Version Control with Git

Using git as part of your daily workflow

Introduction

What is version control?

- a tool to:
  - back-up files
  - save history of changes
  - collaborate and combine changes easily

Why version control?

- to address the following common situations:
  - avoid scripts names _final_final2.py
  - your code used to work and now it doesn't and you want to go back to the working version
  - recreate figures you made 2 years ago
  - find where you introduced a bug
  - avoid accidentally over-writing changes someone else made to the code
  - make your process transparent to others
  - make your code easy to share
  - avoid updating the wrong version of the code
  - know what changes you made when

The Workflow of Local Git

The Workflow of Distributed Git

Telling Git who you are

- type into the shell
  - git config --global user.name “Your Name”
  - git config --global user.email email@address
  - git config --global color.ui auto
- you only have to do this once per machine

git config --list

The Basics

Create a repository

- typing `git init` in any directory will start a repository
  - `mkdir swc_test_repo`
  - `cd swc_test_repo`
  - `git init`
- creates a .git folder with repository information

Start tracking a file

- `git add filename`
- start tracking a file

Help

- git status
  - tells you the status of all files (tracked and untracked) in a directory
- git help or git help command
  - tells you how something works
  - lists possible git commands

Exercise 1:

1. in swc_test_repo create a file called names.txt
2. if you haven't already, create a repository
3. start tracking the file names.txt
4. what is the status of names.txt ?

- review:
  - git init - create repository
  - git add filename - start tracking
  - git status - what is the status of my file
  - git help - help me please

Staging Area

- setting the stage for a commit

Committing changes

- take a picture of your staging area
- `git commit -m "detailed commit message"`
- what if you forget -m? got to your default editor (usually vi), write message, save and close

Don’t let this be you

Exercise 2

1. if you have not already done so, commit names.txt from your staging area to your local repository
2. add the name of someone in the class to names.txt and save
    1. what is the status of names.txt?
    2. Add names.txt to your staging area
    3. what is the status of names.txt
    4. commit names.txt to your local repository
    5. what is the status of names.txt

Cheat sheet:
- git add filename
- git commit -m “message”
- git status
- git help

Undoing Mistakes:

- un-stage a file
  - `git reset HEAD filename`
- un-modify a file
  - `git checkout -- filename`

Exercise 3

1. modify names.txt and save your changes
2. add names.txt to your staging area
3. remove names.txt from your staging area
4. unmodify names.txt

Cheat sheet:

- git add filename
- git commit -m “message”
- git status
- git help
- git reset HEAD filename
- git checkout -- filename

Viewing differences

- everything:
  - `git diff`
- a single file
  - `git diff filename`
- + added since last staging
- - removed since last staging/commit
- compares current file to file in staging area or last committed file

Exercise 4:

1. modify names.txt and save your changes
2. use git diff to find your changes
3. stage your changes
4. run git diff again, do you get a different output?
5. commit your changes (don't forget your commit message)

Cheat sheet:

- git add filename
- git commit -m "message"
- git status
- git help
- git reset HEAD filename
- git checkout -- filename
- git diff filename

Viewing your commit history

- all history:
  - `git log`
- last 2 entries:
  - `git log -2`

Shell commands in git

- git mv
- git rm

Exercise 5

1. create a file
2. start tracking your file
3. commit your file
4. modify your file
5. stage your file
6. commit your changes

Cheat sheet:

- git add filename
- git commit -m “message”
- git status
- git help
- git reset HEAD filename
- git checkout -- filename
- git diff filename

Advanced Git

Why branch?

Actually modify names.txt on the testing branch

Branching

- create a branch called testing
  - git branch testing
- ask git which branch you are on:
  - git branch
- switch to the testing branch
  - git checkout testing

Exercise 6:

1. create a new file called learned.txt and type something new you learned today
2. before you add and commit the file:
    1. is it visible in your file system when you are on the master branch?
    2. what about from the testing branch?
3. add and commit your file to your master branch
4. now is it visible in your file system on the master branch? on testing? why is this different from #2?
5. create a branch called exercise from master
6. move to exercise branch and modify learned.txt
7. add and commit learned.txt to the exercise branch
8. go to the master branch. Are your modifications to learned.txt visible?
9. modify learned.txt on your master branch. Before you commit your changes, try to move to the exercise branch. Can you? What error message do you get?
10. add and commit your changes to the master branch

Cheat sheet:

- git add filename
- git commit -m “message”
- git status
- git help
- git reset HEAD filename
- git checkout -- filename
- git diff filename
- git branch branch_name
- git checkout testing

Merging

http://pcottle.github.io/learnGitBranching/?NODEMO
git commit
git branch exercise
git checkout exercise
git commit
git checkout master
git commit
git merge exercise
git commit

Merging

- go to the branch you want to merge into
- git merge branch_you want_to_merge
- no conflicts: awesome
- conflicts:
  - resolve, add, commit
- merging doesn't delete or modify the merged branch
- delete merged branch
  - git branch -d branch_name

Exercise 7

1. merge learned.txt from the exercise branch to the master branch
2. resolve any conflicts
3. delete the exercise branch

Cheat sheet:

- git add filename
- git commit -m “message”
- git status
- git help
- git reset HEAD filename
- git checkout -- filename
- git diff filename
- git branch branch_name
- git checkout testing
- git merge branch_to_merge
- git branch -d branch_to_delete

Resources

- This presentation
- http://git-scm.com/book/en/
- http://pcottle.github.io/learnGitBranching/

The Workflow of Distributed Git

1. cd out of your swc_local repository
2. git clone --branch students https://github.com/justincely/2014-02-24-esac