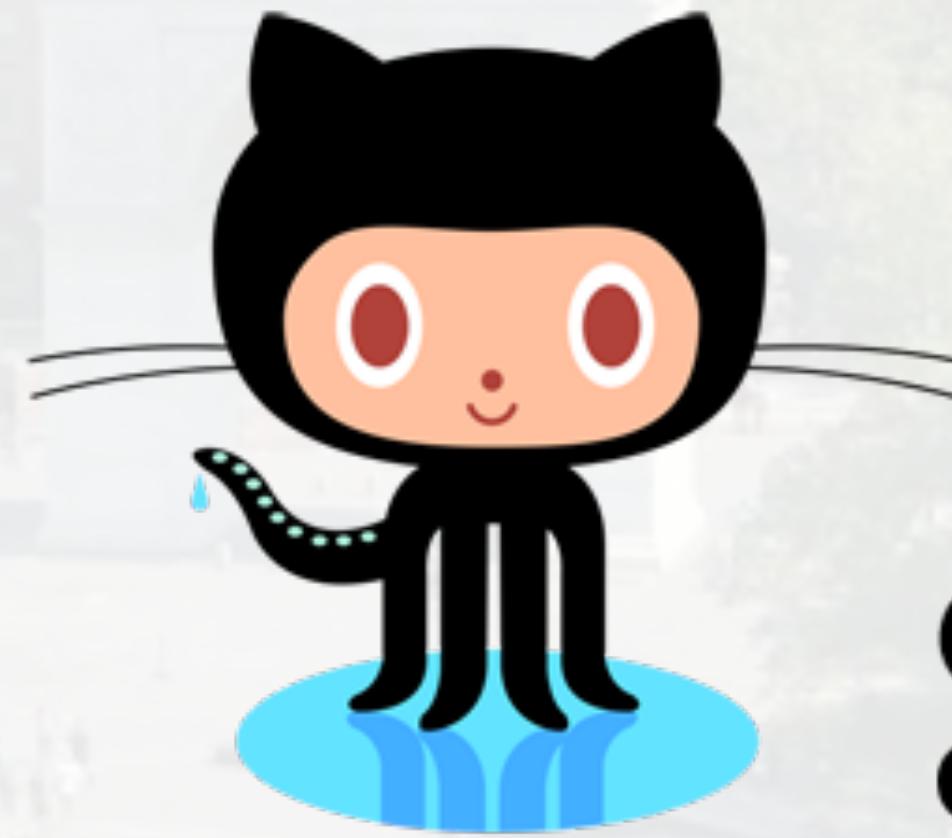


Social Coding With GitHub



github
SOCIAL CODING

Fall 2020, CSCI-GA 2820, Graduate Division, Computer Science

Instructor:
John J Rofrano

Senior Technical Staff Member | DevOps Champion
IBM T.J. Watson Research Center
rofrano@cs.nyu.edu (@JohnRofrano) 

Objectives

The objectives of this class are to:

- Learn the mechanics of using Git
- Introduce the Git Feature Branch Workflow
- Learn how Git encourages Collaboration
- Become a *Social Coder*



“What makes senior developers senior is not that they know the syntax of a given language better, but that they have experience working with large and complex projects with real users and business goals”

-Ariel Camus - medium

EDI



HOW TO BECOME A GIT JEDI



HOW TO BECOME A GIT JEDI

Rule #1: Create a Git repository for every new project



HOW TO BECOME A GIT JEDI

Rule #1: Create a Git repository for every new project

Rule #2: Create a new branch for every new feature



HOW TO BECOME A GIT JEDI

Rule #1: Create a Git repository for every new project

Rule #2: Create a new branch for every new feature

Rule #3: Use Pull Requests to merge code to Master

Source Code Management

- Source Code Management is the practice of tracking versions of source code as it is being developed
- A source code manager (SCM) is a software tool used by teams of programmers to manage source code
- These are also referred to as Version Control Systems (VCS)
- SCMs can be Centralized or Distributed



History of Version Control Systems

- **First Generation:** SCCS (Source Code Control System) RCS (Revision Control System)
 - Local changes tracked for individual files and checked-out files could only be edited locally by one user at a time
- **Second Generation:** CVS (Concurrent Versions System) SVN (Apache Subversion) Perforce Helix Core
 - Centralized version control systems introducing networking which led to centralized repositories that contained the 'official' versions of their projects.
- **Third Generation:** Git. Mercurial. BitKeeper. Darcs (Darcs Advanced Revision Control System)
 - Distributed VCS, all copies of the repository are created equal - there is no central copy of the repository
 - This opens the path for commits, branches, and merges to be created locally without network access and pushed to other repositories as needed

