Open-Source Drilling Community (OSDC) – Technical Seminar Series

Introduction to Git /GitHub
Spring 2023

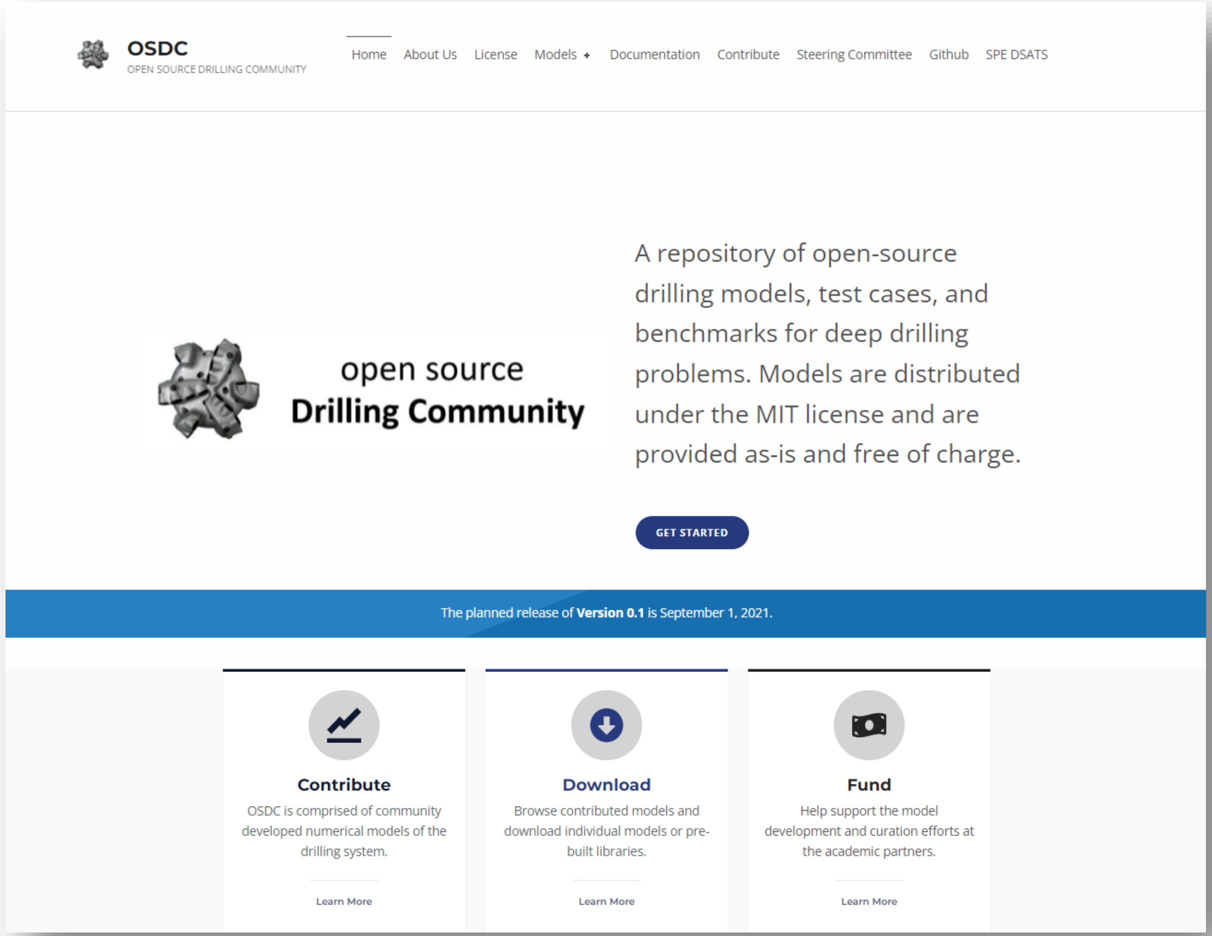
Eduardo Gildin
Petroleum Engineering Department
Texas A&M University

TEXAS A&M UNIVERSITY
Harold Vance Department of
Petroleum Engineering

Open-Source Drilling Community

- To join, visit the Open-Source Drilling Community Website: https://opensourcedrilling.org
  - Add your contact information to the Mailing List / Contribute tab

- MIT Open-Source License – All models, data, and test cases are freely available for academic and commercial use



A repository of open-source drilling models, test cases, and benchmarks for deep drilling problems. Models are distributed under the MIT license and are provided as-is and free of charge.

The planned release of **Version 0.1** is September 1, 2021.

Contribute
OSDC is comprised of community developed numerical models of the drilling system.

Download
Browse contributed models and download individual models or pre-built libraries.

Fund
Help support the model development and curation efforts at the academic partners.

SPE-208794-MS

Progress Toward An Open-source Drilling Community: Contributing And Curating Models

Roman J. Shor, **Shanti Swaroop Kandala** (University of Calgary), **Eduardo Gildin**, **Sam F. Noynaert**, **Enrique Z. Losoya**, **Vivek Kesireddy**, **Narendra Vishnumolakala** (Texas A&M University), **Inho Kim** (Mathworks), **James Ng** (Pason Systems Corporation), **Josh K. Wilson** (Scientific Drilling International), **Eric Cayeux** (NORCE), **Rajat Dixit** (ExxonMobil Services & Technology Pvt Ltd), **Gregory S. Payette** (ExxonMobil Upstream Research), **Ty Cunningham** and **Paul E. Pastusek** (ExxonMobil Upstream Integrated Solutions),

Workshop Objectives

- Main objective: how can you contribute to the OSDC GitHub?
  - Codes you upload → what will happen ?
  - Someone's code → can I contribute/modify/extend and share?
- Git/GitHub introduces the concept of version control and data/code repositories
  - You will see only the basic/concepts
- You will learn :
  - How to maintain version control
  - Create a data repository
  - Share data
  - Clone/branch repositories
  - Check and Create GitHub webpages

More Training:

- https://github.com/apps/github-learning-lab
- https://skills.github.com/
- https://github.com/topics/learning-lab

Workshop Series – Tentative Schedule*

- Feb 27. Eduardo: GitHub Part 1
- Mar 20. Eduardo: GitHub Part 2 → more features
- Mar 27. Eric & Gilles: Rheological Model Calibration from Couette Rheometer and pipe rheometer
- Apr 24. Eric & Gilles: Packaging Open-Source Code as a Microservice for seamless interoperability across multiple programming languages

* Subject to change → keep vising the OSCD webpage for more info

What is Git/GitHub?

- Git → Git is a version control system (VCS) to keep track of changes to files, projects and data over time.
  - You have already used some form of VCS !!!!
- GitHub → GitHub is a hosting service for Git repositories. Simplifies the process of sharing code between developers.

GitHub & GitHub Enterprise are independent
→ https://it.tamu.edu/services/websites-applications-and-software/design-development-and-administration/github/
→ **https://github.tamu.edu/**

Other common examples of version control

- These are very primitive examples of version control
  - File naming – (e.g., project_v1.doc, project_v2.doc,… project_vfinal.doc)
  - Microsoft Word : Track changes
  - Adobe Photoshop: History (see changes to an image)
  - Undo: Crtl+Z (Windows), Cmd-Z (Mac)
- But, in real cases (big projects), there is no substitute for a VCS (like Git)

What is Revision Control (VCS)?

- Version Control System (VCS) or Revision Control or Source Control (SCM*) describes the process of monitoring changes in sets of information
  - Especially text changes (like code) →Source code management tool
- In the software development process, revision control is the management of changes made over time. These changes can be to source code, project assets, or any other information that goes into the finished product.
  - The collection of revisions and their metadata is called a **repository** or **repo**. → usually hosted on a networked server.

*SCM – source code management

Repository

- The sets of information are usually documents, source codes, large web repositories or alike → any file
- The set of all information (usually files) under revision control makes a *repository*
- The repository represents a step-by-step chronological record of every change made to help project managers revert all or part of the project to a previous state if necessary.
  - A set of changes to a single or multiple pieces of information (files) constitute a revision of the repository, and in the case of software a set of revisions defines a new version.

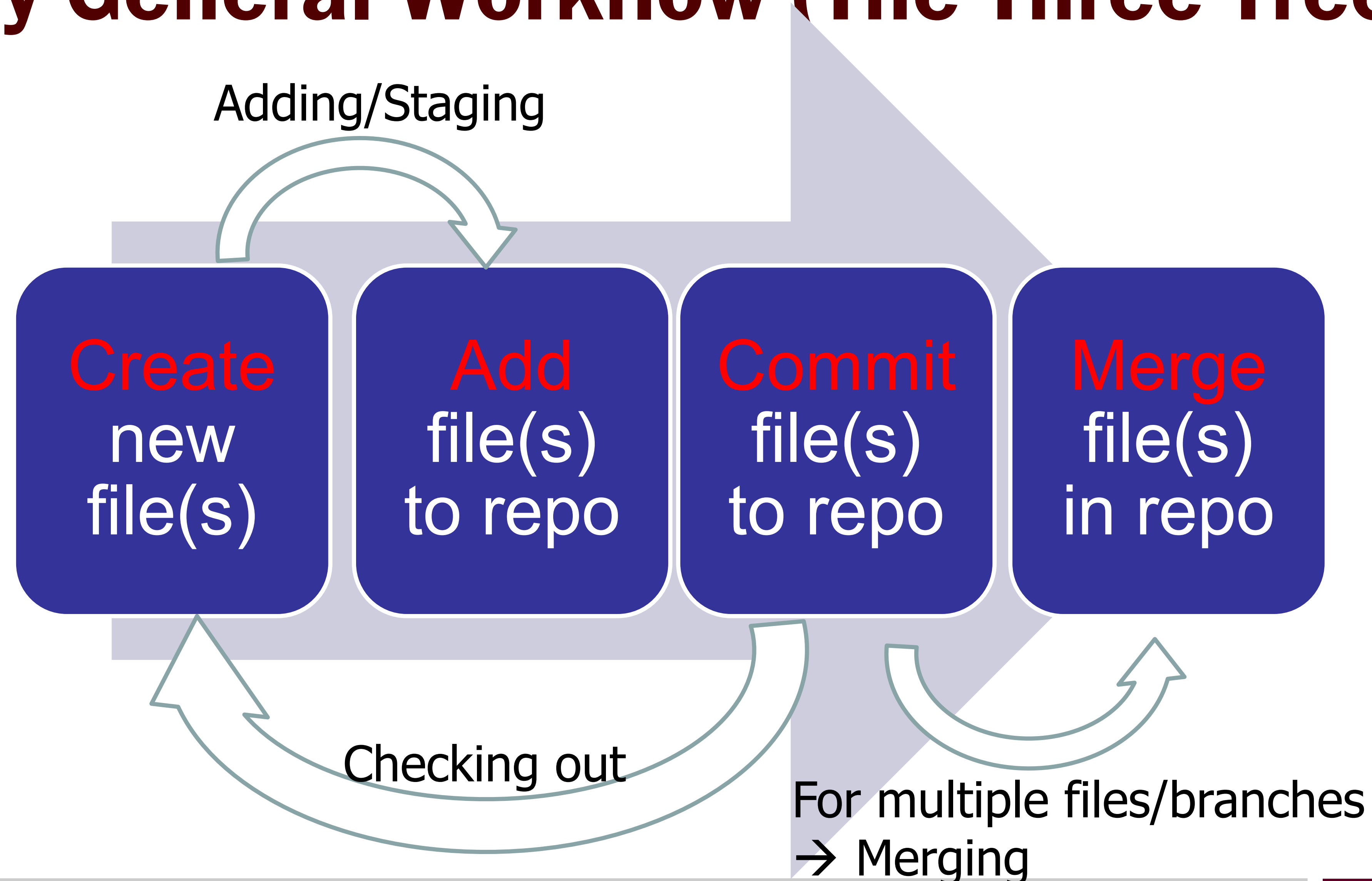
Source: Dr. Jens Saak (Max Planck Institute)

Revision Purposes

- The revisions get assigned a unique name that may be an identification number or a human readable text.
- The main purposes of revision control can be summarized as the following items:
  - *Logging of changes*: at any later stage of development of the information it is clear which change has been added by whom and when this happened.
  - *Recovery* of earlier states of the single pieces of information: accidental or erroneous changes can be identified and rolled back.
  - *Archiving*: It is possible to get back to each state of the set of information, e.g. to make computational results reproducible.
  - *Coordination* of joint work on the information by several collaborators.
  - *Parallel* development of multiple branches of the information with the possibility to merge single branches back to a main development stream.

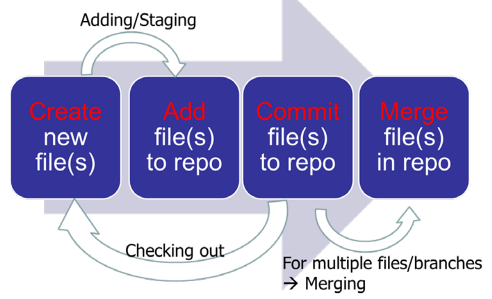
Source: Dr. Jens Saak (Max Planck Institute)

Very General Workflow (The Three Trees)



For multiple files/branches
→ Merging

Very General Workflow

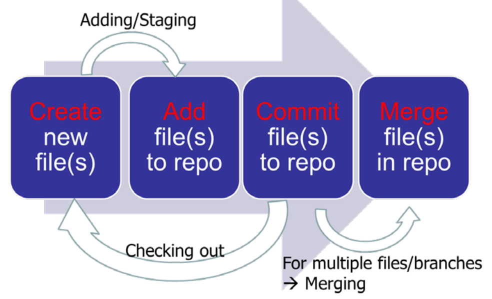


- If the developer has created a new file that should become part of the project, the file must be **added** to the repository. The file is uploaded to the repository, and anyone else working on the project can see and use the file.

- If the developer wants to edit a file that is already part of the project, the file must be **checked out**. The act of checking out downloads the desired revision of the file to the developer's local version of the project. Usually, the revision that a developer wants to edit is the most recent revision: this revision is known as the "head".

- After the developer edits the file locally and is ready to add it to the official version of the project, the file can be checked in. This action is also known as **making a commit**. The developer is asked to write a summary of what changes were made and why. These comments, with the updated version of the file, are uploaded to the repository.

Very General Workflow



- If someone else has checked in revisions to the same file since the last time the developer checked it out, the system announces that **there are conflicts**. It calculates the differences line-by-line, and the developers who made the changes must agree upon how their individual changes should **be merged**. **The merging** is usually done manually: the developers compare the conflicting versions and decide how to resolve them into one document.

- If there are no conflicts, **the new version is updated in the repository**, and the entire project receives a new revision number, permanently and uniquely identifying its current state.

