

# Version Control with Git and GitHub & Good Coding Practices

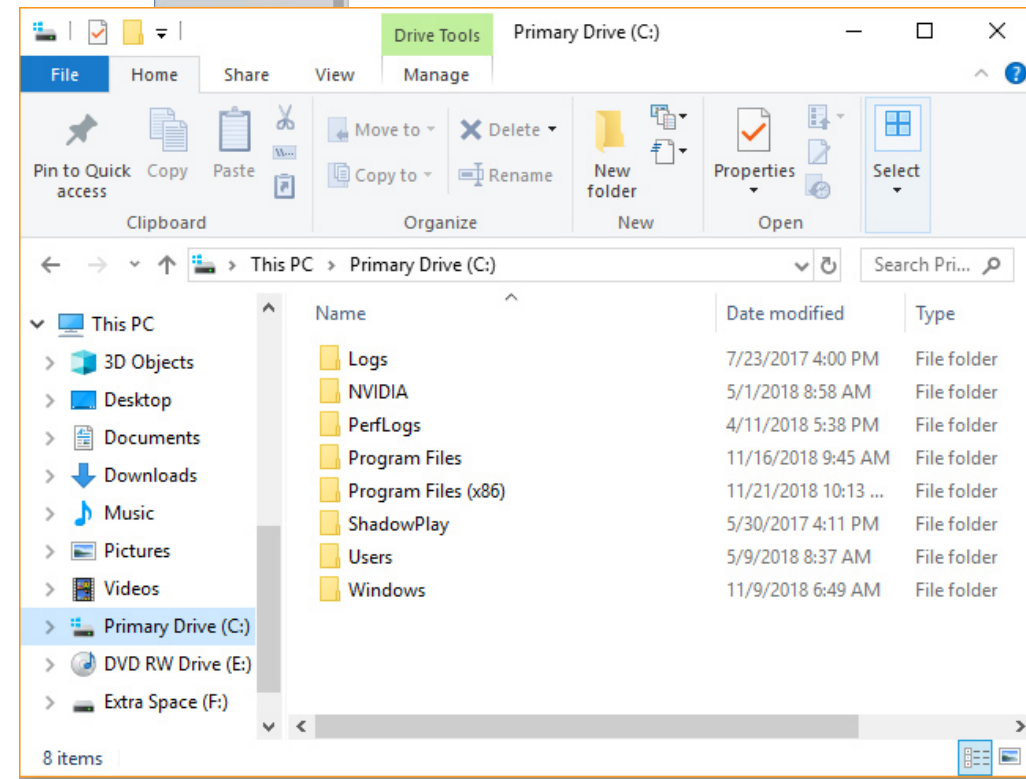
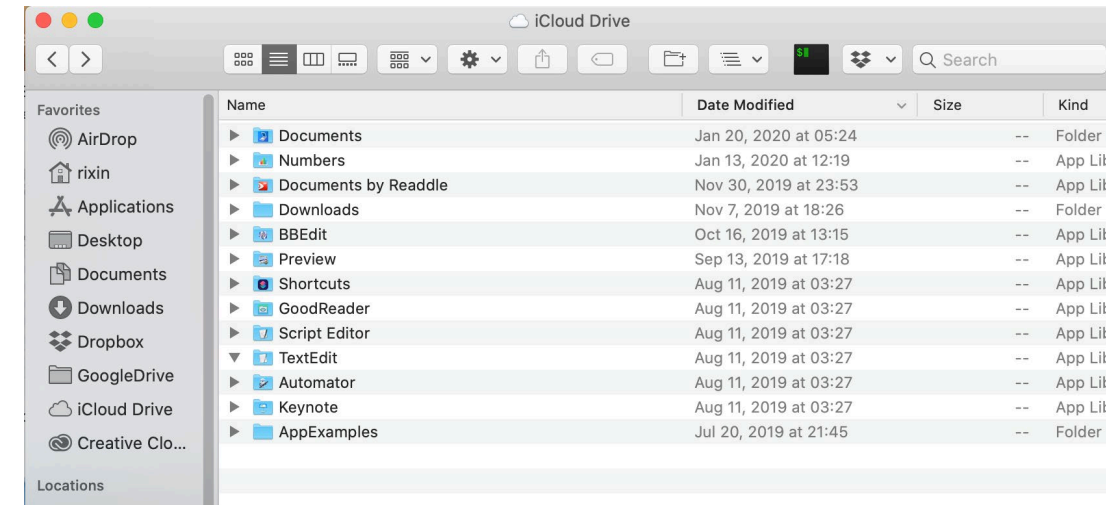
ASTR400B

Rixin Li

# Before we start... Notes for Using Commands in Linux

“Where you are” matters:

- Just like any file operations in Finder (macOS) or File Explorer (Windows)
- Commands in Linux may act differently at different places
- So, before executing a command: find out what is your current working directory / what is your current path: use **pwd**



# Before we start... Notes for Hidden Files in Linux

Any file whose name begins with a “ . ”

- Similar in macOS; though not true in Windows (has other settings)
- Use **ls -a** to view hidden files
- relative path to files
  - . = the current directory
  - .. = the parent directory