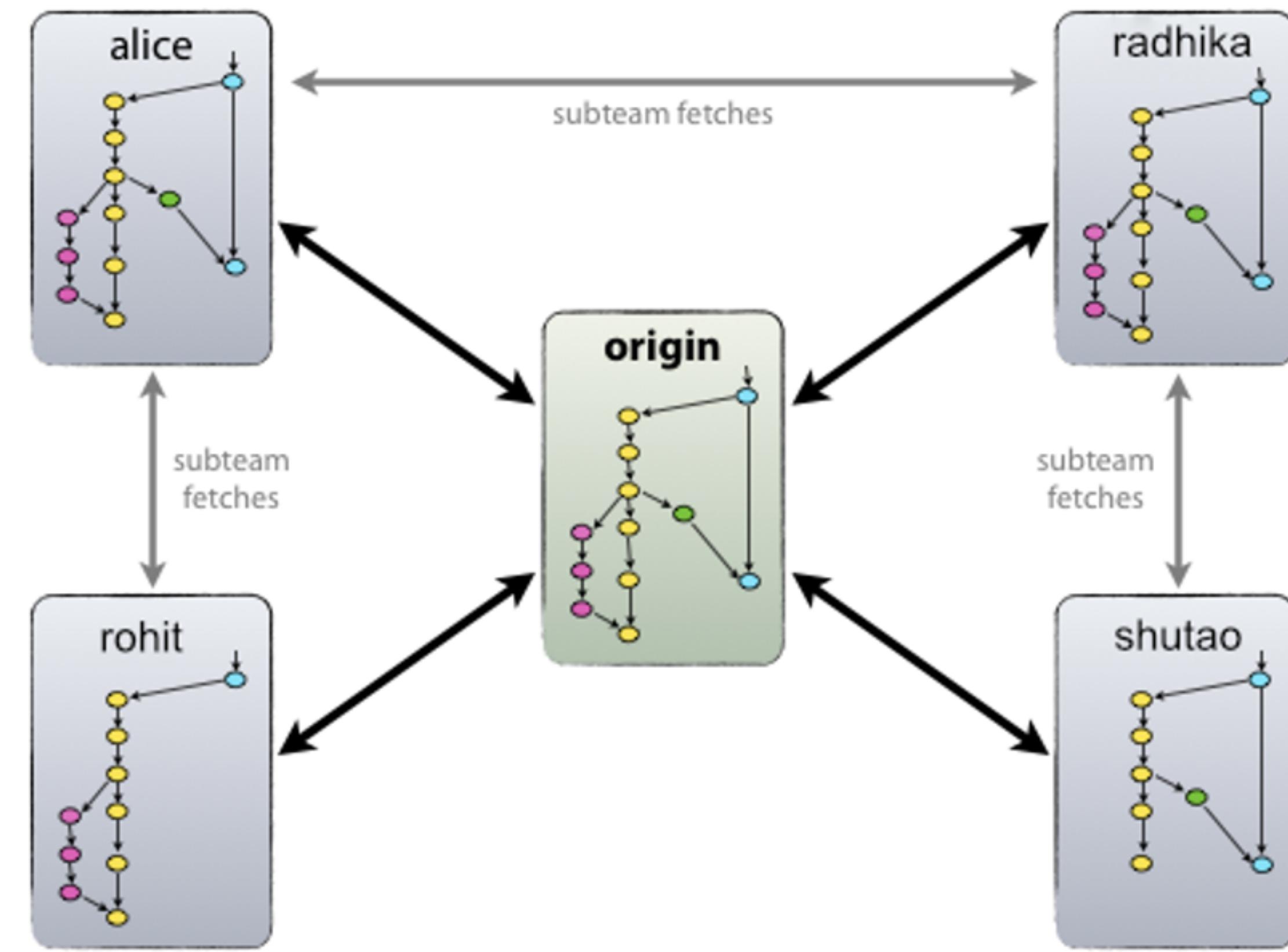
Distributed or Centralized

- Distributed revision control (DRCS) takes a peer-to-peer approach, as opposed to the client-server approach of centralized systems
- Rather than a single, central repository on which clients synchronize, each peer's working copy of the codebase is a bona-fide repository
- Distributed revision control conducts synchronization by exchanging patches (change-sets) from peer to peer



Decentralized systems Like Git

- No canonical, reference copy of the codebase exists by default; only working copies.
- Common operations (such as commits, viewing history, and reverting changes) are fast, because there is no need to communicate with a central server.
- Communication is only necessary when pushing or pulling changes to or from other peers.
- Each working copy effectively functions as a remote backup of the codebase and of its change-history, providing natural protection against data loss.