Some Historical Perspective (1/3)

- There are 5 important version control system before Git
  - Source Code Control System (SCCS)
    - 1972 – closed source (from AT&T)
    - UNIX only – free with UNIX – very popular
    - Keep the original doc and save a snapshot of the changes
  - Revision Control System (RSC)
    - 1982, Multiple platform / PC's
    - Faster → smarter storage strategy
    - Stored latest version and sets of changes (reverse) → faster

V1 → V2→v3→ V4→ V5

snapshot snapshot snapshot

Issues: Work with individual file one at a time! → no project tracking

V1 → V2→v3→ V4→ V5 Usually wants the current version

Source: Kevin Skoglund

Some Historical Perspective (2/3)

- Evolution!
  - Concurrent Versions Systems (CVS)
    - 1986-1990: open source
    - Multiple files → entire project
    - Concurrent files → multi-user repositories –
    - Can store repo on a remote server → more than one user can work on multiple files
  - Apache Subversion (SVN)
    - 2000: open source→ Faster
    - Allow for non text files
    - Track → files/directories as a whole (collectively) → instead of version 3, file is in revision 3.
    - Stayed the most popular version of version control until Git

Some Historical Perspective (3/3)

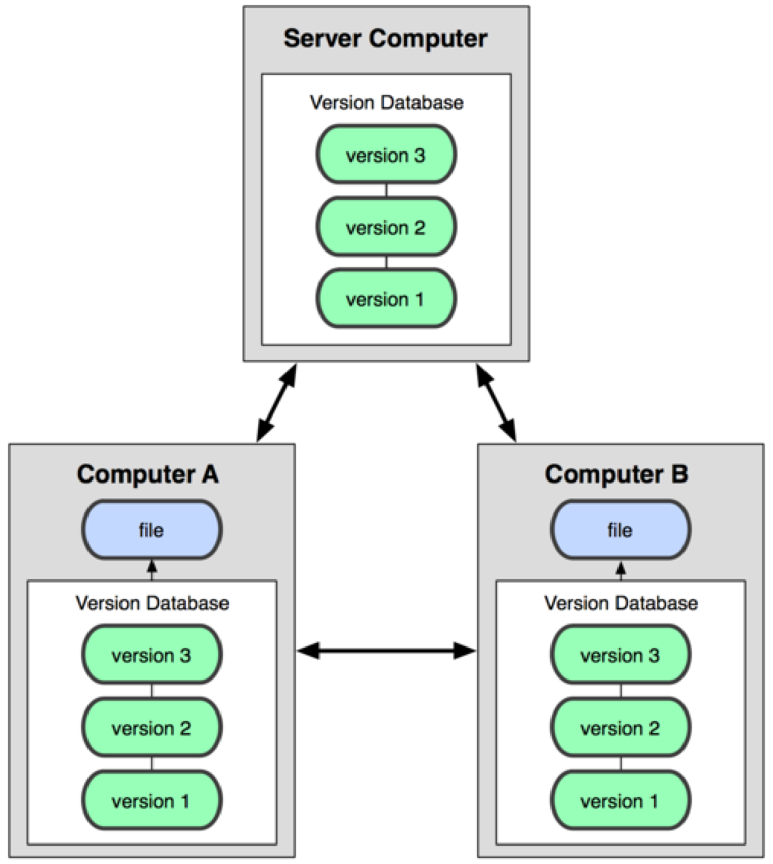
- Milestone → <u>Distributed version control</u>
  - <u>BitKeeper SCM</u>
    - 2000-2005: (closed source)
    - Create “community” version (free) → adopted in Linux Kernel (free) → but it is proprietary software!
  - <u>Git is Born!</u>
    - April 2005 by Linus Torvalds (Linux creator)
    - Open source and free!
    - Compatible with Linux, macOS, Windows
    - Faster than other SCMs (100X) and Better Safeguards

Git – Explosion of popularity

- GitHub Launched in 2008 to host Git repositories
- Revision Control Meets Social Media! → function like a webpage
- Some Stats
  - 2009: over 50,000 repos, over 100,000 users
  - 2011: over 2 million repos, 1 million users
  - 2018: bought by Microsoft
  - 2019: over 57 million repos, over 28 million users

What is Distributed Version Control?

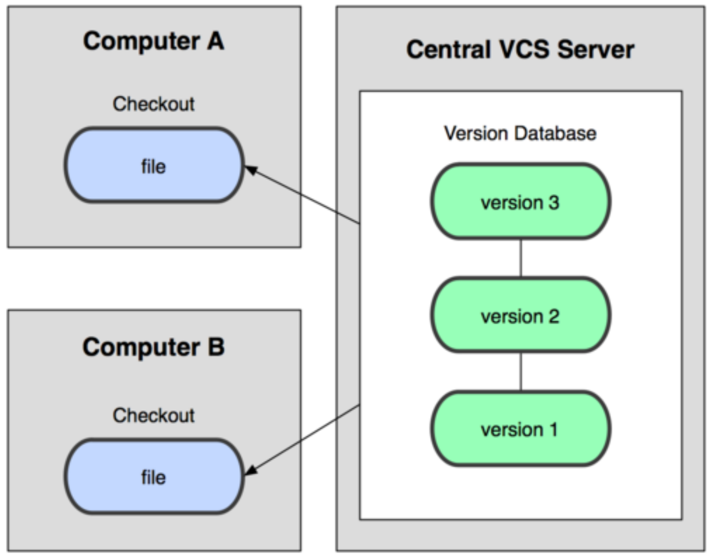
- Main idea
  - Different users maintain their own repository
  - No central repository
  - Track changes and not versions → track change sets
  - When ready → merge change sets or “apply patches”
- Benefits
  - No need to communicate with a central server
  - No network access required
  - No single failure point
  - Encourages collaboration → “forking of projects”
  - Developers can work independently
  - Change sets can be accepted/rejected → “pull request”



TEXAS A&M UNIVERSITY
Harold Vance Department of Petroleum Engineering

Centralized X Distributed Models

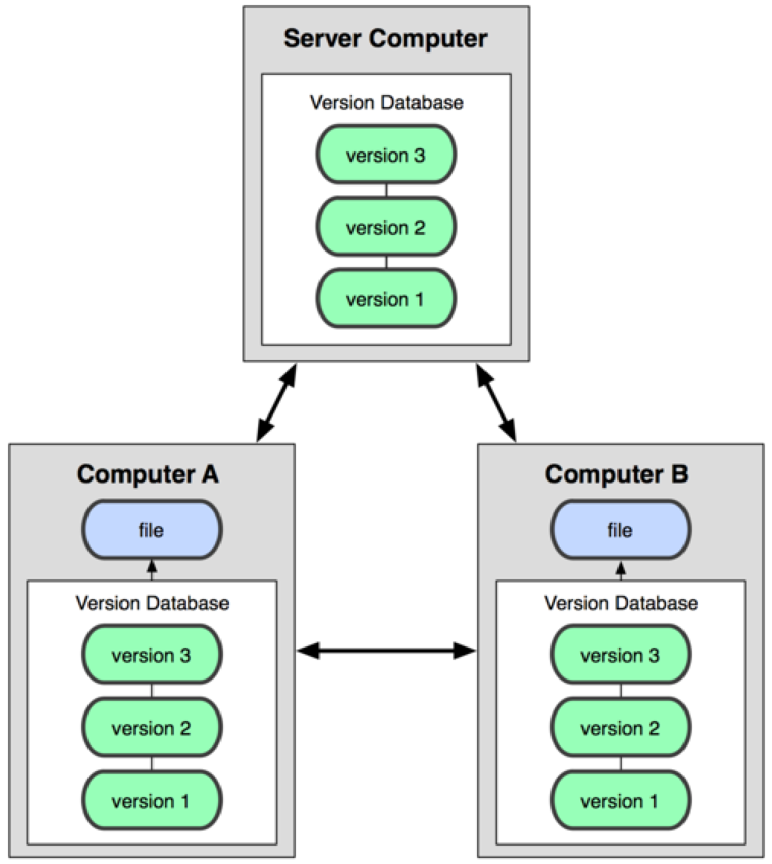
Centralized Model



(CVS, Subversion, Perforce)

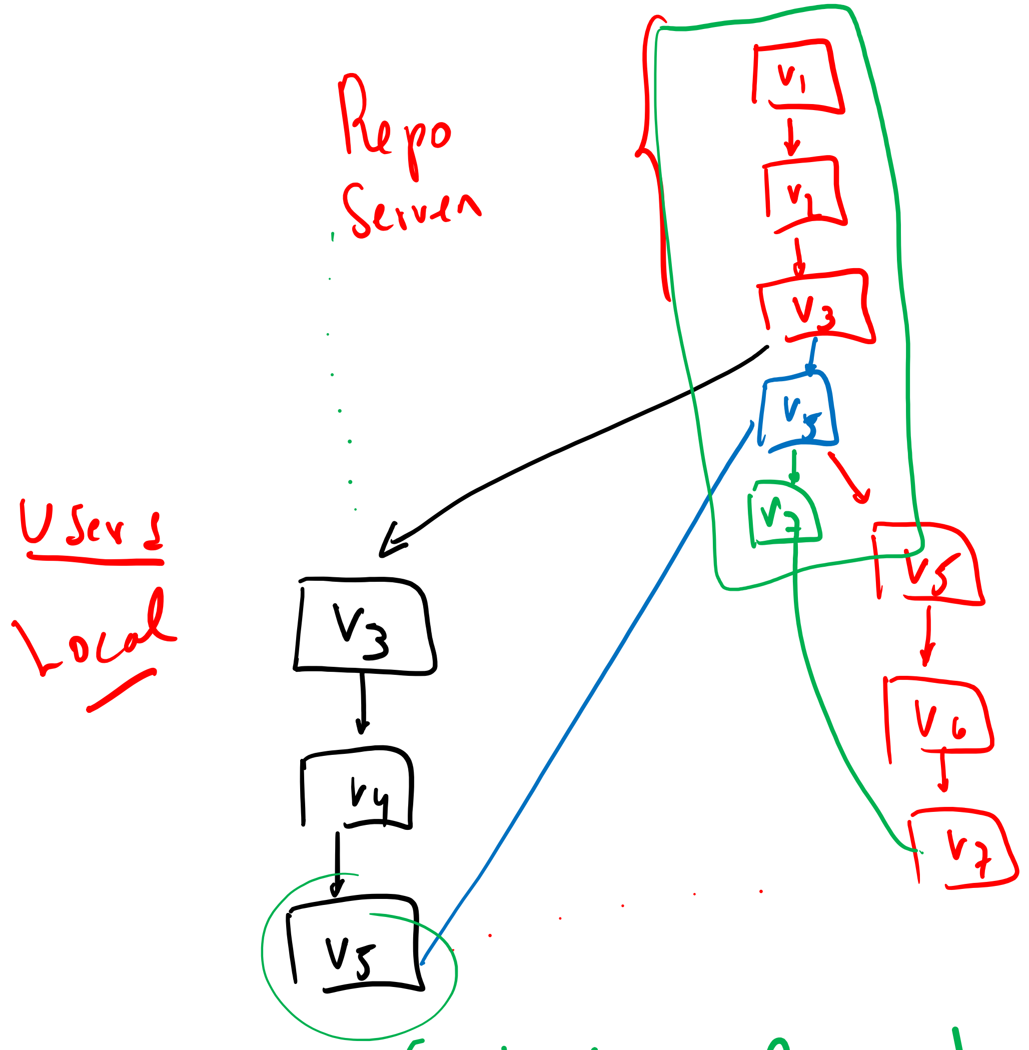
- No central repository
- Different users maintain their own repo
- Changes are stored as change sets → track changes and not versions
- “merge” in change sets
- Many working copies

Distributed Model

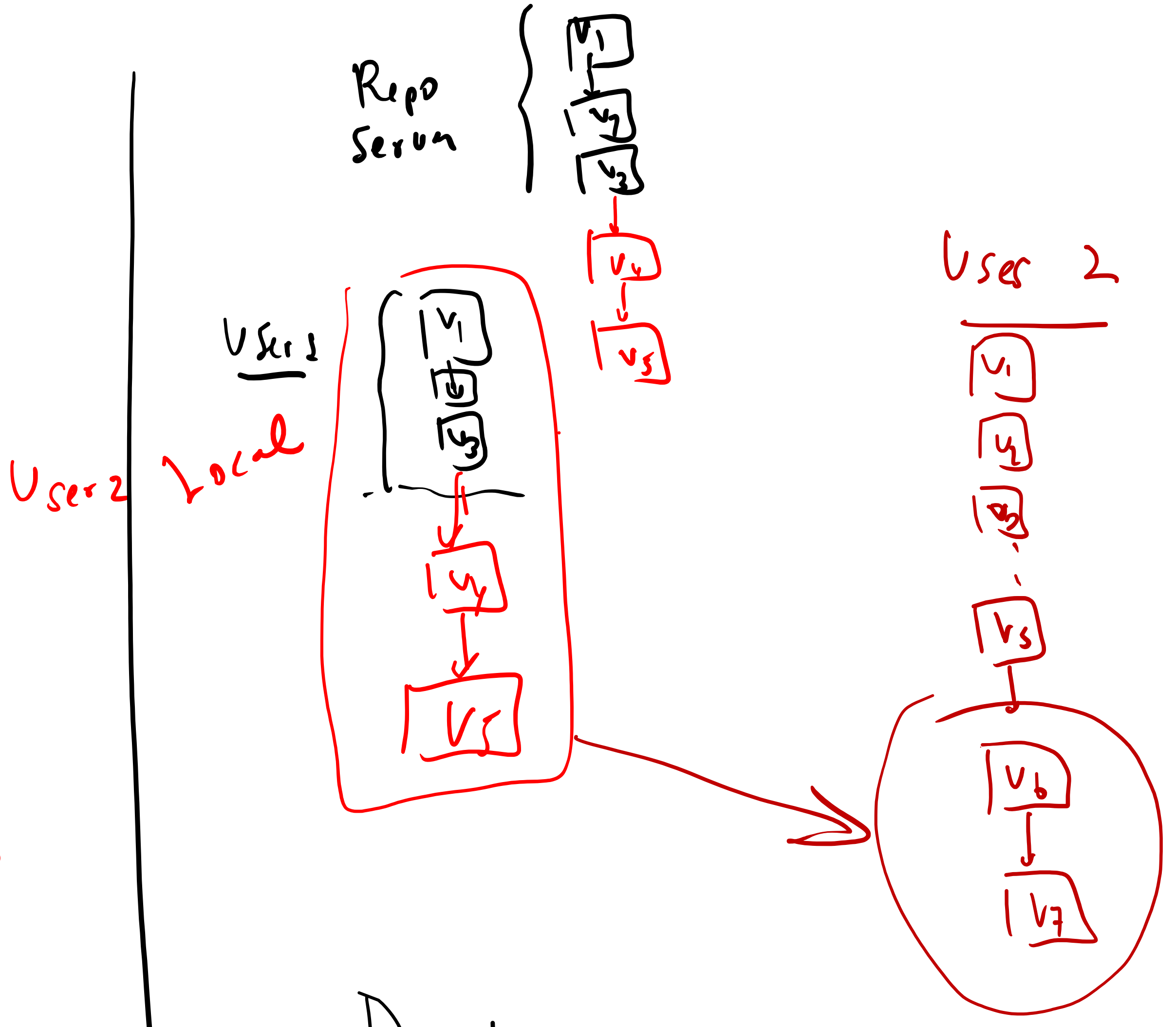


(Git, Mercurial)

