tmux attach` to open the top-level session.
  - You can even automate this further, and try to attach on login.
- Where is my mouse?!!!
  - Use `shift+click` to highlight with your mouse.
  - May want to bring the current *pane* to full-screen temporarily with `<cmd-seq>+Z`.
  - `<cmd-seq>` is `ctrl+B` by default, but can change it.
  - Un-fullscreen with another `<cmd-seq>+Z`
- Others exist, such as `terminator` and `screen`.

Further `tmux` Customization

- Configurations go in `~/.tmux.conf`.
- Save your layouts with `teamocil`!
  - `gem install teamocil`
  - Visit their page for how to set things up: http://www.teamocil.com/
- First run `tmux`, then launch `teamocil <name>`.

Closing a Git Branch

Closing a Branch

- AFTER you have merged a branch in and are ready to get rid of it, it is a good idea to "archive" it before deleting the branch entirely.
- You still have the history if you don't do this, but it is easier to restore / recover if you need to.

```
# http://stackoverflow.com/a/10243236/3814202
# create a "tag"
>>> git tag archive/<branchname> <branchname>
#     e.g.: archive/lec13_csv lec13_csv
# by default, they do not get pushed online. but we can:
>>> git push origin archive/<branchname>
#              e.g.: archive/lec13_csv
# now delete the branch locally, and on the remote
>>> git branch -d <branchname> # note: is lower case!!!
>>> git checkout master
>>> git push origin --delete <branchname>
```

References I

[1] B. Abrahao, H. Abu-Libdeh, N. Savva, D. Slater, and others over the years.
Previous cornell cs 2043 course slides.