Git

- A distributed source code management (SCM) tool invented by Linus Torvalds in 2005 for Linux kernel development
- Code is kept in a repository and every developer has a full copy
- Works locally without any server
- Works remotely with GitHub, GitLab, & BitBucket



Git

- The first distributed version control that changed the workflow:
 - A switch fundamental from the developer asking, “Can you add me to version control?”
 - To making their own copy, changing the code, and then checking in their contribution.
- This change was revolutionary in that developers no longer needed to be invited to contribute to open source code repositories as well as proprietary code.



GitHub

- A web site that hosts git repositories
- Founded in 2007, has 40 million registered developers and was acquired by Microsoft for a whopping \$7.5 billion in 2018
- Free and Paid accounts
- Adds ability to track Issues and Bugs
- Provides webhooks to integrate other tools
- ...and a whole lot more



Bitbucket

- A web site that hosts git repositories
- Founded in 2010 and owned by Atlassian
- Free and Paid accounts
- Adds ability to track Issues and Bugs
- Provides webhooks to integrate other tools
- ...and a whole lot more



Bitbucket

GitLab

- Founded in 2011 as an OpenSource competitor to GitHub
- You can run your own GitLab servers for free
- Also runs a web site to host git repositories
- Adds ability to track Issues and Bugs
- Provides webhooks to integrate other tools
- ...and a whole lot more



GitLab

We will use GitHub for this class

...but we could have used any one of these



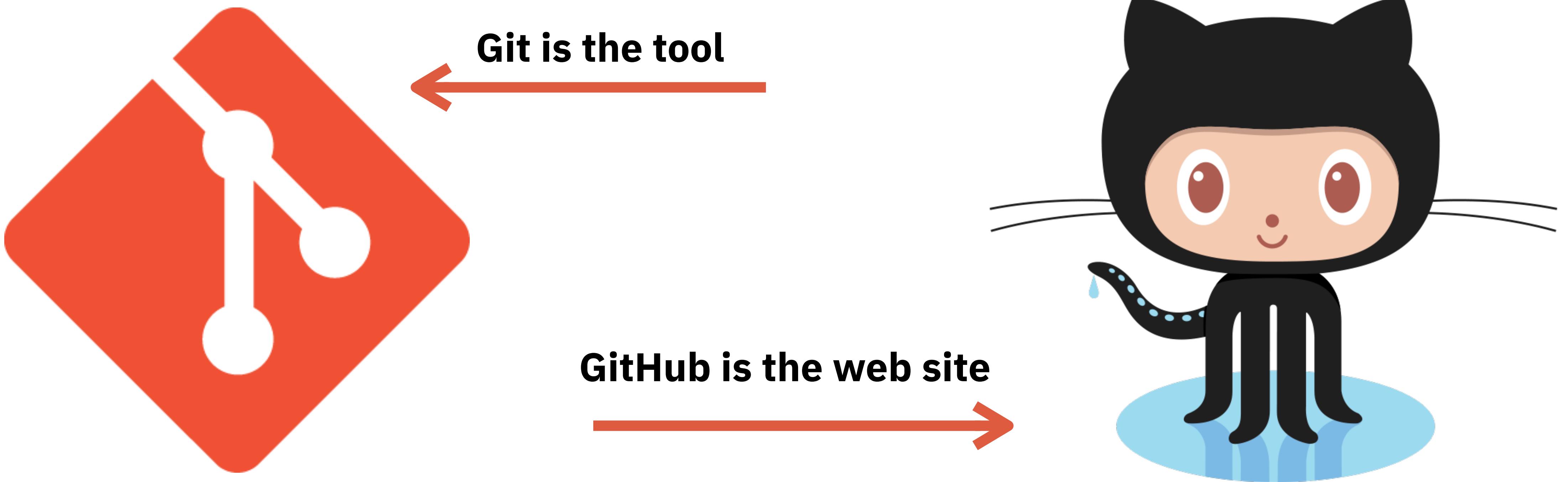
GitLab



Bitbucket



Git vs GitHub



Why You Need to Know Git

- The use of Git and software code repositories has fundamentally changed the way enterprises do software development for proprietary code and open source code alike.
- In addition, open source software has blossomed because some of the Git repositories host open source code for free.
- Most significantly, Git and code repositories have facilitated the DevOps methodology that is so impactful to software development.