Pics by Ruth Anderson, images from http://git-scm.com/book/en/

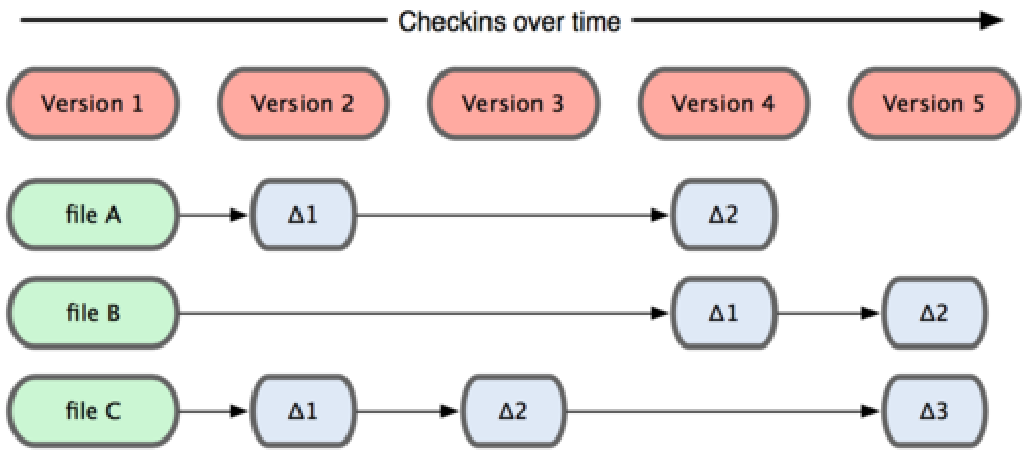


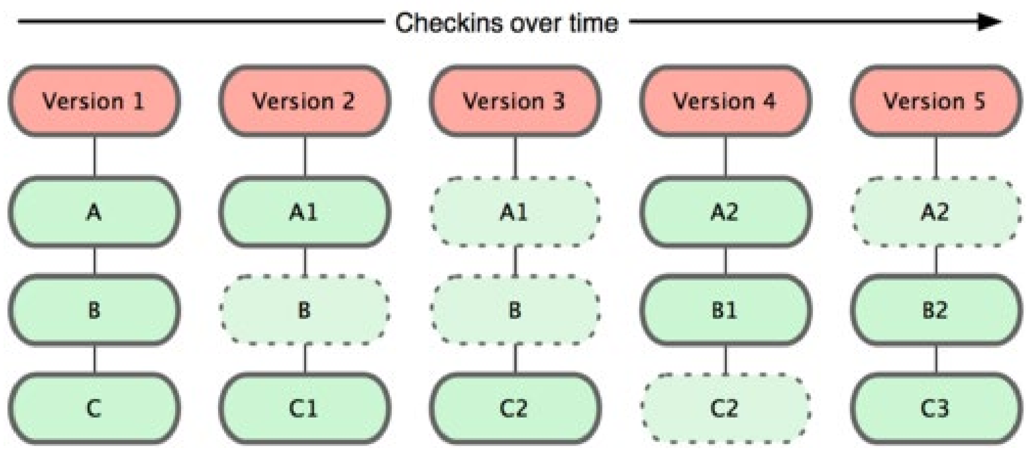
Centralized Repository VCS



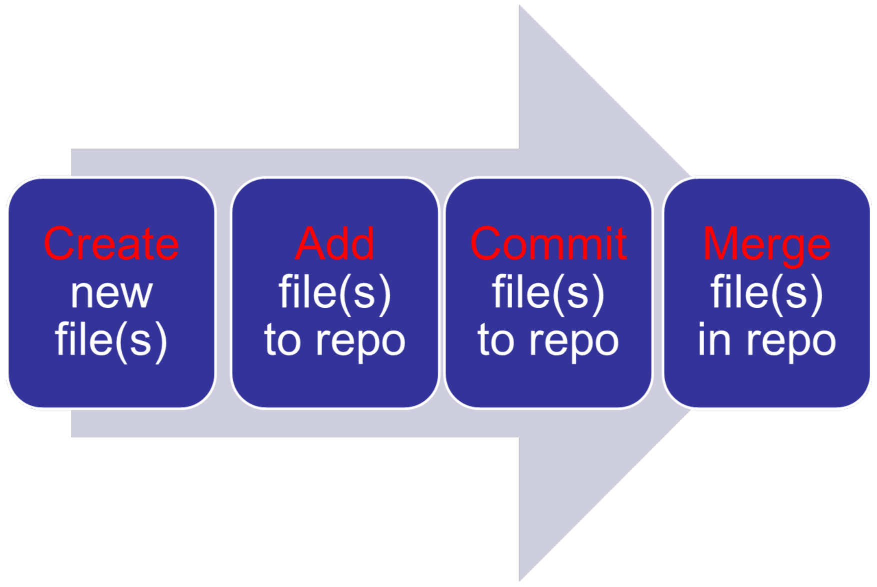
Distributed VCS

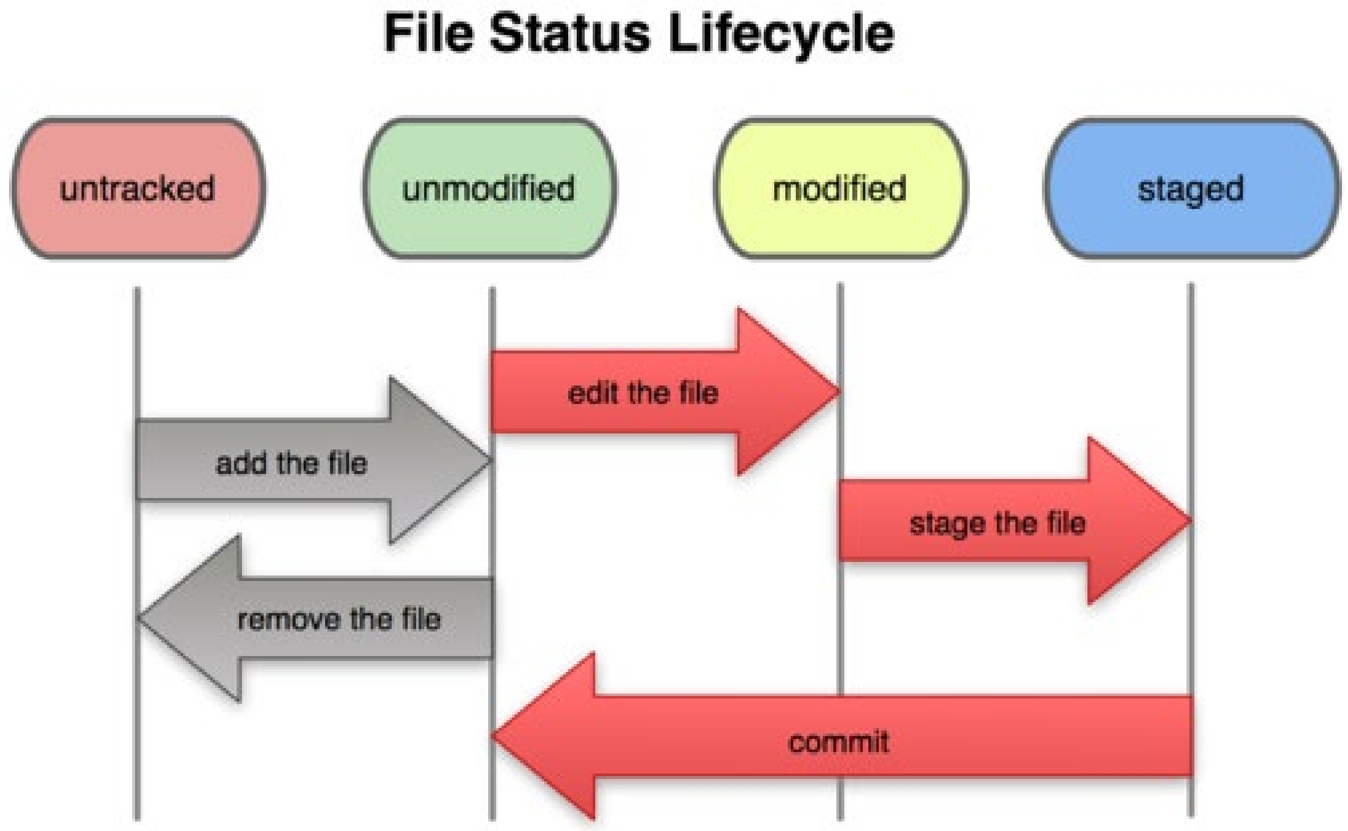
Git Takes Snapshots



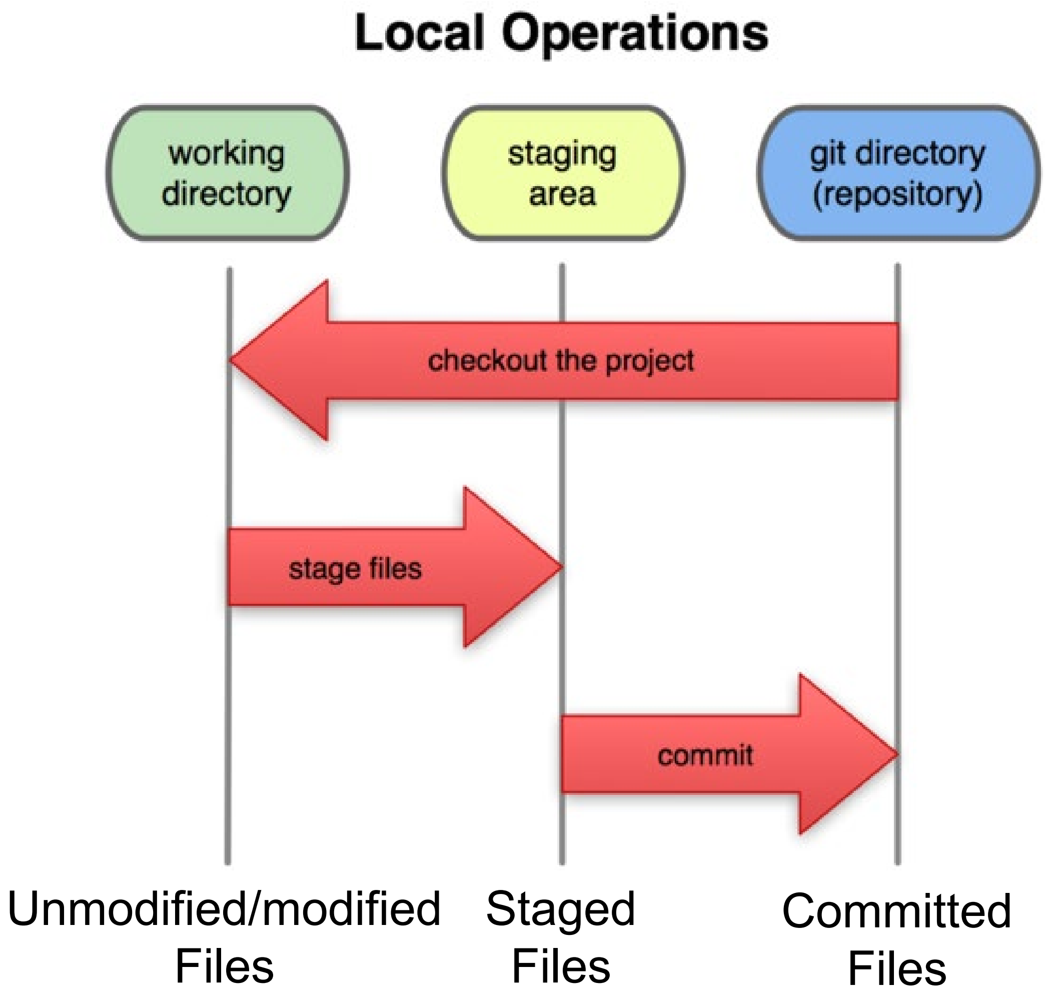


Git file lifecycle



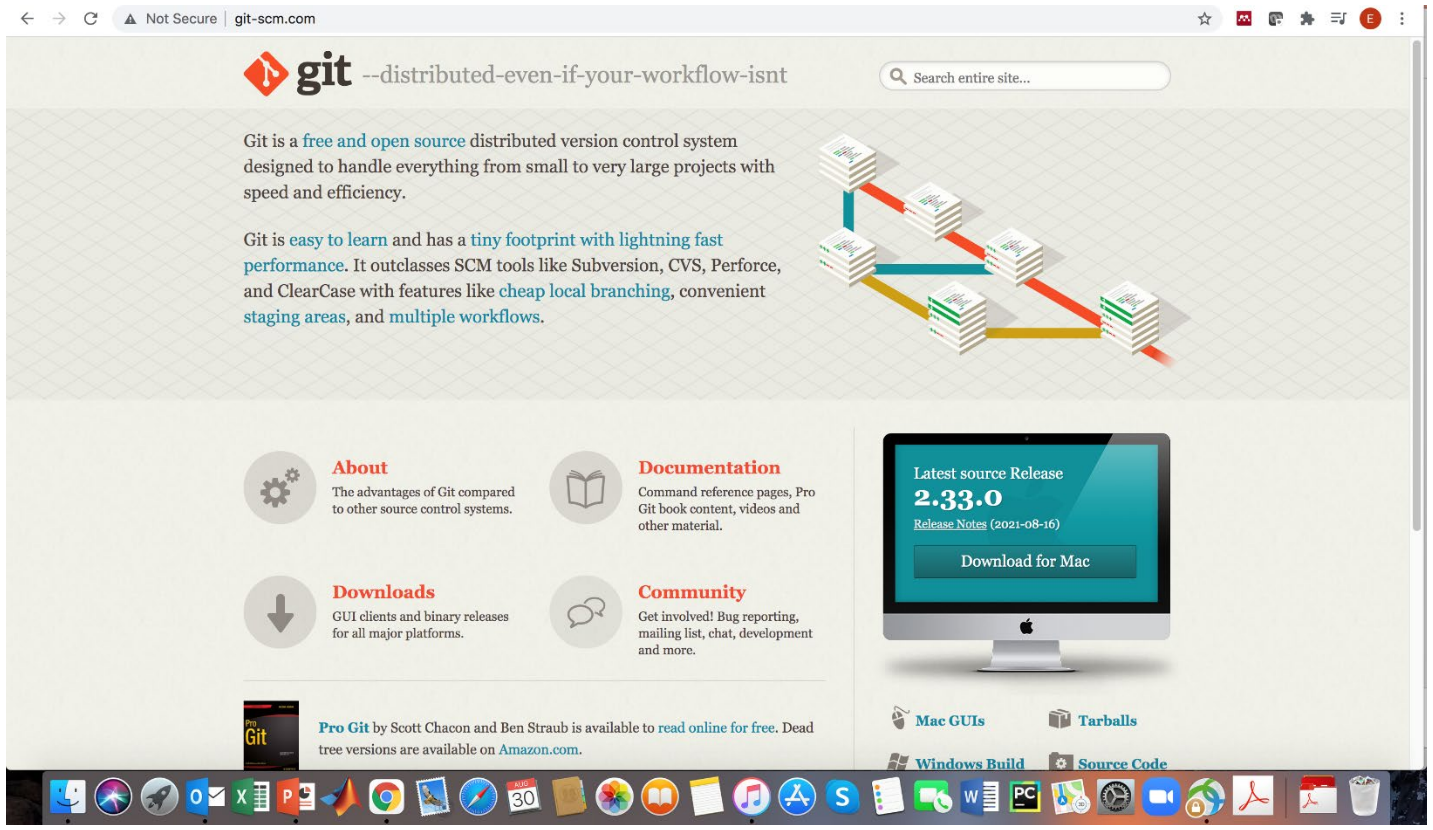


A Local Git project has three areas



Download/Install Git

- Mac users → if new mac probably you have a version of Git
- Mac/Windows → go to http://git-scm.com/
- It will recognize your computer (mac/windows) → download


Not Secure | git-scm.com

git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.33.0
Release Notes (2021-08-16)
Download for Mac

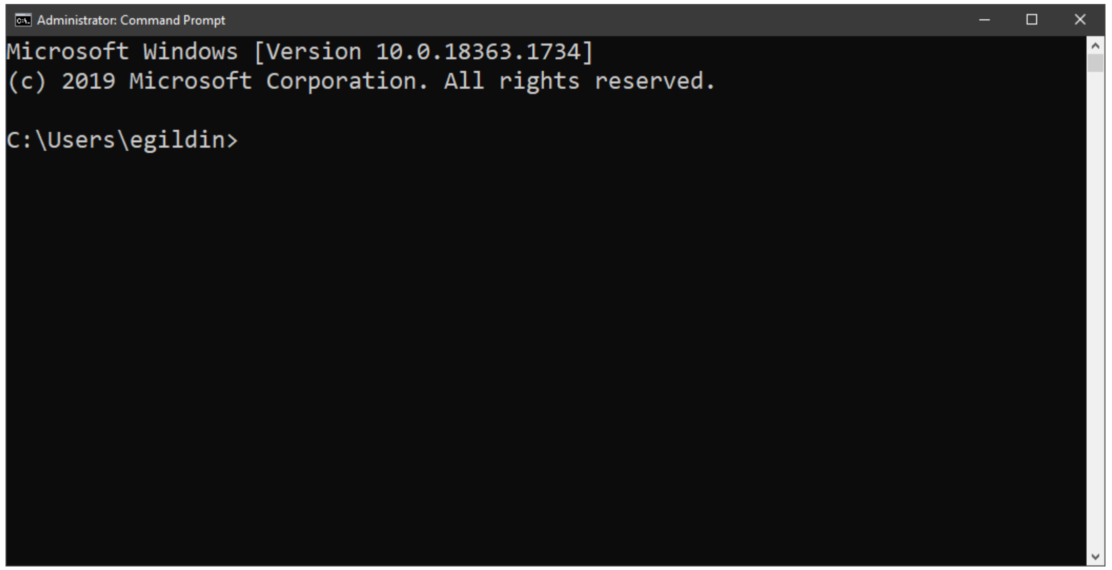
Mac GUIs Tarballs
Windows Build Source Code

Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Great step-by-step → https://www.stanleyulili.com/git/how-to-install-git-bash-on-windows/

UINIX TERMINAL

- Historically created with a Bash (Bourne Again Shell) environment
  - A shell is a terminal application used to interface with an operating system through written commands.
  - Set of command line utility programs that are designed to execute on a Unix style command-line environment.
- Command prompt or terminal
  - Windows → `cmd`
  - Mac → `terminal`
  - Especially for Git → `Git bash`



```
Microsoft Windows [Version 10.0.18363.1734]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\egildin>
```

UNIX/LINUX Commands

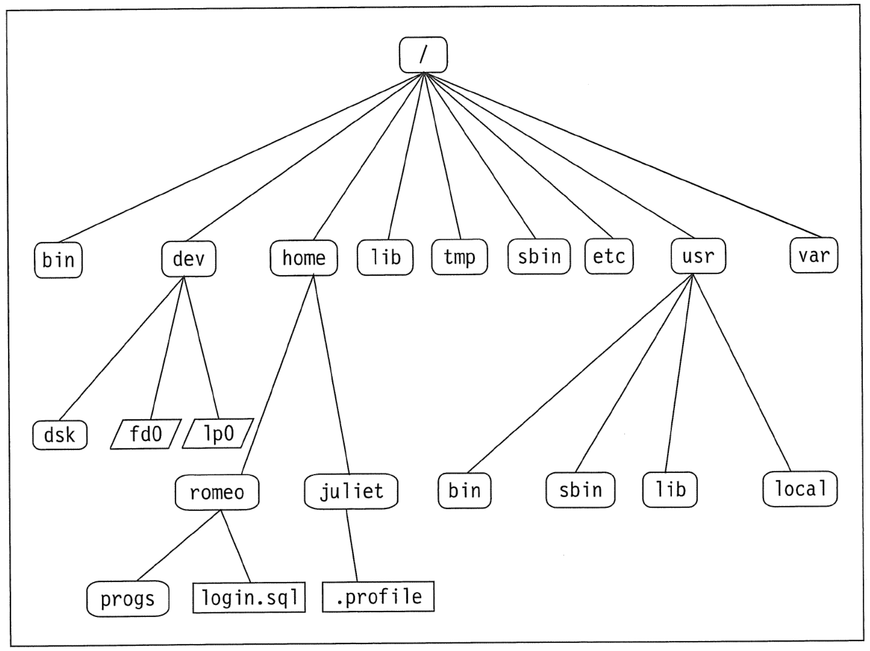
- List all files and directories
  - Simple list → `ls`
  - All hidden → `ls -a`
  - All details → `ls -l`
  - All the Files in the Descending Order of their Size → `ls -ls`
  - Combine some → `ls -la`
- Quick edit with VI
  - `vi file_name.txt`
  - Switch to command mode → `Esc`
  - Save and exit → `:wq` or `ZZ`
- Add a file
  - `touch file_name.txt`

→ many nicer word documents/code developments exist (notepad, emacs, atom)

UNIX/LINUX Ideas

- UNIX → everything is seem as a file system (even printers, hard drives, etc)
- External drives, for instance, mounted as device file
- Basic Commands*
  - Print→ `pwd`
  - Change → `cd`
  - Create new folder → `mkdir`
  - Remove folder → `rm`
  - Copy files → `cp`
  - Move Files → `mv`
  - Clear screen → `clear`
  - Concatenate → `cat filename`
  - Create empty file (for Mac) → `touch filename`
  - Create empty file (for windows) → `copy con filename`

(reads data from the file and gives their content as output)

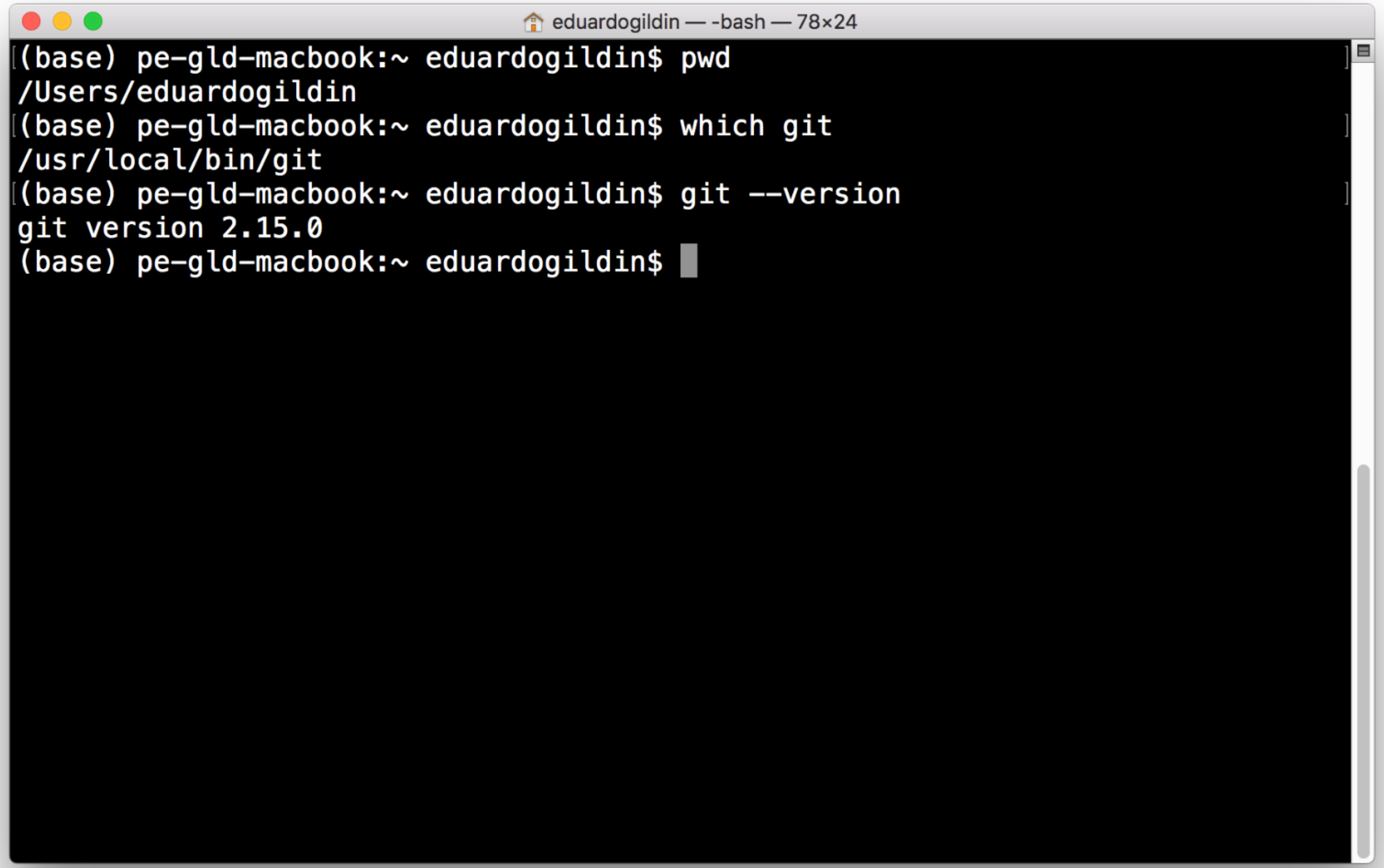


Source: S. Das. *Your UNIX/Linux: The Ultimate Guide.* Third. McGraw-Hill, Inc.,

*Linux Command Cheat Sheet → https://www.guru99.com/linux-commands-cheat-sheet.html

Few Commands – test if properly installed

- Print working directory
  - pwd
- Path for git
  - which git
- Git version
  - git --version



eduardogildin — -bash — 78×24

```
(base) pe-gld-macbook:~ eduardogildin$ pwd
/Users/eduardogildin
(base) pe-gld-macbook:~ eduardogildin$ which git
/usr/local/bin/git
(base) pe-gld-macbook:~ eduardogildin$ git --version
git version 2.15.0
(base) pe-gld-macbook:~ eduardogildin$ 
```

Some configurations

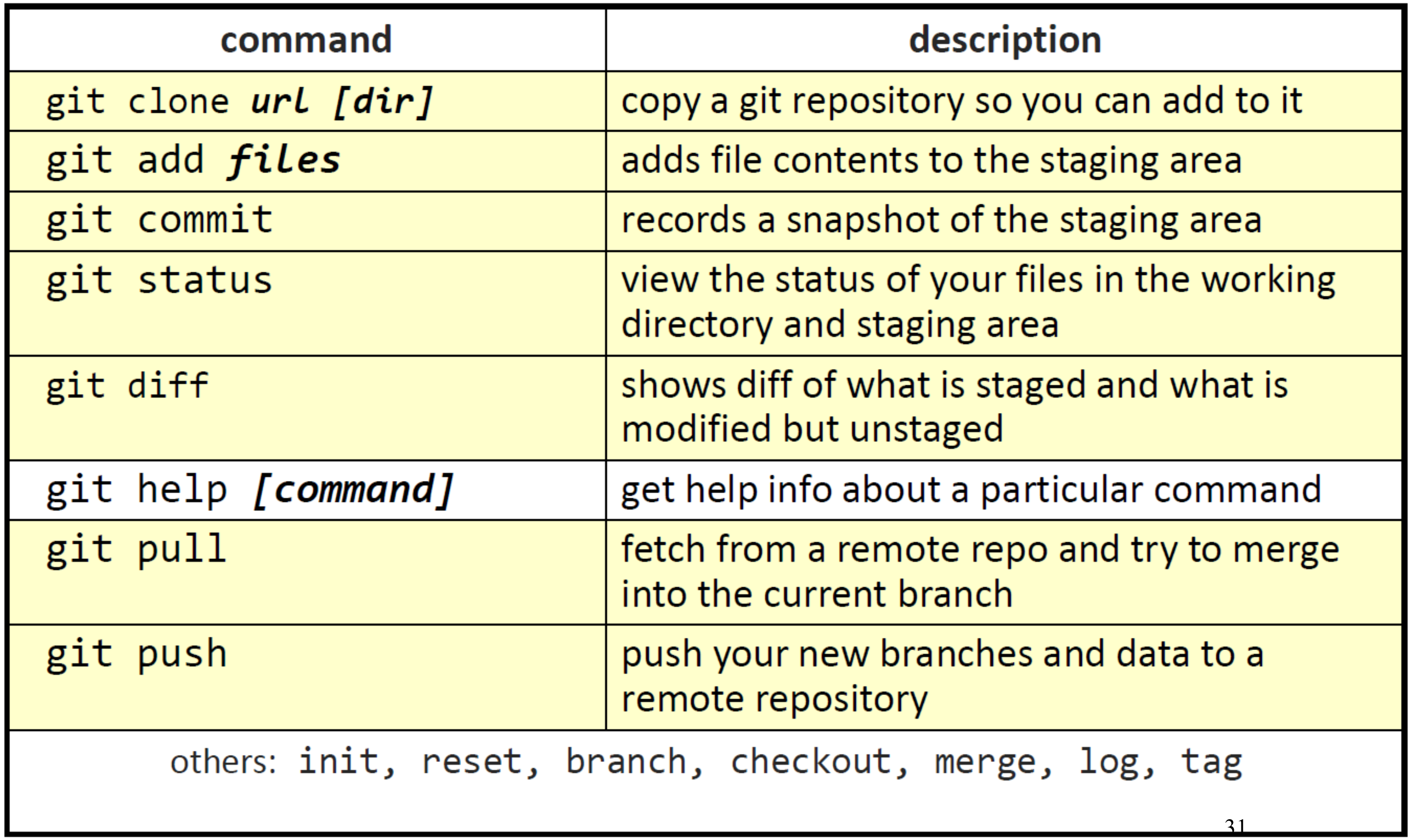
- `git config --global user.name “Eduardo_Gildin”`
- `git config --global user.email egildin@tamu.edu`
- `git config –list`
- `git config user.name`
- `cd ~`
- `ls –la`
- `cat .gitconfig`
- `git config --global core.editor “notepad.exe”`

Some configurations – pick an editor

- `git config --global core.editor “notepad.exe”`
- Or
- `git config --global core.editor “atom --wait”`
- `git config --global color.ui “true”`
- `cat .gitconfig`
- `cat .git/config`

- Git help

Git Commands



command	description
git clone ***url [dir]***	copy a git repository so you can add to it
git add ***files***	adds file contents to the staging area
git commit	records a snapshot of the staging area
git status	view the status of your files in the working directory and staging area
git diff	shows diff of what is staged and what is modified but unstaged
git help ***[command]***	get help info about a particular command
git pull	fetch from a remote repo and try to merge into the current branch
git push	push your new branches and data to a remote repository

others: init, reset, branch, checkout, merge, log, tag

Initialize a project (Git)

- Create a new folder (this will be your project or git repo)
- Navigate to this folder on the command line
- Folder is empty, but if you look closer, there is .git file there → this is where git will have all commands to track the repository
- See folder (list files) → `ls -la .git`
- See insider .git → `ls -la .git`

Let's try a simple example for Git Workflow

1. Create a new folder (your REPO) and go there →
   `pwd` → `mkdir FirstProject` → `ls (or ls -la)` →
   `cd FirstProject` (can use TAB for out complete)
2. Create a new file → `touch file1.txt`
3. Edit file → `vi file1.txt`
4. Check what is inside → `cat file1.txt`
5. `git status` → you may see untracked files
6. `git add filename (or git add.)`→ moving to the staging tree
7. `Git status` → what happened?

Git does not know anything about it. Any changes will not be tracked!

Working directory

Let’s try a simple example for Git Workflow

6. … `Git status` → what happened?
7. `git commit --m “Succinct text for commit msg”`
8. `git log` or `git log –n 3` (or any number) → limits the number of logs
9. `git status`
10. `Let’s repeat the process and create more files →`

<u>Create 2 new files</u>
file1.txt
file2.txt
file3.txt

⟹ `Git status` ⟹ What happened?

Now, what happens if you edit the files

1. Git status → anything to commit?
2. Edit `file1.txt` → any modification
3. `git status ?` → changes not staged for commit → git recognizes this is a tracked file and it is in the repo. Also, it recognizes the version is different.
4. Need to add to staging? → `git add file1.txt` → now you are adding the changes!
5. Before we commit, make changes to file2, and file3 → git status?
6. `git add file2.txt` → `git status`
7. `git commit –m “made changes to first and second files”`
8. `git status` → `git log` → what do you see?

Now, what happens if you edit the files?

1. Add third file (`git add file3.txt`) and commit
2. `git commit -m “modified all third file”`
3. Edit file1.txt → any modification
4. Git status? → Git log? → all three of the tree agree now!