• Example:

```
ln -s ../astr400b/MW_000.txt ./MW_000.txt
```

- Note the hidden folder **.git** (we will get to this later)

```
rixin@nimoy:~/astr400b/ASTR400B_2020$ pwd
/home/rixin/astr400b/ASTR400B_2020
rixin@nimoy:~/astr400b/ASTR400B_2020$ ls
Homeworks Lectures README.md Syllabus.pdf
rixin@nimoy:~/astr400b/ASTR400B_2020$ ls -a
.  .. .git Homeworks Lectures README.md Syllabus.pdf
rixin@nimoy:~/astr400b/ASTR400B_2020$ ls -al
total 148
drwxrwxr-x 5 rixin rixin 4096 Jan 23 11:04 .
drwxrwxr-x 3 rixin rixin 4096 Jan 23 11:04 ..
drwxrwxr-x 8 rixin rixin 4096 Jan 23 11:04 .git
drwxrwxr-x 4 rixin rixin 4096 Jan 23 11:04 Homeworks
drwxrwxr-x 2 rixin rixin 4096 Jan 23 11:04 Lectures
-rw-rw-r-- 1 rixin rixin 129 Jan 23 11:04 README.md
-rw-rw-r-- 1 rixin rixin 126044 Jan 23 11:04 Syllabus.pdf
```

LS(1)	User Commands	LS(1)
<b>NAME</b> ls - list directory contents		
<b>SYNOPSIS</b> ls [OPTION]... [FILE]...		
<b>DESCRIPTION</b> List information about the FILES (the current directory by default). Sort entries alphabetically if none of <b>-cftuvSUX</b> nor <b>--sort</b> is specified.  Mandatory arguments to long options are mandatory for short options too.  <b>-a, --all</b> do not ignore entries starting with .		

# Version Control Systems (VCS)

One definition: “A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. ” (Wikipedia)

Return to Zero



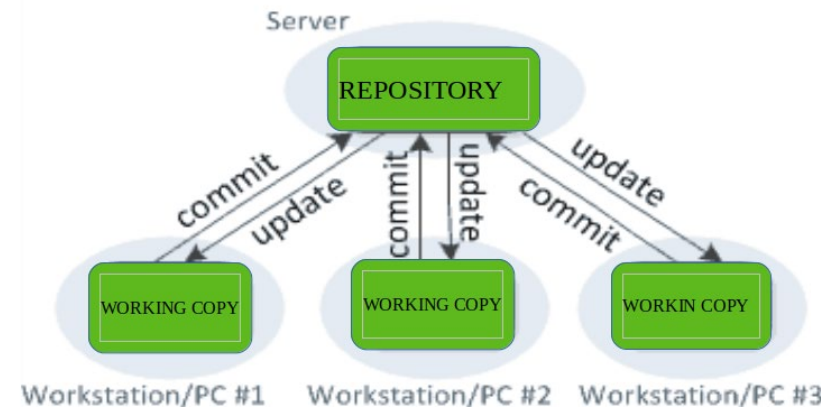
Reasons to use VCS:

- Any thoughts?

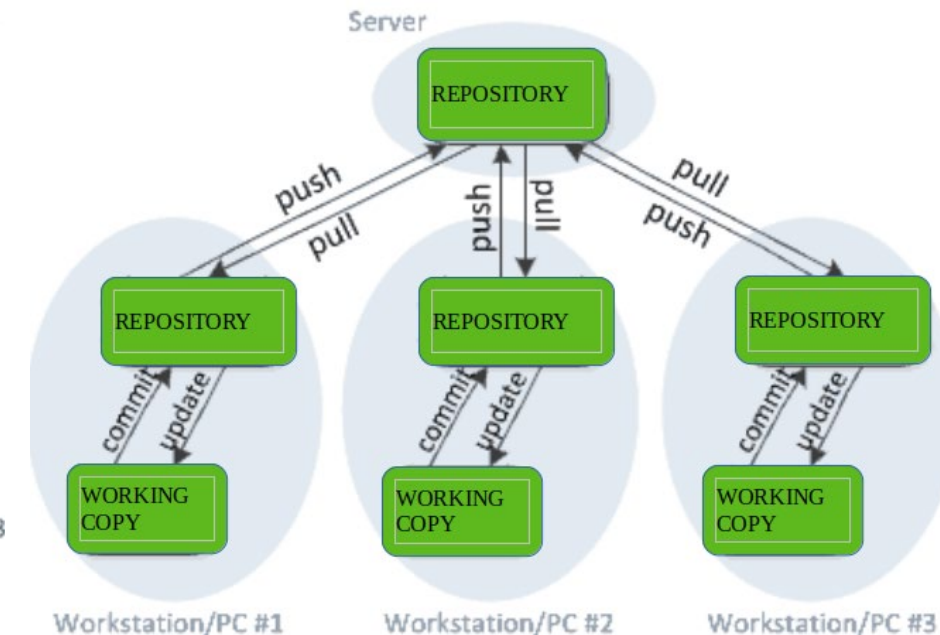
Types of VCS:

- Local VCS
- Centralized VCS
- Distributed VCS

Centralized version control



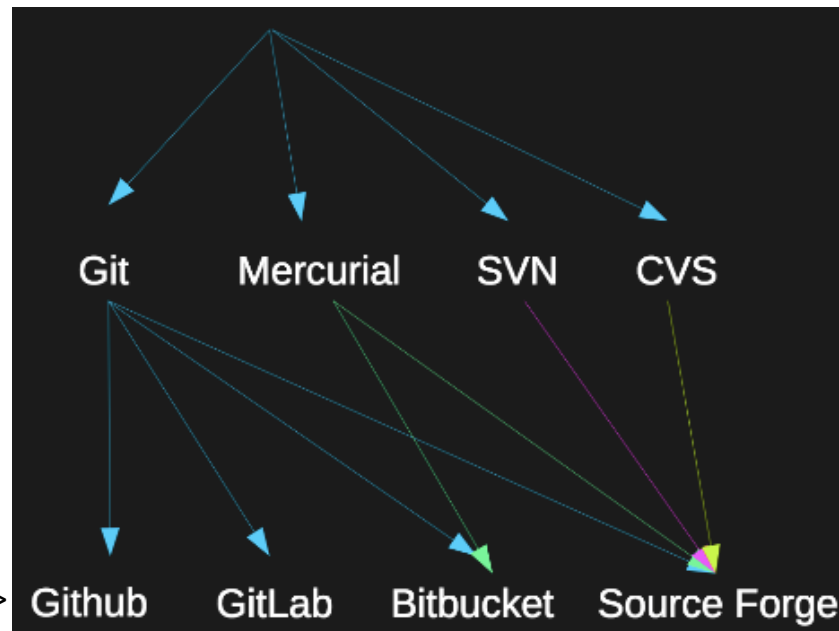
Distributed version control





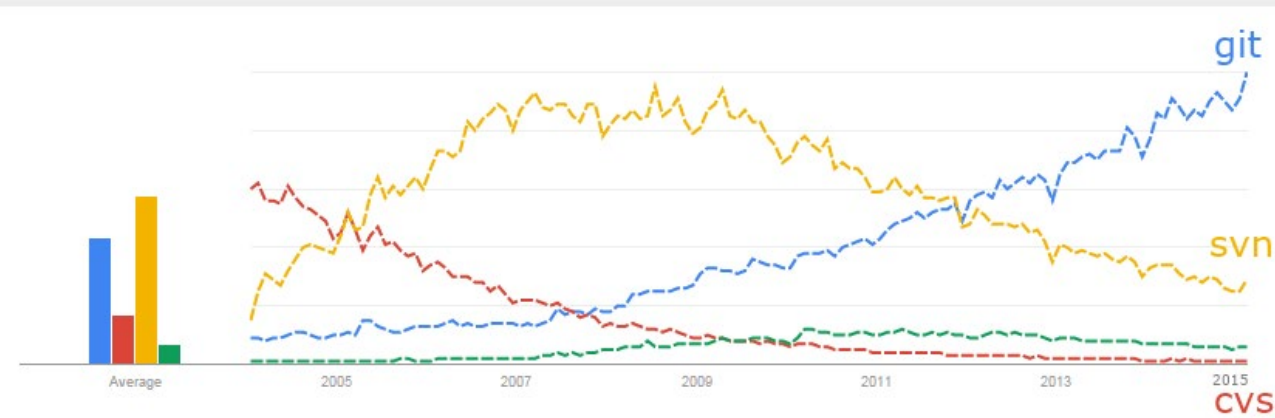
# How many of you have actively used Git/GitHub or any other version control systems?

