

Week 1 - Introductions, Unix and Git

IBS 796 - Advanced Python Programming in Bioinformatics (Python)

8/29/2016

Robert Petit

Administrative Announcement

- No class next week.
- Next class is on September 12th.

Today's Outline

1. Introductions
2. Goals/Overview of the Course
3. Getting Setup
4. Basic Unix Commands
5. Git/GitHub

Introductions

Around the Room

1. Who are you?
2. Coding Experience?
3. Hopes/goals for this class?

Goals/Overview of the Class

Goals of this Course

- Specifically
 - Make the terminal you most used application
 - Become a fellow Python developer!
 - Getting first hand experience in microbial genomics
- Broadly
 - Becoming comfortable as a developer
 - Properly setting up your problem
 - Efficiently scouring the infinite abyss of Google

What this course is (and is not)

- It is...
 - Python based
 - Application based
 - Comprehensive
 - An open forum
- It is not...
 - A computer science course
 - A bioinformatics course
 - Written in stone

Weekly Class Outline

- Class demonstrations of previous week's assignment
 - Volunteer based or random order
- **Short** set of slides helpful for next assignment
- A set of homework problems to work through

Weekly Assignments

- Multiple parts, progressive in difficulty
- Collaboration is strongly encouraged!
 - Be sure you can explain the code though!
 - GitHub is great for collaboration.
- Easily demonstrated to the class
 - Make sure laptop works with projector!

Expectations and Grading

- “Show up and do your best.” - Dave Cutler

Getting Setup

A few requirements...

- You will need:
 - A laptop
 - With admin privileges
 - Python
 - Version 2.7
 - Preferably in Unix
 - GitHub account

Mac and Linux Users

- You are set... just know how to get to your terminal!

- Verify Python 2.7 is default:

```
rpetit3-mac:~ rpetit$ python --version  
Python 2.7.12
```

Windows 7 Onwards Users

- Windows 10
 - Use “Unix Services in Windows”
 - Git Bash/Console/cmdr setup
- Windows 7, 8
 - Take advantage of free upgrade (if applicable)
 - Git Bash/Console/cmdr setup
 - Set up a virtual machine
- Older Windows
 - Might be time to upgrade?
- If you haven't figured it out yet, see me after class!

Text Editors

- Too each their own!
 - Pick one get comfortable with it.
 - [Many to choose from](#)
- GUI
 - Sublime Text 3 (cross platform)
 - Notepad++ (Windows)
 - TextWrangler (or BBedit) (Mac)
- Terminal
 - emacs
 - vim (vi)
 - nano

Basic Unix Commands

Basic Unix Commands

- ls
- cd
- pwd
- more & less
- head & tail
- cat & zcat
- chmod
- man
- curl/wget
- ssh
- scp
- screen
- python
- exit

Git and GitHub

Git (/ɡɪt/) is a version control system that is used for software development and other version control tasks. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

- Wikipedia ([https://en.wikipedia.org/wiki/Git_\(software\)](https://en.wikipedia.org/wiki/Git_(software)))

The name "git" was given by Linus Torvalds when he wrote the very first version. He described the tool as "the stupid content tracker" and the name as (depending on your mood):

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh*t": when it breaks

THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



Those Shell Commands...

```
mpetit3@RP3-DESKTOP MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
        [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

The more common ones

- clone
 - Makes a local copy of a repository
- add
 - Marks a file to be included in the repository
- pull
 - Updates the repository to the latest changes
- commit
 - Creates a checkpoint (or version) of changes
- push
 - Updates the remote repository

GitHub

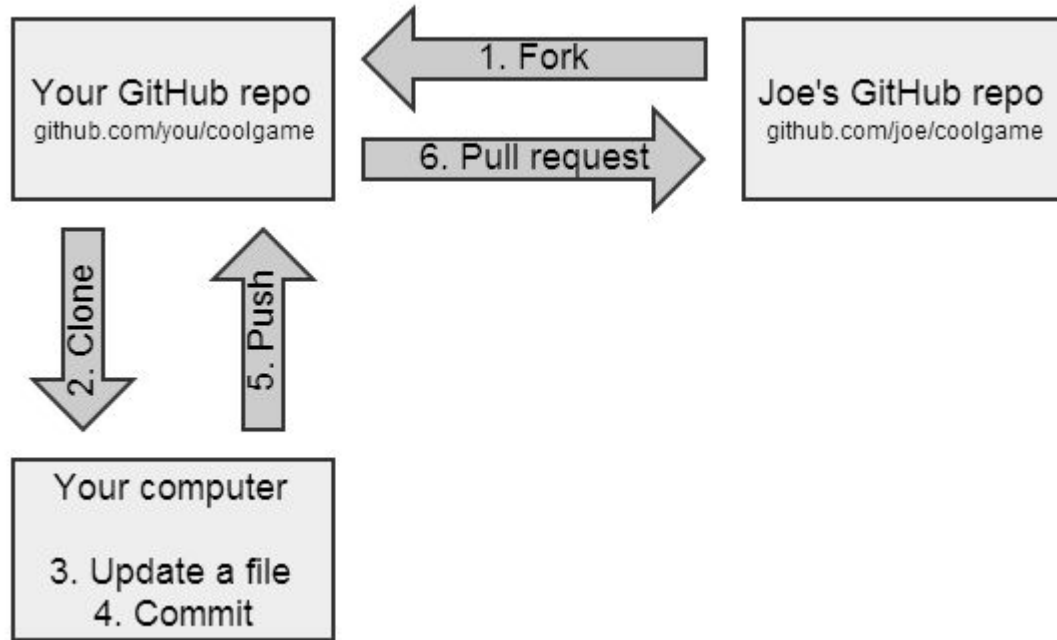
- git repository hosting site
- Most “social” of the git sites
 - Watch, Star, Comments, Issue Tracking, etc...
 - Individuals vs Organizations
- Pro-open source
 - Public repos are free
 - Private repos cost money
- [GitHub Desktop](#)
 - Utility to help visualize working with git



GitHub - Terminology

- GitHub allows collaborative coding, which is not necessarily built into git
- Fork
 - Makes a personal copy of some else's repository
- Pull Request
 - Allows you to submit changes back to another person's repository

Clone/Push vs Fork/Pull Request



GitHub Education

- Allows free discounts to GitHub services
 - Mostly access to private repos
- Applicable to students and researchers
 - Students (individual) get renewable 2 year discount
 - Researchers (organization) get free for life discount for their lab
- Access to the [GitHub Education Pack](#)
- [Request Your Discount!](#)



Next Week's Assignment

- Play with git
 - Create a new repo on GitHub
 - clone, commit, push, pull, add, etc...
 - BROWNIE POINTS: Investigate setting up your .gitconfig
- Create a personal [GitHub Page](#)
 - Read about GitHub Pages
 - Create repo to act as your accounts GitHub page
 - Clone locally
 - BROWNIE POINTS: Investigate setting up SSH-keys
 - Make some changes, commit them and push them
 - Be ready to show it all off September 12th!
 - Page and commit history
- [Explore GitHub](#)
 - Fork a repo you are interested in
 - Can be anything
 - BROWNIE POINTS: Submit a pull request fixing an issue