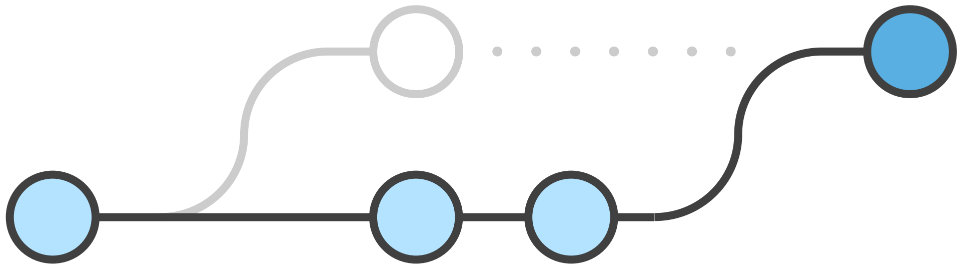
Compare files –diff

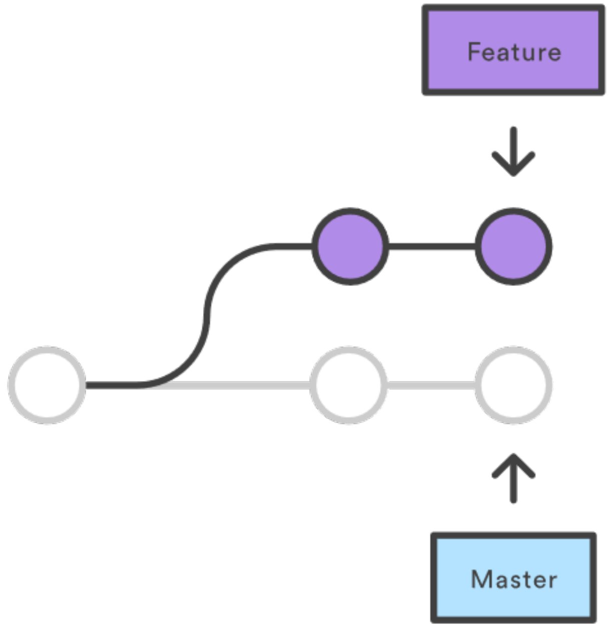
1. Modify file1.txt for instance (add 2,3 lines) → save
2. Git status? → there is a change? What was the change? → if someone open the new file1.txt, there is no info what has changed.
3. Need to ask git what has exactly changed
4. use `diff` command (UNIX) → `git diff`
5. Compare file a to file b → a is at repo; b at local directory
6. + → added to the file (green)
7. Minus → subtracted from file, plus → added to the file
8. File really long → shows only snapshots of text that has changed.

Now we know what the changes are and get decide to commit or not with the new version → see more on "pull requests in GitHub"

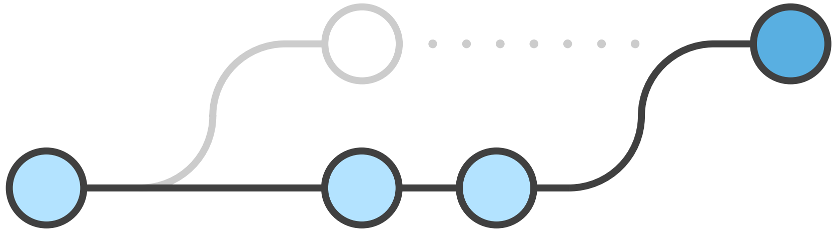
Branching (1/3)

- Master branch is for production
- Major Bugs fixing or feature developments should go into other branches





Branching (2/3)



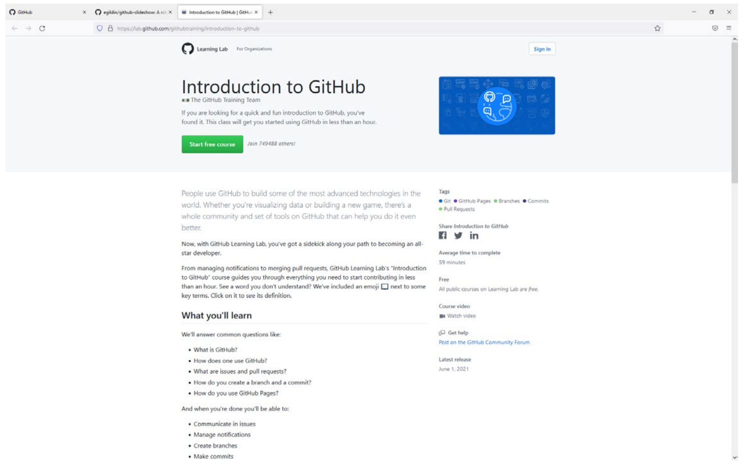
- check file → see what is in there
  - `cat filename`
- Create a new branch
  - `git checkout -b branch_name` (e.g., myfixes or new_fixes)
- List all branches (* is the current branch)
  - `git branch`
- Make changes in the file and make a commit
  - `vi filename` → change something
  - `commit -am "Message"` (recall `-a to` bypass add/stage; –m for message)
- NOTE: this commit happened into the branch and not in the master branch
  - You can see this by → `git log`

Branching (3/3)

- Now, let's return to the master branch
  - `Git checkout master`
- If you look at the file, it was the previous version
  - `cat filename`
- To get these changes to master use merge
  - `git merge master new_fixes`
- If you look AGAIN at the file → new version
  - `cat filename`
  - `git log`
- Once you are done with the fixes, you can delete the branch!
  - `git branch -D branchname` → `git branch`

Revision Control Meets Social Media

- GitHub → Cloud based remote repository server for Git
  - Many open source projects use it, such as the Linux kernel.
  - You can get free space for open source projects or you can pay for private projects.
- Question: Do I have to use github to use Git?
  - Answer: No!


GitHub
egildin/github-slideshow: A rol
Introduction to GitHub | GitHu
https://lab.github.com/githubtraining/introduction-to-github
Learning Lab
For Organizations
Sign in
Start free course

Introduction to GitHub

The GitHub Training Team

If you are looking for a quick and fun introduction to GitHub, you've found it. This class will get you started using GitHub in less than an hour.

Join 749488 others!

People use GitHub to build some of the most advanced technologies in the world. Whether you're visualizing data or building a new game, there's a whole community and set of tools on GitHub that can help you do it even better.

Now, with GitHub Learning Lab, you've got a sidekick along your path to becoming an all-star developer.

From managing notifications to merging pull requests, GitHub Learning Lab's "Introduction to GitHub" course guides you through everything you need to start contributing in less than an hour. See a word you don't understand? We've included an emoji 📖 next to some key terms. Click on it to see its definition.

What you'll learn

We'll answer common questions like:

- What is GitHub?
- How does one use GitHub?
- What are issues and pull requests?
- How do you create a branch and a commit?
- How do you use GitHub Pages?

And when you're done you'll be able to:

- Communicate in issues
- Manage notifications
- Create branches
- Make commits

Tags

Git, GitHub Pages, Branches, Commits, Pull Requests

Share *Introduction to GitHub*

Average time to complete

59 minutes

Free

All public courses on Learning Lab are *free*.

Course video

Watch video

Get help

Post on the GitHub Community Forum

Latest release

June 1, 2021

TEXAS A&M UNIVERSITY
Harold Vance Department of Petroleum Engineering

But, GitHub allows you to

- Getting a deeper look at the changes/commits and project management
- Ability to integrate collaboration seamlessly into your Git repository
  - Look at Issues
  - Pull requests
  - Visualize Projects
  - Multiple Organizations
  - Multiple Teams

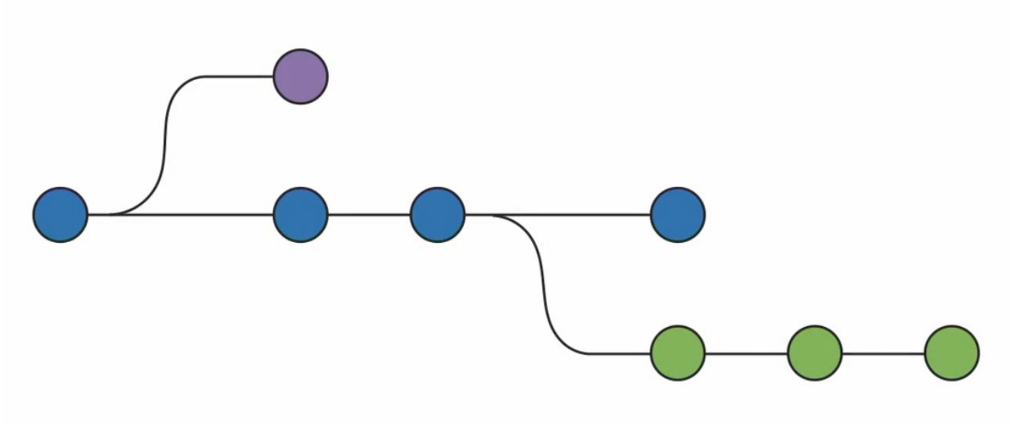
$\}$ Been able to discuss and collaborate changes made to your repo with other developers, contributors and other team members

GitHub Pages – Like Webpage

- Let’s start with a few examples (from people I know)
  - https://github.com/collections/github-pages-examples
  - https://github.tamu.edu/RDCRG
  - https://opensourcedrilling.org/
  - https://seismicreservoirmodeling.github.io/SeReM/
  - https://tonycsw2905.github.io/
  - https://github.com/pymor

GitHub Workflow

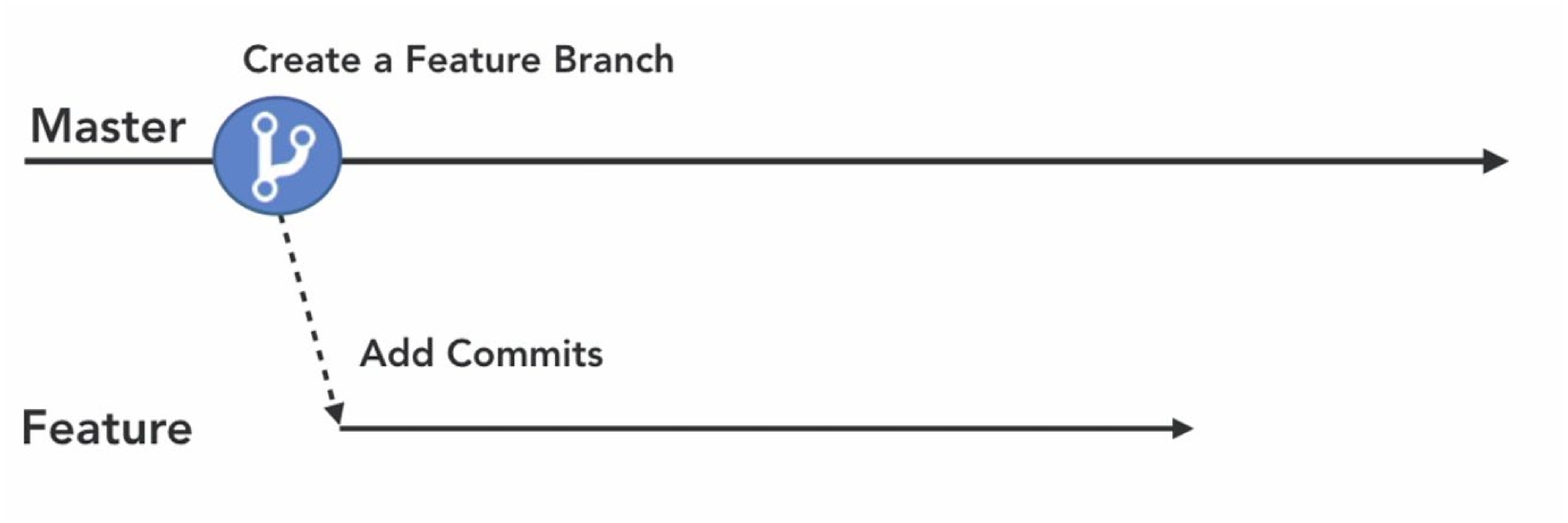
- Recall, in Git all of our work in organized into branches
  - You can see this as a timeline how the project has changed over time
  - There is one branch that is automatically created → master
  - Contains all the records associated with the changes in the project → ultimately is the one taken for production



Production
Master

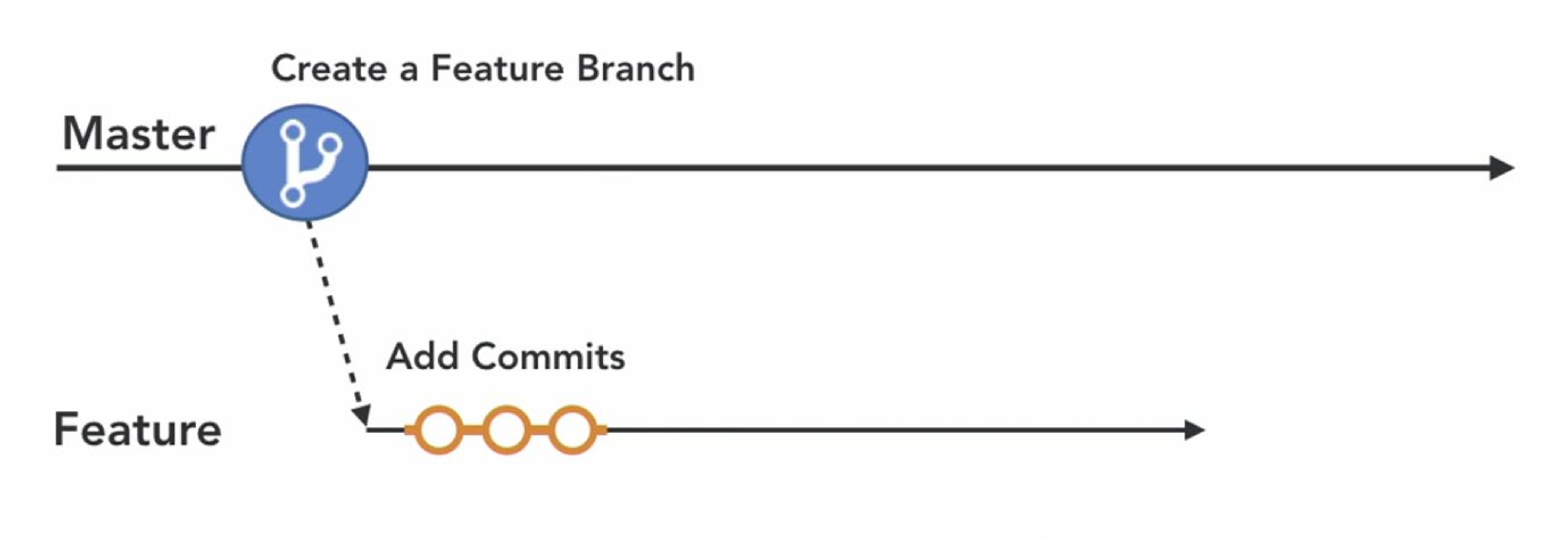
Source: Aaron Stewart (GitHub Essentials)

GitHub Workflow (1/5) - Branch



Source: Aaron Stewart (GitHub Essentials)