various VCS =>



various hosting service for code =>

Interest over time



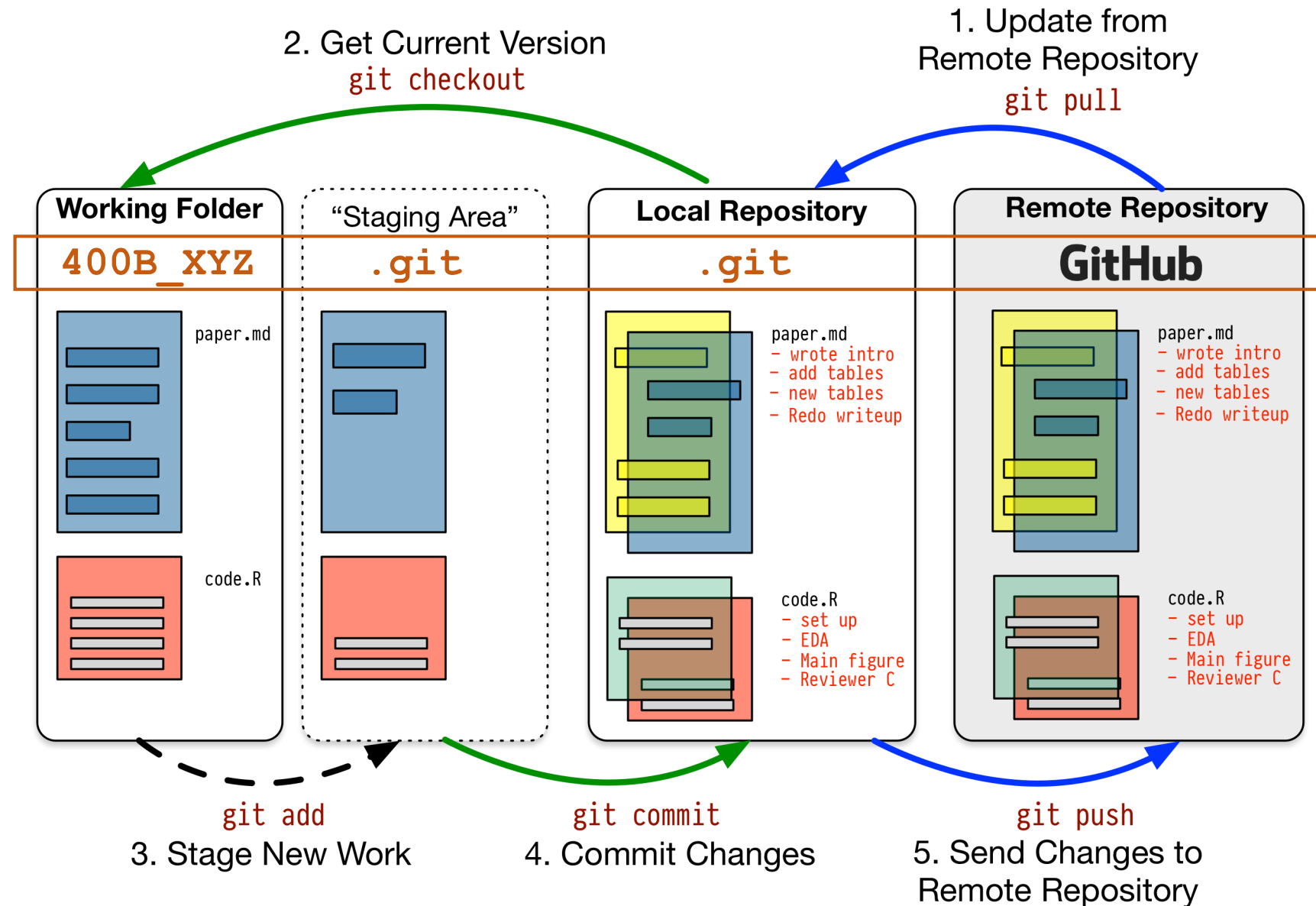
# Github features

- **Open source software makes science reproducible!**  
(Astronomy: LSST, DES, NASA)
- **Student pack** allows for unlimited public repositories and some private repositories (sign up with UA email)
- **Host web pages** (set up your personal webpage with [pages.github.com](https://pages.github.com))
- **Email notifications** about code changes made by collaborators
- **Issues:** Set reminders to yourself to fix bugs or report bugs for established software packages (astropy, numpy, etc.)
- **Documentation:** Set up all code repositories with README files detailing code specifications, dependencies, etc.

# A Typical Git Workflow & Glossary

Git is a **distributed** VCS

- You & collaborators can have as many working folders as you want anywhere (e.g., both on your laptop and Nimoy)
- For Step 1&2, you may also clone a repo by `git clone repo_link`
- You've tried Step 3-5



# In Practice: Check Status First

- Check `git status` before working
- Local changes will be shown if any

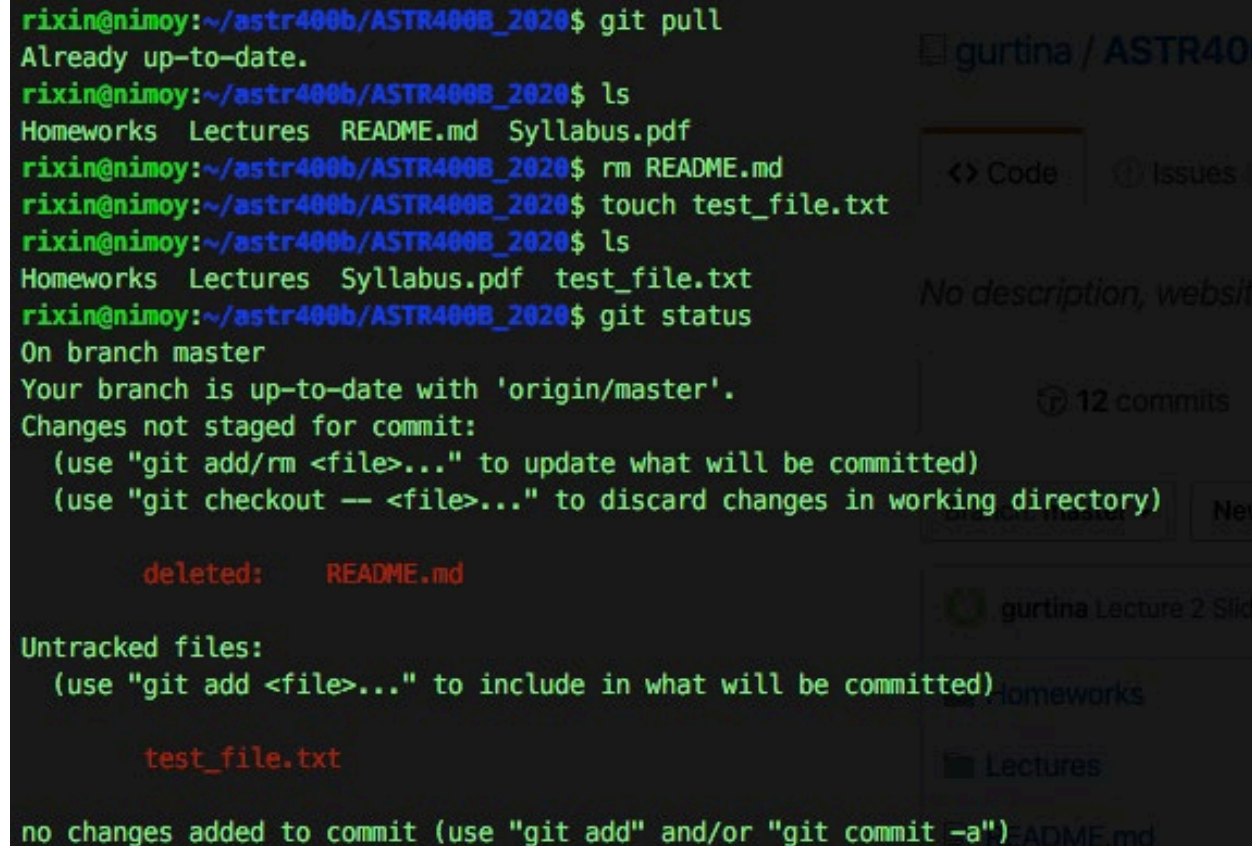
```
rixin@nimoy:~/astr400b/ASTR400B_2020$ git pull
Already up-to-date.
rixin@nimoy:~/astr400b/ASTR400B_2020$ ls
Homeworks  Lectures  README.md  Syllabus.pdf
rixin@nimoy:~/astr400b/ASTR400B_2020$ rm README.md
rixin@nimoy:~/astr400b/ASTR400B_2020$ touch test_file.txt
rixin@nimoy:~/astr400b/ASTR400B_2020$ ls
Homeworks  Lectures  Syllabus.pdf  test_file.txt
rixin@nimoy:~/astr400b/ASTR400B_2020$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test_file.txt

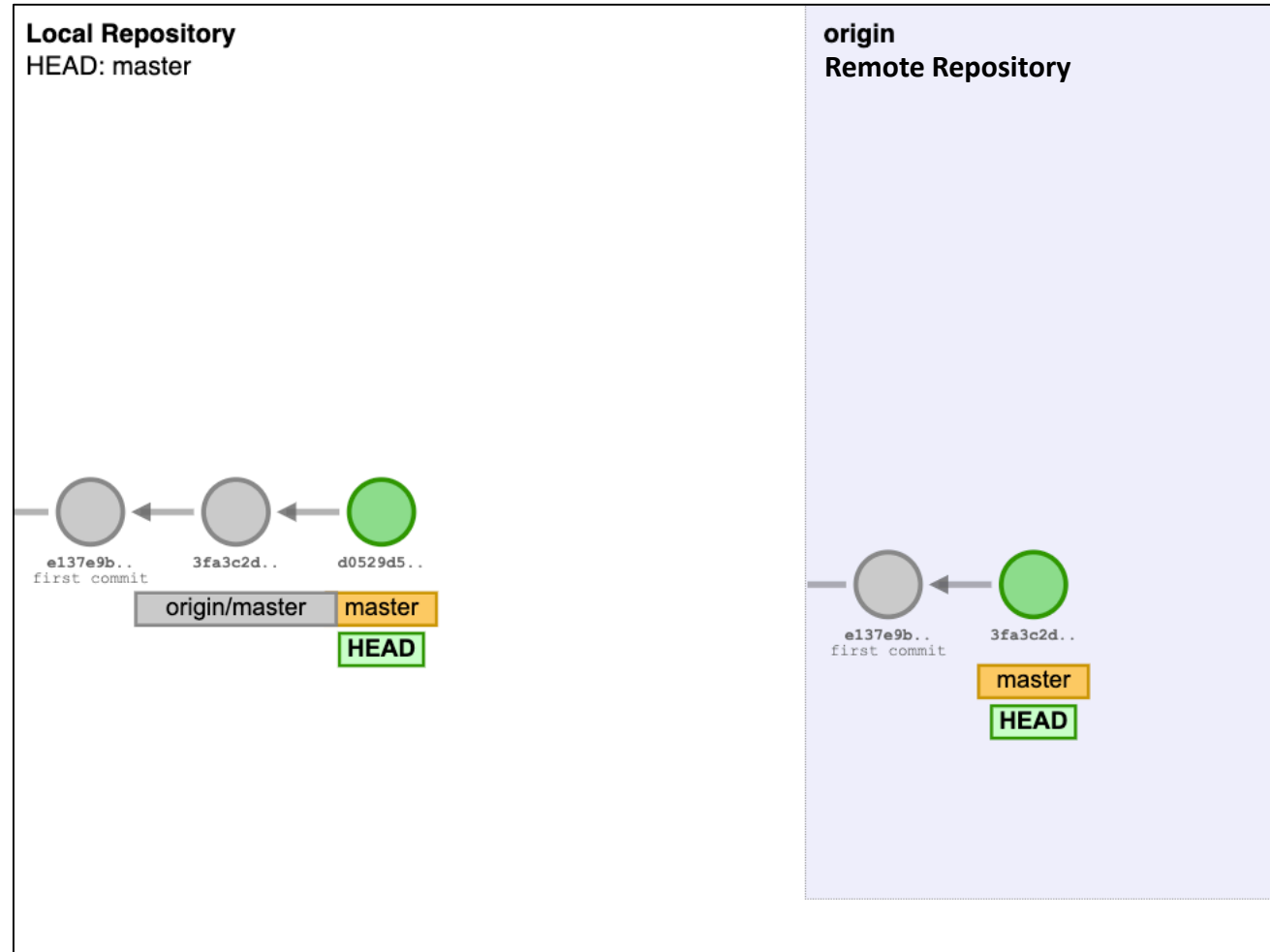
no changes added to commit (use "git add" and/or "git commit -a")
```