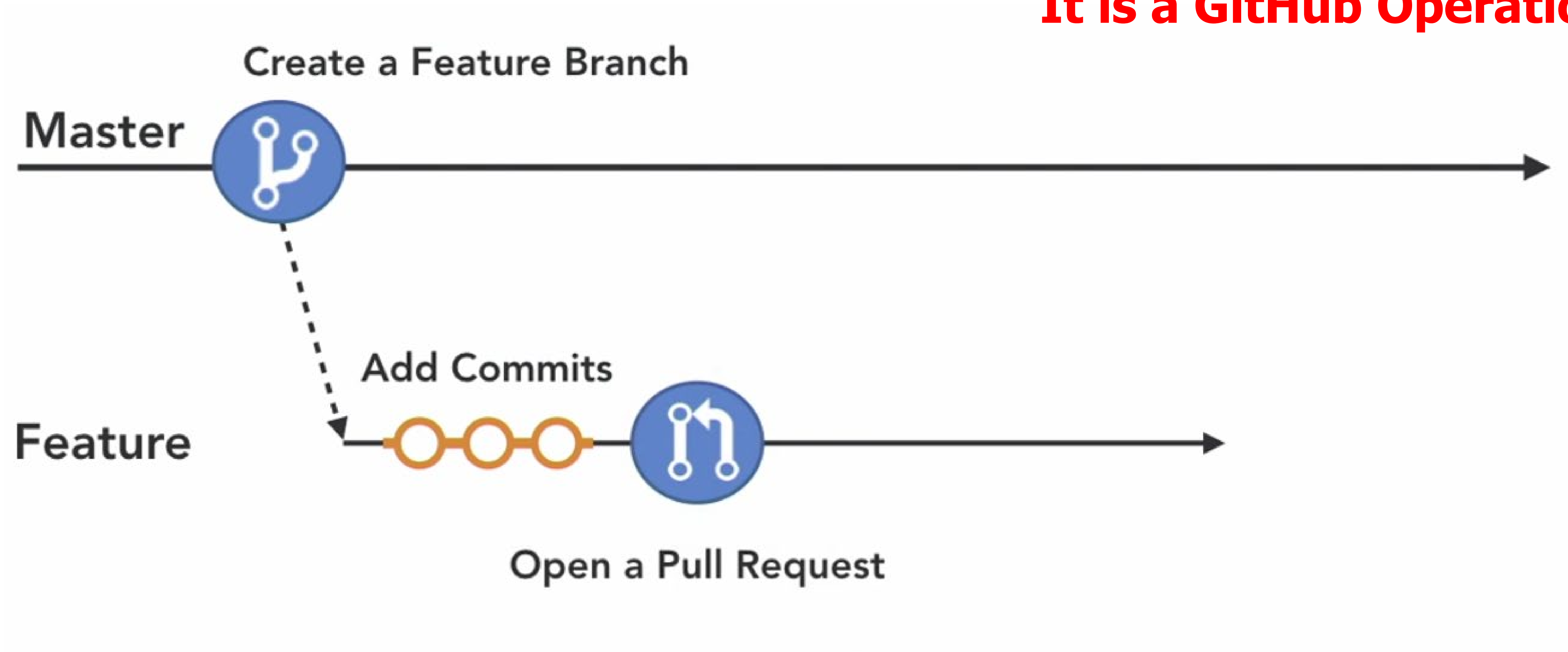
GitHub Workflow (2/5) – make commits at branch



Source: Aaron Stewart (GitHub Essentials)

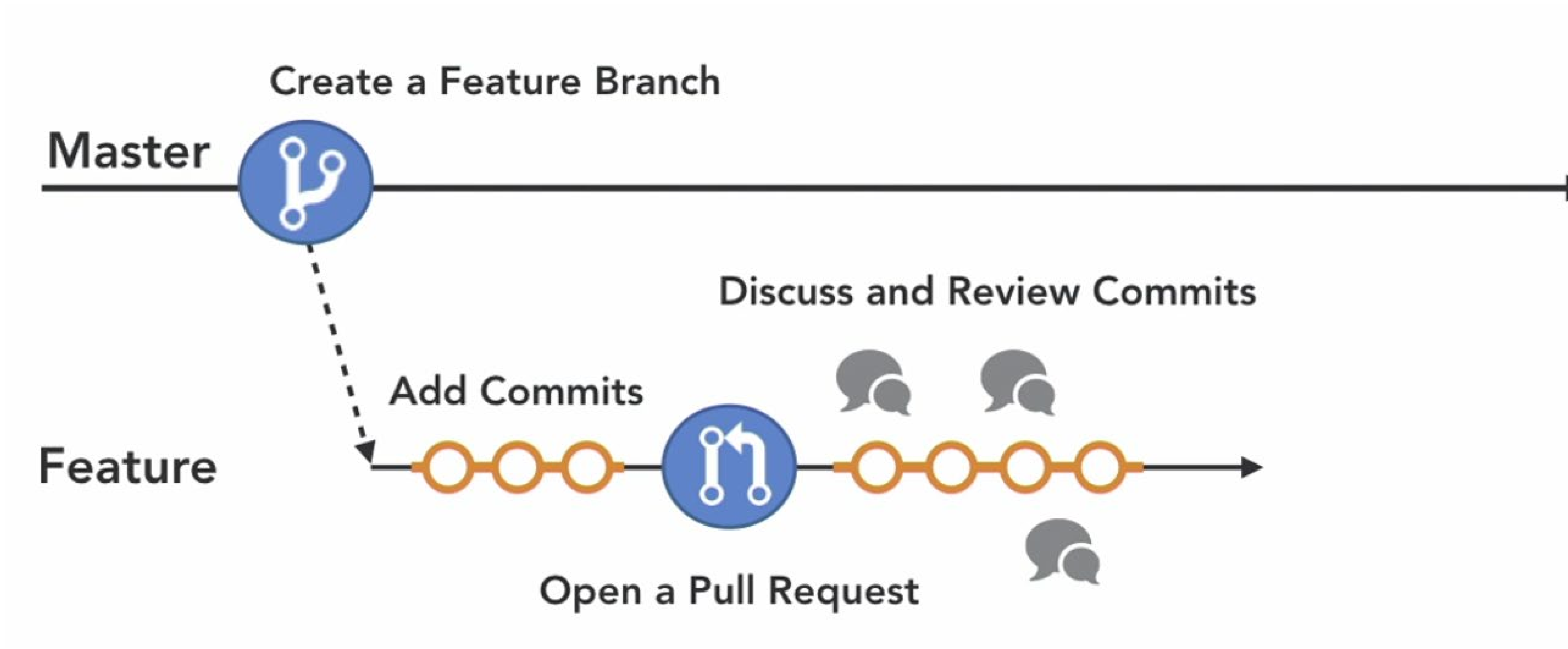
GitHub Workflow (3/5) – Open Pull Request

It is a GitHub Operation

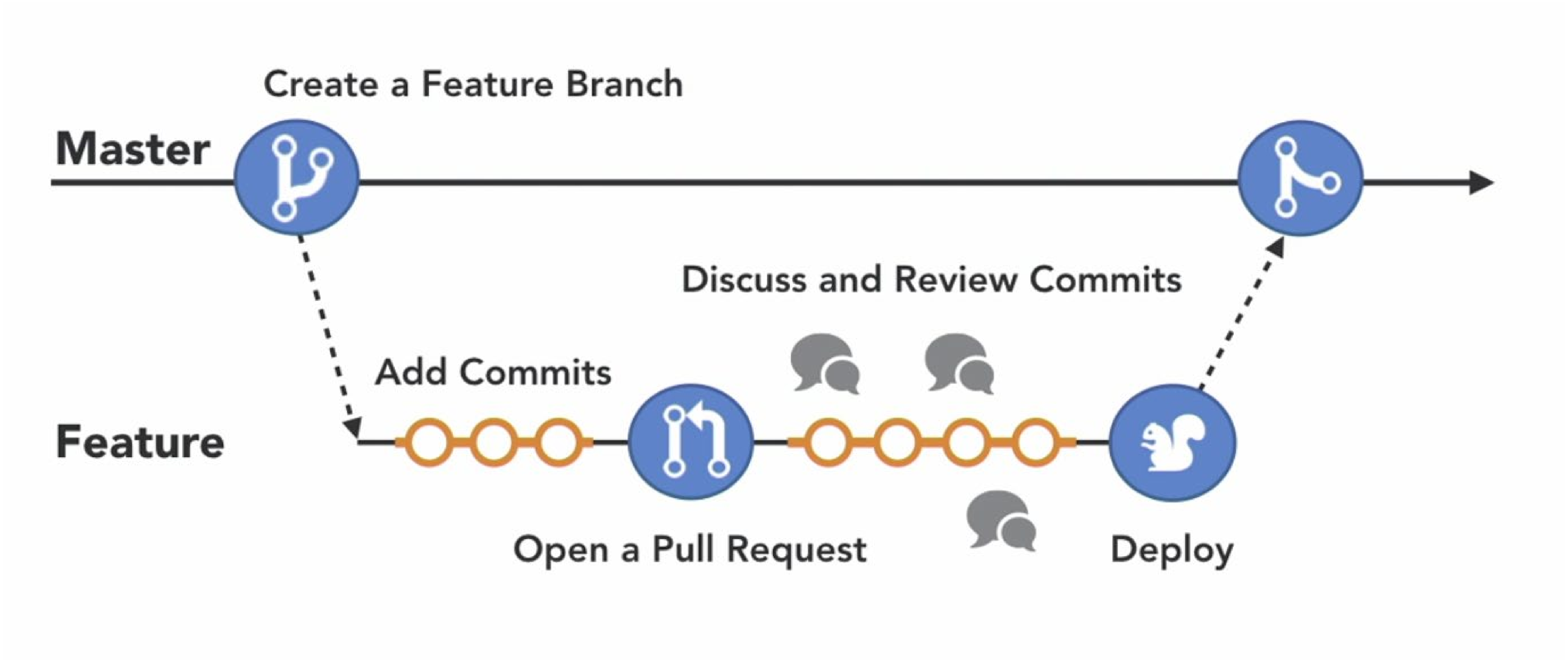


Source: Aaron Stewart (GitHub Essentials)

GitHub Workflow (4/5) – Discussion and review of changes



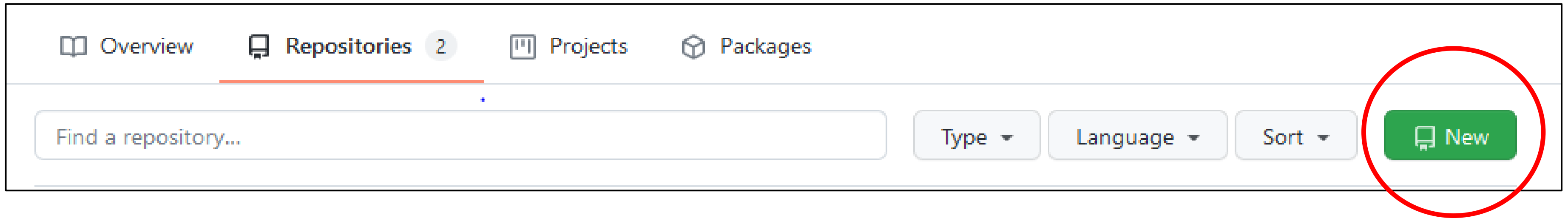
GitHub Workflow (5/5) – Deploy and Merge



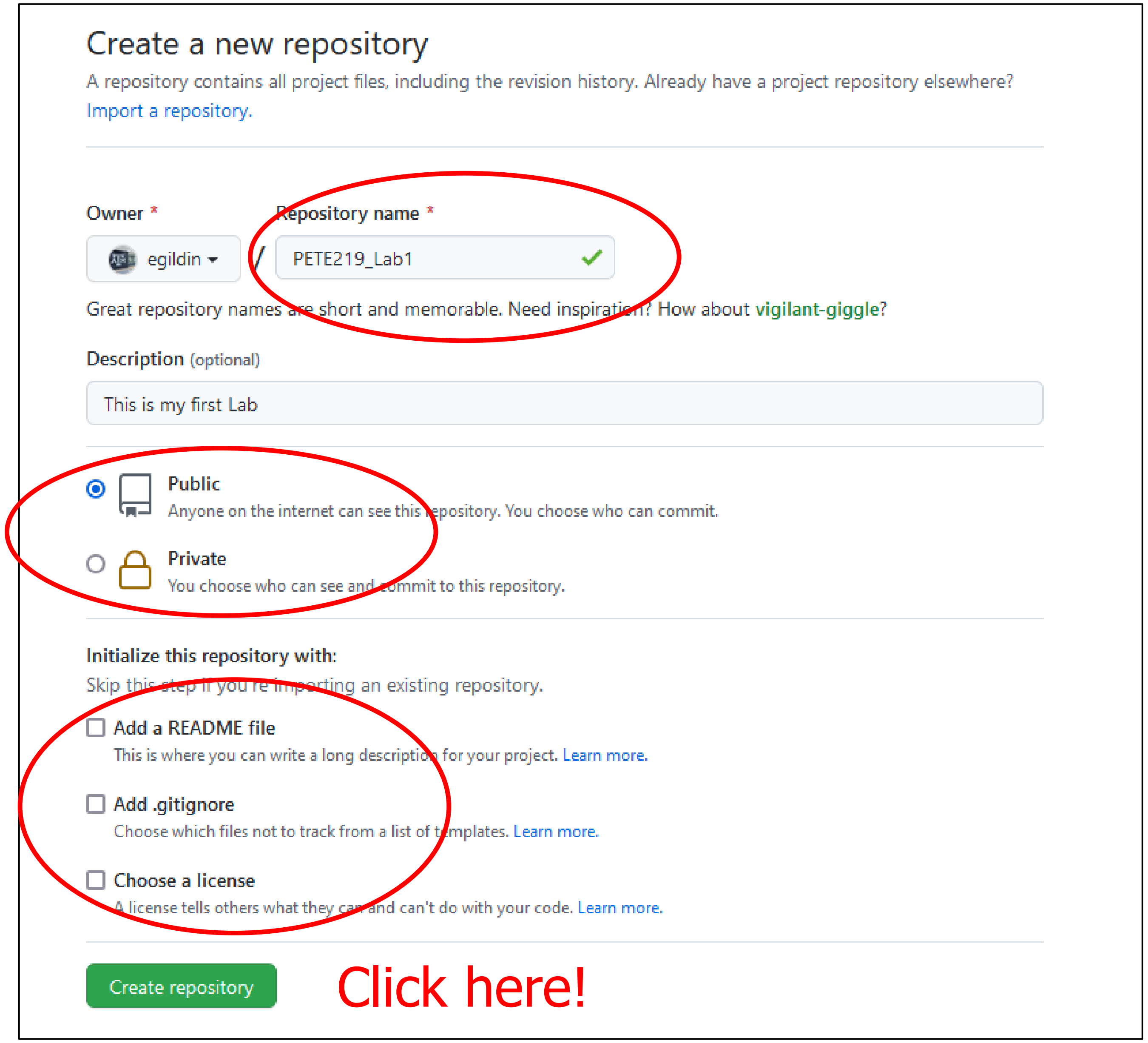
Source: Aaron Stewart (GitHub Essentials)

Create GitHub repository

- Since git works under the hood in GitHub, pretty much all the steps we have taken so far can be replicated in GitHub. You can also download the desktop version (desktop.github.com)
- Let's start with a brand new repo
- Once logged in → create a new repository



  - Add information → the repository name is like the name of the folder you want to become a repo → this is like `git init` in that folder → example: PETE219_Lab1



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * egildin / Repository name * PETE219_Lab1 ✓

Great repository names are short and memorable. Need inspiration? How about vigilant-giggle?

Description (optional)

This is my first Lab

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

☐ Add .gitignore
Choose which files not to track from a list of templates. Learn more.