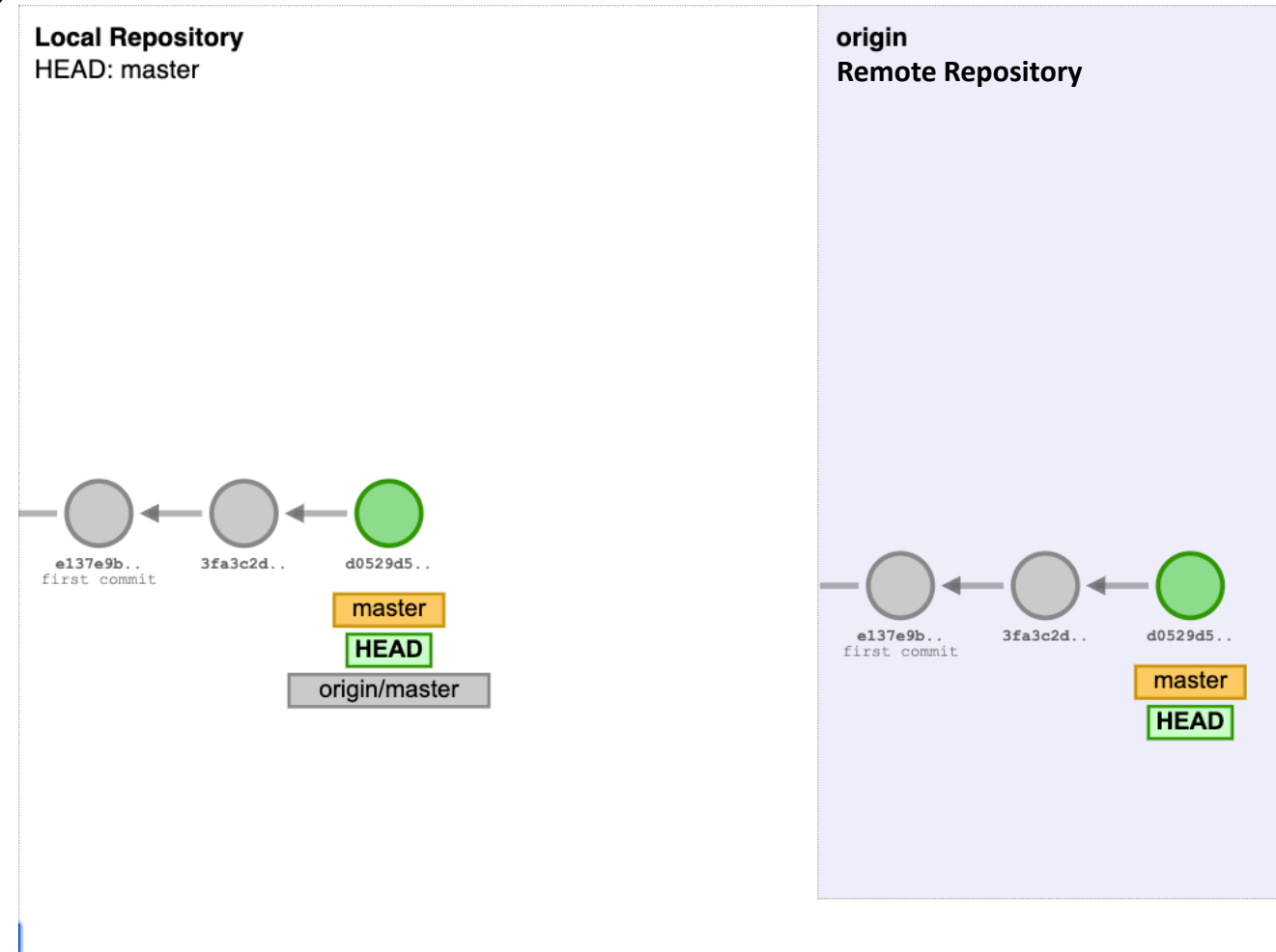
# In Practice: Commit with Good Descriptions

- Check `git status` before working
- Differentiate local and remote
- Every operation before `git push` takes place locally
- After making *enough* changes (e.g., finish one homework), `git add` your changes and then `git commit` your changes
- **Commit with descriptions** so you always know what has changed  
`git commit -m "finished HW2"`



# In Practice: Commit with Good Descriptions

- Check `git status` before working
- Differentiate local and remote
- Every operation before `git push` takes place locally
- After making *enough* changes (e.g., finish one homework), `git add` your changes and then `git commit` your changes
- **Commit with descriptions** so you always know what has changed  
`git commit -m "finished HW2"`