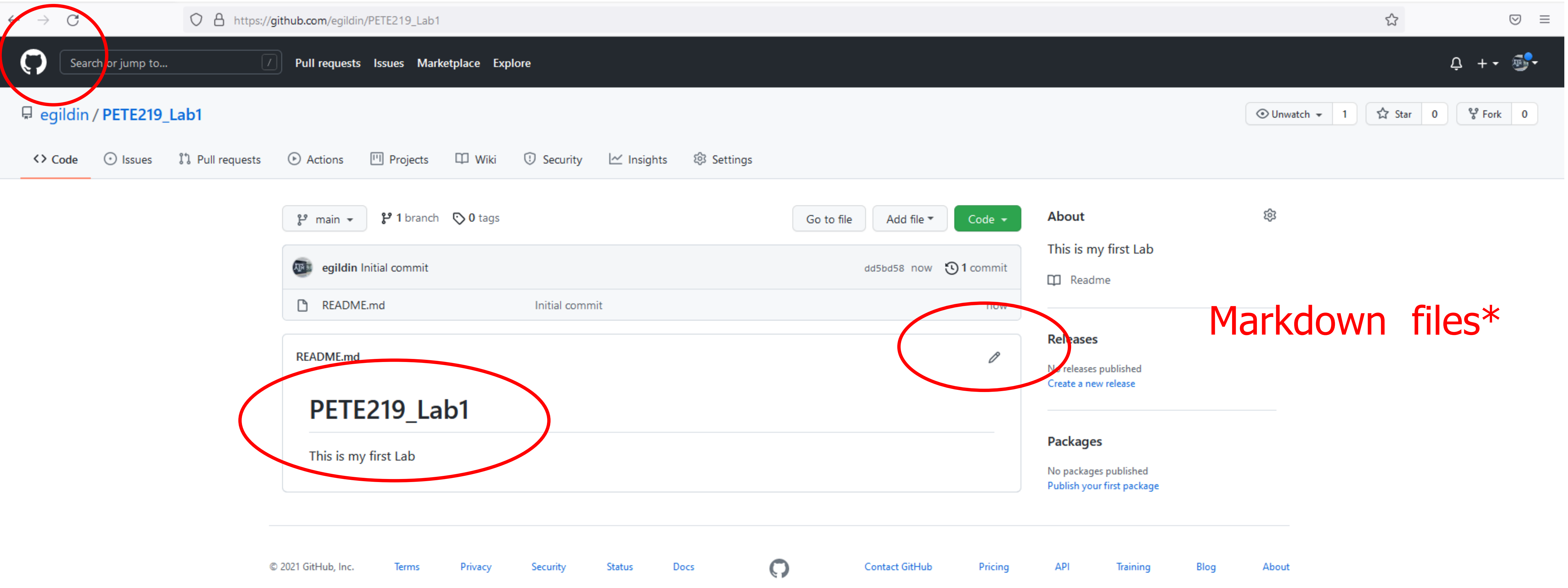
☐ Choose a license
A license tells others what they can and can't do with your code. Learn more.

Project (folder) name

Public or Private?

What is Readme file?
What is .gitignore?
Licenses?

Some more functionality → edit README.md



https://github.com/egildin/PETE219_Lab1

egildin / PETE219_Lab1

README.md

PETE219_Lab1

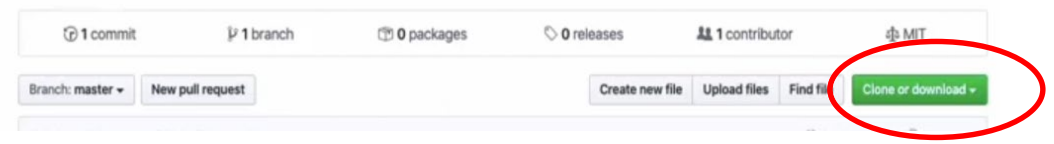
This is my first Lab

Markdown files*

* https://daringfireball.net/projects/markdown/

Fork and Clone Repo

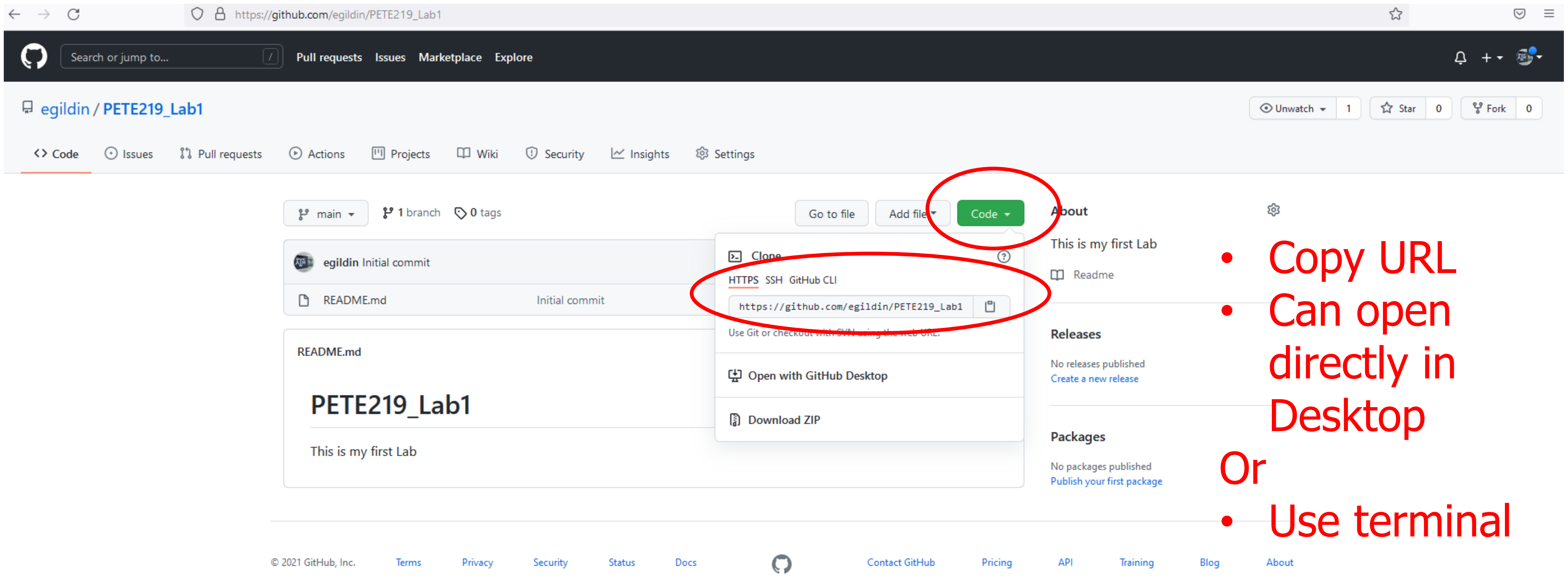
- To Clone the repo → get local copy
  - Go to clone or download button



  - Copy url to clipboard
  - In terminal → `git clone URL` (paste)
- Files shows up to your folder
- Edit files

TEXAS A&M UNIVERSITY
Harold Vance Department of Petroleum Engineering

Clone Repo → save to a local folder


https://github.com/egildin/PETE219_Lab1

egildin / PETE219_Lab1

Clone

HTTPS SSH GitHub CLI

```
https://github.com/egildin/PETE219_Lab1
```

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

README.md

PETE219_Lab1

This is my first Lab

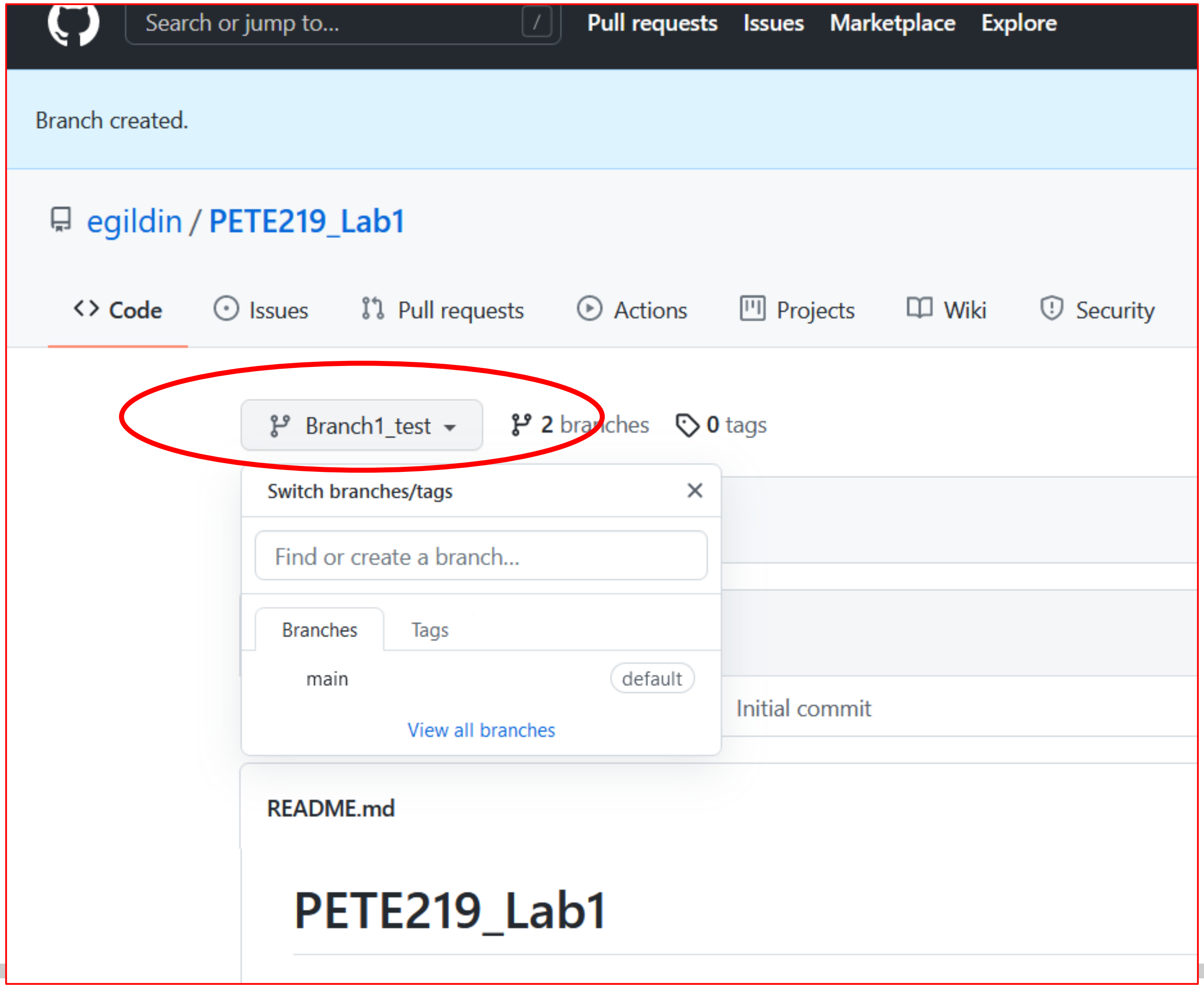
- Copy URL
- Can open directly in Desktop

Or

- Use terminal

Git clone

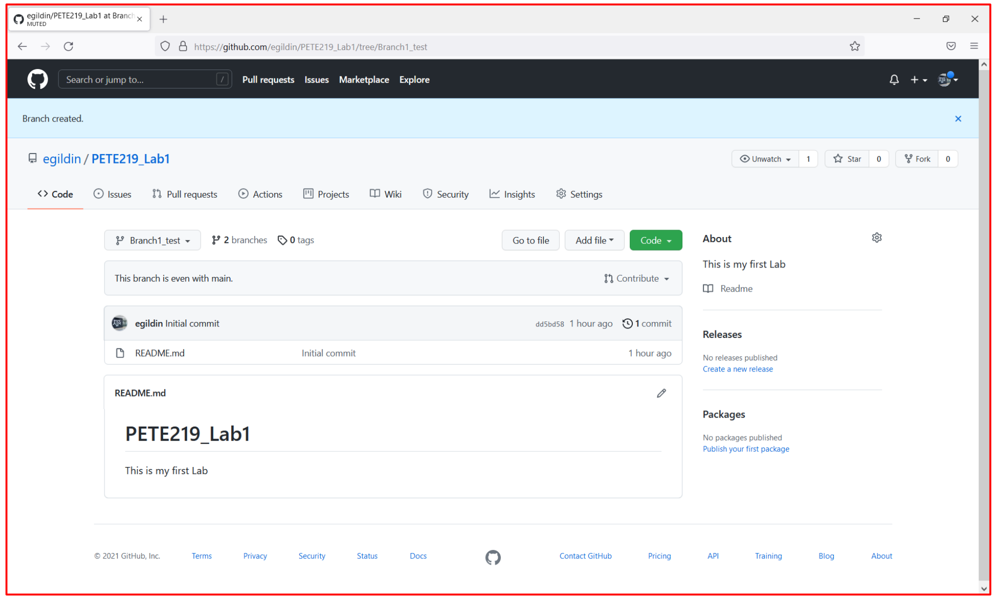
Branches



Branch created.

egildin / PETE219_Lab1

PETE219_Lab1



egildin/PETE219_Lab1 at Branch

https://github.com/egildin/PETE219_Lab1/tree/Branch1_test

Branch created.

egildin / PETE219_Lab1

This branch is even with main.

egildin Initial commit dd5bd58 1 hour ago 1 commit

README.md Initial commit 1 hour ago

README.md

PETE219_Lab1

This is my first Lab

About

This is my first Lab

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

© 2021 GitHub, Inc.

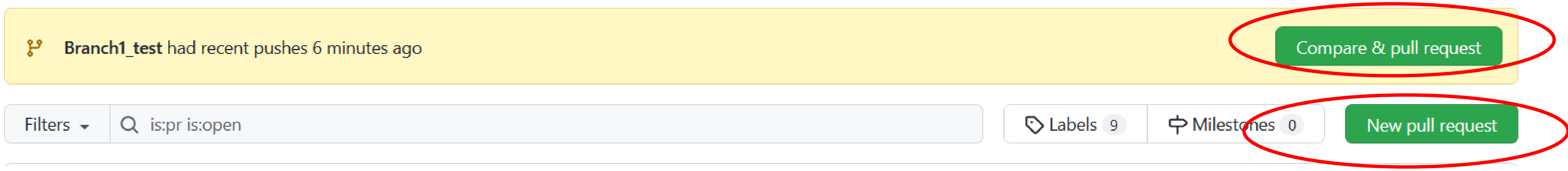
- Created an exact replica of my repo

Make a Commit

- Switch to new Branch
- Make some changes, to say, the README file → commit to the change
  - what git command is this?
- View history
  - what git command is this?
- Switch back to the master branch → what do you see?
- View history
  - what git command is this?

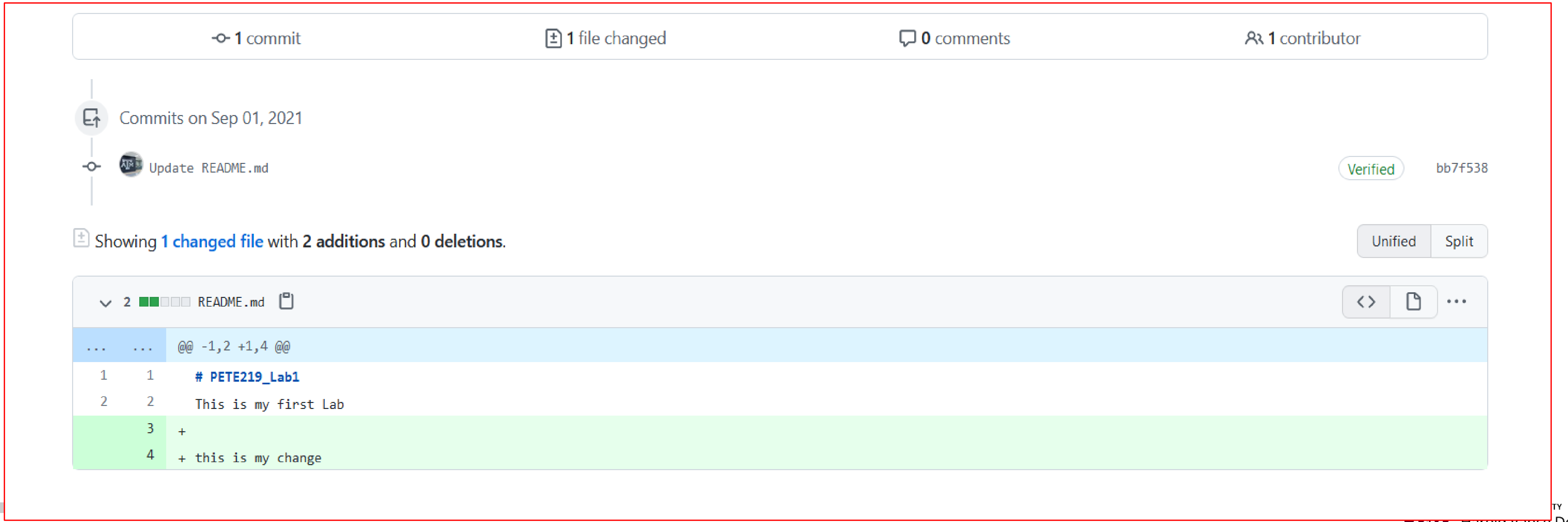
Open and Merge Pull Requests

- Notifies developers about the changes you have pushed to a branch in a repository
- Allow people to acknowledge and review changes before merging into the main file
- Go to Pull Request → Click on Open New Pull Request



TEXAS A&M UNIVERSITY
Harold Vance Department of Petroleum Engineering

- Start with Compare & Pull request →
  - Scroll down to the bottom of the page
  - What do you see? Which git command is this?



1 commit | 1 file changed | 0 comments | 1 contributor

Commits on Sep 01, 2021

Update README.md Verified bb7f538

Showing **1 changed file** with **2 additions** and **0 deletions**.

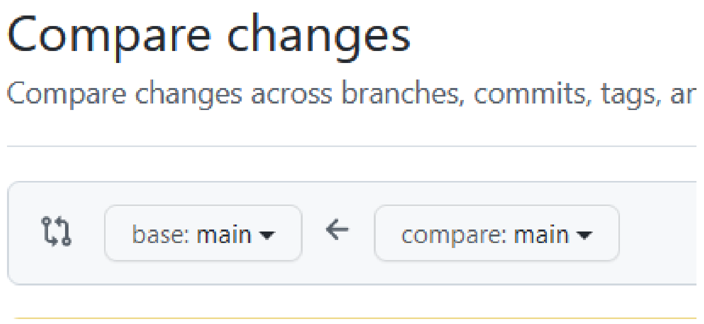
Unified Split

2 README.md

```
@@ -1,2 +1,4 @@
  # PETE219_Lab1
  This is my first Lab
+
+ this is my change
```

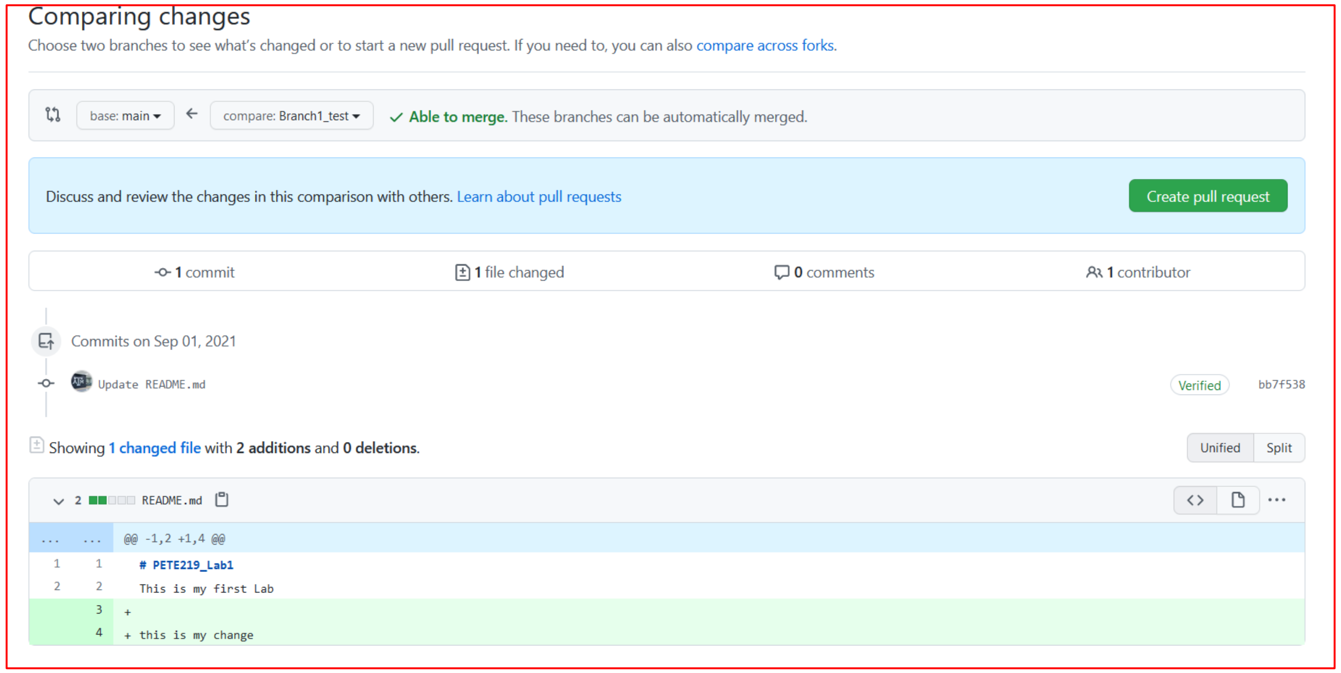
Harold Vance Department of
Petroleum Engineering

- Let’s use Create a New Pull Request
  - Select Master and Branches
  - Select the Branch you have created
  - Create the Pull Request
  - Now you can MERGE
  - Confirm
  - Can Delete the Branch
  - Which git commands have we used?



Compare changes

Compare changes across branches, commits, tags, ar



Comparing changes

Choose two branches to see what’s changed or to start a new pull request. If you need to, you can also compare across forks.

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. Learn about pull requests

1 commit · 1 file changed · 0 comments · 1 contributor

Commits on Sep 01, 2021

Update README.md — Verified — bb7f538

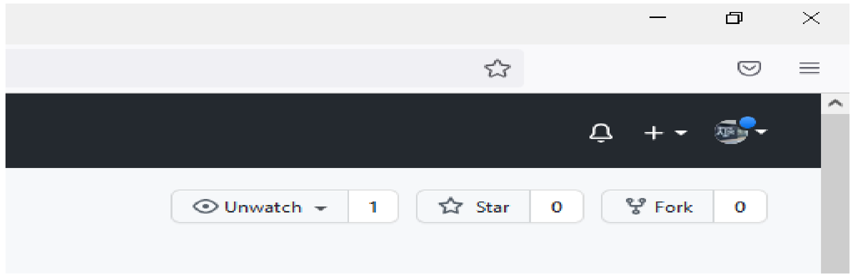
Showing 1 changed file with 2 additions and 0 deletions.

```
2 README.md
@@ -1,2 +1,4 @@
 # PETE219_Lab1
 This is my first Lab
+
+ this is my change
```

TEXAS A&M UNIVERSITY
Harold Vance Department of Petroleum Engineering

Starting from someone's else Github

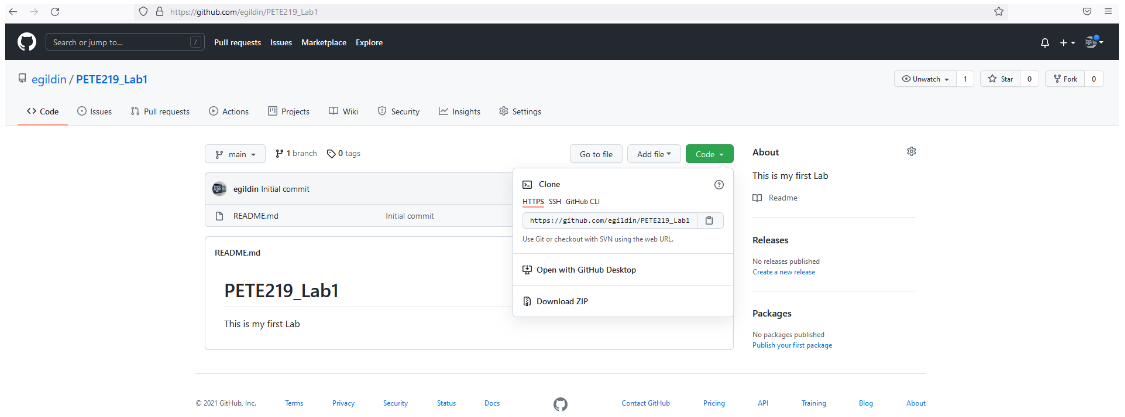
- Forking→ A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
- Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.



TEXAS A&M UNIVERSITY
Harold Vance Department of Petroleum Engineering

Note: Clone a repo vs Download a zip

There is a fundamental difference between clone vs download. When you **clone** a repo, you make a copy of the complete history of the git repo *including* the `.git` folder. When you **download** the repo, you just download the source files of the most recent commit of the default branch *without* the .git folder. Essentially, when you download you can't make use of any git commands, you can't view the commit messages, you are just not using git at all.

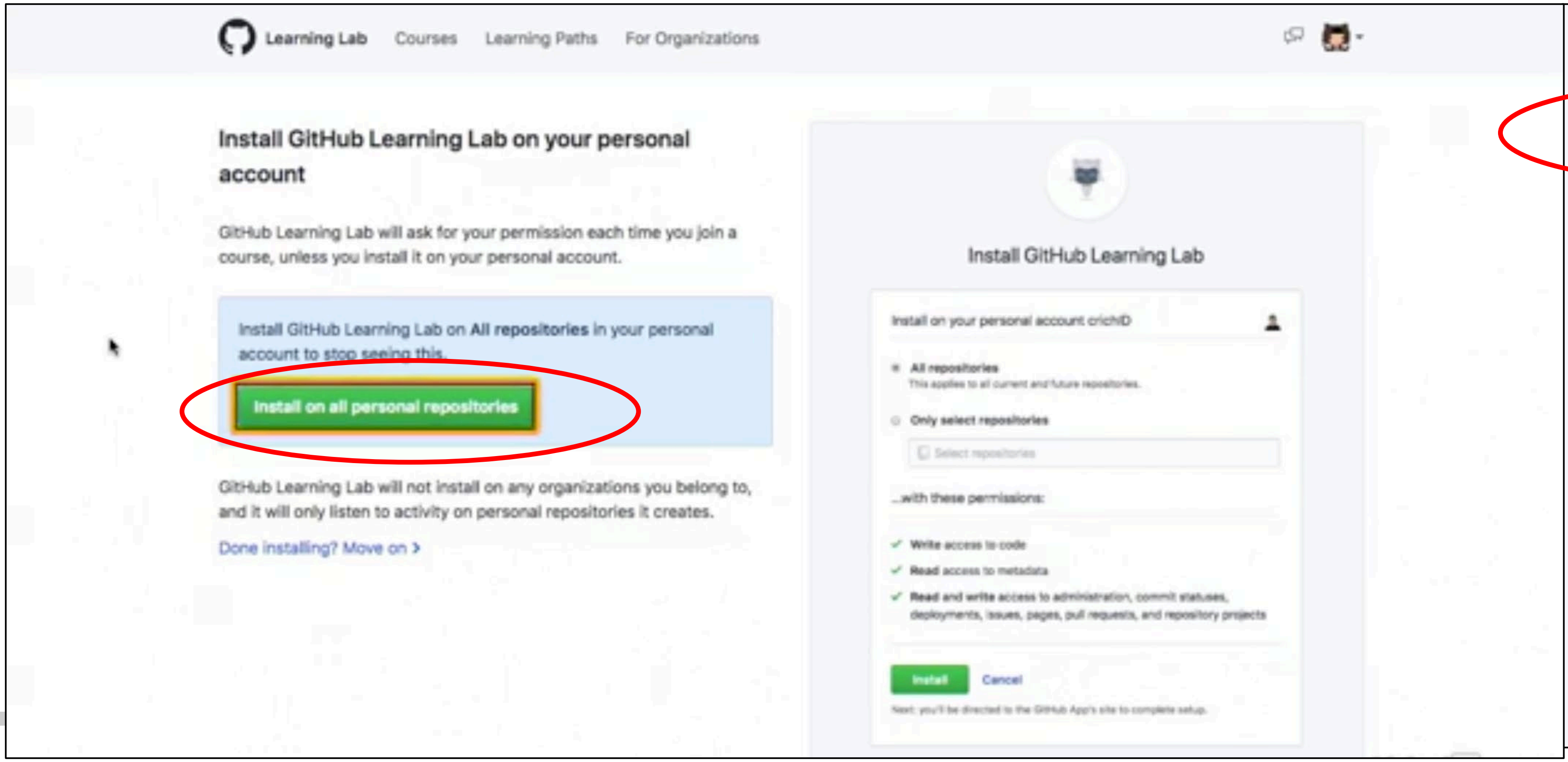


More training!!

1. Create a GitHub page from the GitHub Lab tutorial
   a. Follow step-by-step commands as described in:
   https://lab.github.com/githubtraining/introduction-to-github

Learning Lab - Github

- If not installed yet, you will need to install the app into your repo



Install GitHub Learning Lab on your personal account

GitHub Learning Lab will ask for your permission each time you join a course, unless you install it on your personal account.

Install GitHub Learning Lab on **All repositories** in your personal account to stop seeing this.

Install on all personal repositories

GitHub Learning Lab will not install on any organizations you belong to, and it will only listen to activity on personal repositories it creates.

Done installing? Move on >

Install GitHub Learning Lab

Install on your personal account crichID

[✓ **All repositories**]
This applies to all current and future repositories.

Only select repositories

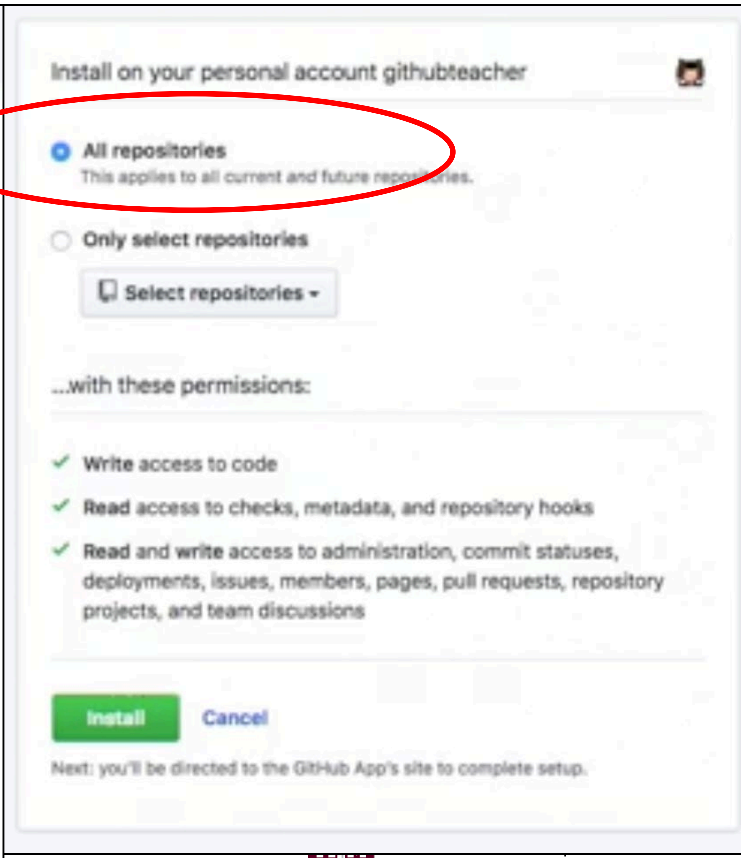
Select repositories

...with these permissions:

✓ **Write** access to code
✓ **Read** access to metadata
✓ **Read and write** access to administration, commit statuses, deployments, issues, pages, pull requests, and repository projects

Install Cancel

Next: you'll be directed to the GitHub App's site to complete setup.



Install on your personal account githubteacher

[✓ **All repositories**]
This applies to all current and future repositories.

Only select repositories

Select repositories

...with these permissions:

✓ **Write** access to code
✓ **Read** access to checks, metadata, and repository hooks
✓ **Read and write** access to administration, commit statuses, deployments, issues, members, pages, pull requests, repository projects, and team discussions

Install Cancel

Next: you'll be directed to the GitHub App's site to complete setup.