# In Practice: Revert Changes before Committing

- How to revert changes depends on if you have already committed
- Before
  - `git reset --hard origin/master`
  - `git clean -f`
- This will destroy local modifications and remove untracked files

```
rixin@nimoy:~/astr400b/ASTR400B_2020$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    README.md

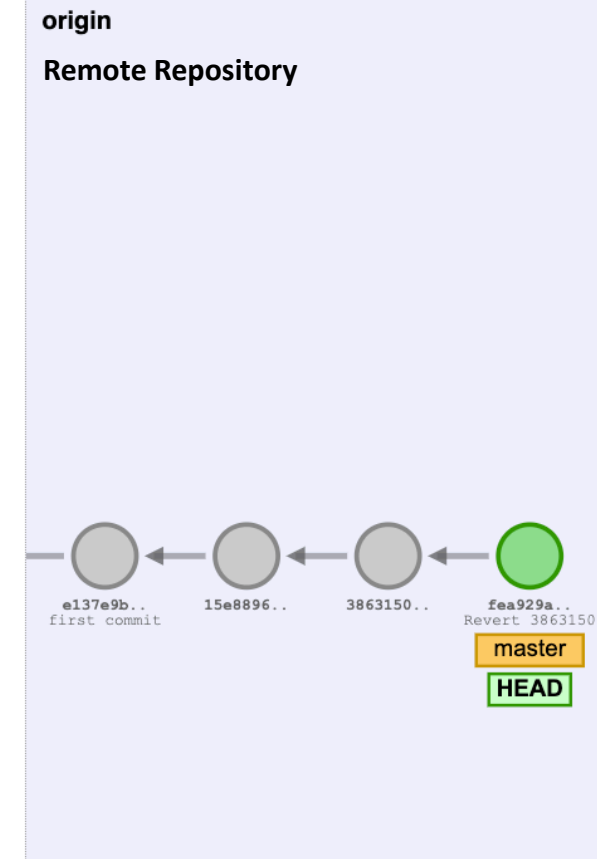
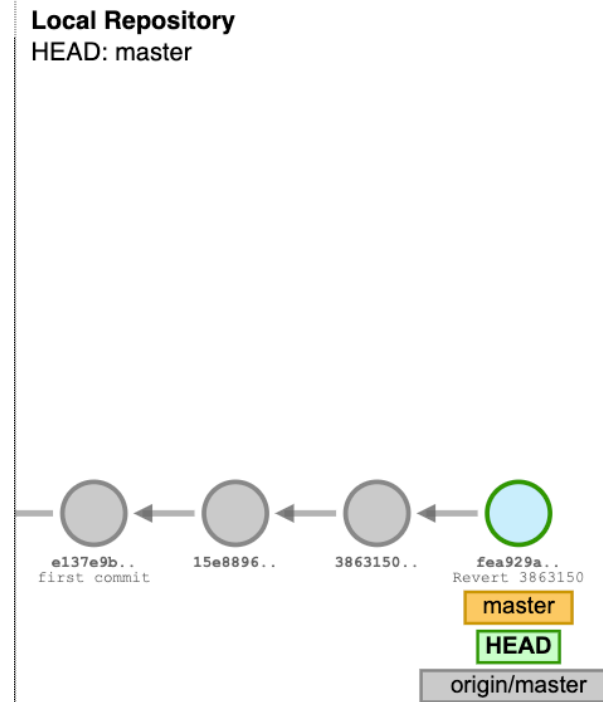
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test_file.txt

no changes added to commit (use "git add" and/or "git commit -a")
rixin@nimoy:~/astr400b/ASTR400B_2020$ git reset --hard origin/master
HEAD is now at 2d34869 Lecture 2 Slides
rixin@nimoy:~/astr400b/ASTR400B_2020$ git clean -f
Removing test_file.txt
rixin@nimoy:~/astr400b/ASTR400B_2020$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

# In Practice: Revert Changes after Committing

- How to revert changes depends on if you have already committed
- Before
  - `git reset --hard origin/master`
  - `git clean -f`
- This will destroy local modifications and remove untracked files
- After (it's tricky)
  - `git revert HEAD`
  - `git push`
- This will in fact create another commit



# Resources to learn Git

- <https://guides.github.com/>
  - Official Guides from GitHub



## Understanding the GitHub flow

GitHub flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly. This guide explains how and why GitHub flow works.

🕒 5 minute read



## Hello World

The easiest way to get started with GitHub. In this guide you'll complete a time honored "Hello World" exercise, and learn GitHub essentials.

🕒 10 minute read



## Getting Started with GitHub Pages

GitHub Pages are a great way to showcase some open source projects, host a blog, or even share your résumé. This guide will help get you started on creating your next website.

🕒 10 minute read



## Git Handbook

Learn about version control—in particular, Git, and how it works with GitHub.

🕒 10 minute read



## Forking Projects




## Be Social



# Resources to learn Git

- <https://guides.github.com/>
  - Official Guides from GitHub
- <https://git-scm.com/doc>
  - Videos, also doc in other lang


 **git** --distributed-is-the-new-centralized

Search entire site...


AboutDocumentationReferenceBookVideosExternal LinksDownloadsCommunity

## Documentation


### Reference

**Reference Manual**

The official and comprehensive **man pages** that are included in the Git package itself.


 Quick reference guides: [GitHub Cheat Sheet](#) (PDF) | [Visual Git Cheat Sheet](#) (SVG | PNG)

### Book


**Pro Git**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).


### Videos

**Git Basics:**  
What is Version Control?  
Episode #1


**What is Version Control?**  
Length: 05:59

**Git Basics:**  
What is Git?  
Episode #2

**What is Git?**  
Length: 08:15

**Git Basics:**  
Get Going with Git  
Episode #3

**Get Going with Git**  
Length: 04:26

**Git Basics:**  
Quick wins with Git  
Episode #4

**Quick Wins with Git**  
Length: 05:06

# Resources to learn Git

- <https://guides.github.com/>
  - Official Guides from GitHub
- <https://git-scm.com/doc>
  - Videos, also doc in other lang
- <https://try.github.io/>
  - Interactive learning by using Git commands; great visualization of the version tree

## Resources to learn Git

### Learn by reading

#### Git Handbook

Git, GitHub, DVCS, oh my! Learn all the lingo and the basics of Git.

#### Cheat Sheets

Keep these handy! Reference sheets covering Git commands, features, SVN migrations, and bash. Available in a multiple languages.

### Learn by doing

#### Learn Git branching

Try Git commands right from your web browser. Featuring some of your soon-to-be favorites: branch, add, commit, merge, revert, cherry-pick, rebase!

#### Visualizing Git

Look under the hood! Explore how Git commands affect the structure of a repository within your web browser with a free explore mode, and some constructed scenarios.

#### Git-It

You've downloaded Git, now what? Download Git-It to your machine and you'll get a hands-on tutorial that teaches you to use Git right from your local environment, using commands on real repositories.

### Looking for GitHub Training?

#### ✓ GitHub Learning Lab

Get the skills you need without leaving GitHub. GitHub Learning Lab takes you through a series of fun and practical projects, sharing helpful feedback along the way.

#### ✓ Professional training

Whether you're just getting started or you use GitHub every day, the GitHub Professional Services Team can provide you with the skills your organization needs to work smarter.



# Resources to learn Git

---

- <https://guides.github.com/>
  - Official Guides from GitHub
- <https://git-scm.com/doc>
  - Videos, also doc in other lang
- <https://try.github.io/>
  - Interactive learning by using Git commands;  
great visualization of the version tree
- <https://teamtreehouse.com/library/introduction-to-git>
- (etc.)

# Exercises (30 mins) => see solutions in the class repo

---

1. Clone the class repo so you can pull updates in the future to get slides and HWs  
Command: `git clone https://github.com/gurtina/ASTR400B_2020.git`
2. Now let's work on your homework repository.
  - 1) Create a folder named "git\_test" under your home directory
  - 2) Clone your own HW repo in the directory "git\_test" you just created
  - 3) Now you have a new working repo, `cd` into this copy of repo
  - 4) Create a text file "lecture3.txt" and write "Lecture 3 test" into it
  - 5) Add, commit, and push your changes ("`git add .`" will add all changes)
  - 6) Create a directory "L3" and move "lecture3.txt" into it
  - 7) Again, add, commit, and push your changes
3. Now go to your original homework repository folder that you did your homework and pull the changes you made inside "git\_test"
  - 1) Try to revert the last commit you just made if you have time

# Good Coding Practices (Readability)

---

## Comment!

- All of your code should have comments. Why?
  1. You may need to look back at your code after some time and comments will help jog your memory.
  2. Others may need to read your code to collaborate with you, grade your assignments, or to build some extension of what you've written.



# Good Coding Practices (Readability)

---

## Organize your code logically

- Section your code so that related lines can be found in the same place, separated by line break from other parts of your code.
- A good general order to stick to for this class is:
  - A short comment on the goal of the code.
  - Imports
  - Functions
  - Read data
  - Do something with your data
  - Close and save your results.
- ***Good, final versions of your code should follow your thought process, but not necessarily the order in which you wrote it.***

# Good Coding Practices (Readability)

---

## Naming

- Do not rename variables in a single script!
  - example: `this_number = 2` --> `this_number = 2*2`
  - instead: `this_number = 2` --> `new_number = 2*2`
- lowercase: modules, functions, packages
- constants: all caps
- class names and exceptions: capital words (CapWords)

# Good Coding Practices (Readability)

## White spaces

- Immediately inside parentheses, brackets or braces.

Yes: `spam(ham[1], {eggs: 2})`

No: `spam( ham[ 1 ], { eggs: 2 } )`

- Between a trailing comma and a following close parenthesis.

Yes: `foo = (0,)`

No: `bar = (0, )`

- Immediately before a comma, semicolon, or colon:

Yes: `if x == 4: print x, y; x, y = y, x`

No: `if x == 4 : print x , y ; x , y = y , x`

- Immediately before the open parenthesis that starts the argument list of a function call:

Yes: `spam(1)`

No: `spam (1)`

- Immediately before the open parenthesis that starts an indexing or slicing:

Yes: `dct['key'] = lst[index]`

No: `dct ['key'] = lst [index]`

- More than one space around an assignment (or other) operator to align it with another.

Yes:

```
x = 1
y = 2
long_variable = 3
```

No:

```
x           = 1
y           = 2
long_variable = 3
```

# Good Coding Practices (Readability)

Most-used Style Guide:

PEP8 <= use it as a reference

<https://www.python.org/dev/peps/pep-0008/>

## PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

### Contents

- [Introduction](#)
- [A Foolish Consistency is the Hobgoblin of Little Minds](#)
- [Code Lay-out](#)
  - [Indentation](#)
  - [Tabs or Spaces?](#)
  - [Maximum Line Length](#)
  - [Should a Line Break Before or After a Binary Operator?](#)
  - [Blank Lines](#)
  - [Source File Encoding](#)
  - [Imports](#)
  - [Module Level Dunder Names](#)