How many of you know Git?



How many of you know Git?



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 MEMORIZIZE 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.





Git Command Workflow



workspace

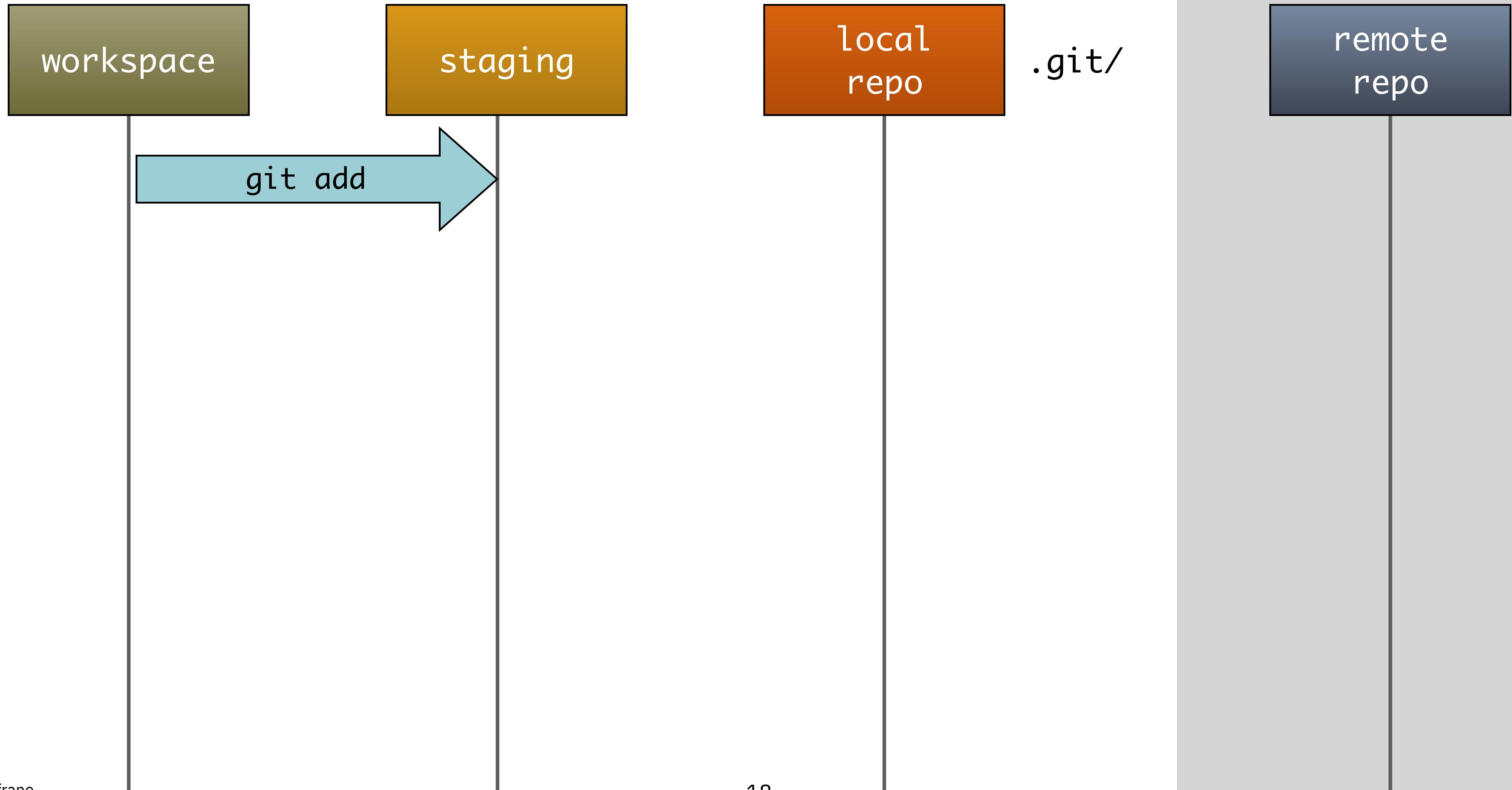
staging

local
repo .git/

remote
repo

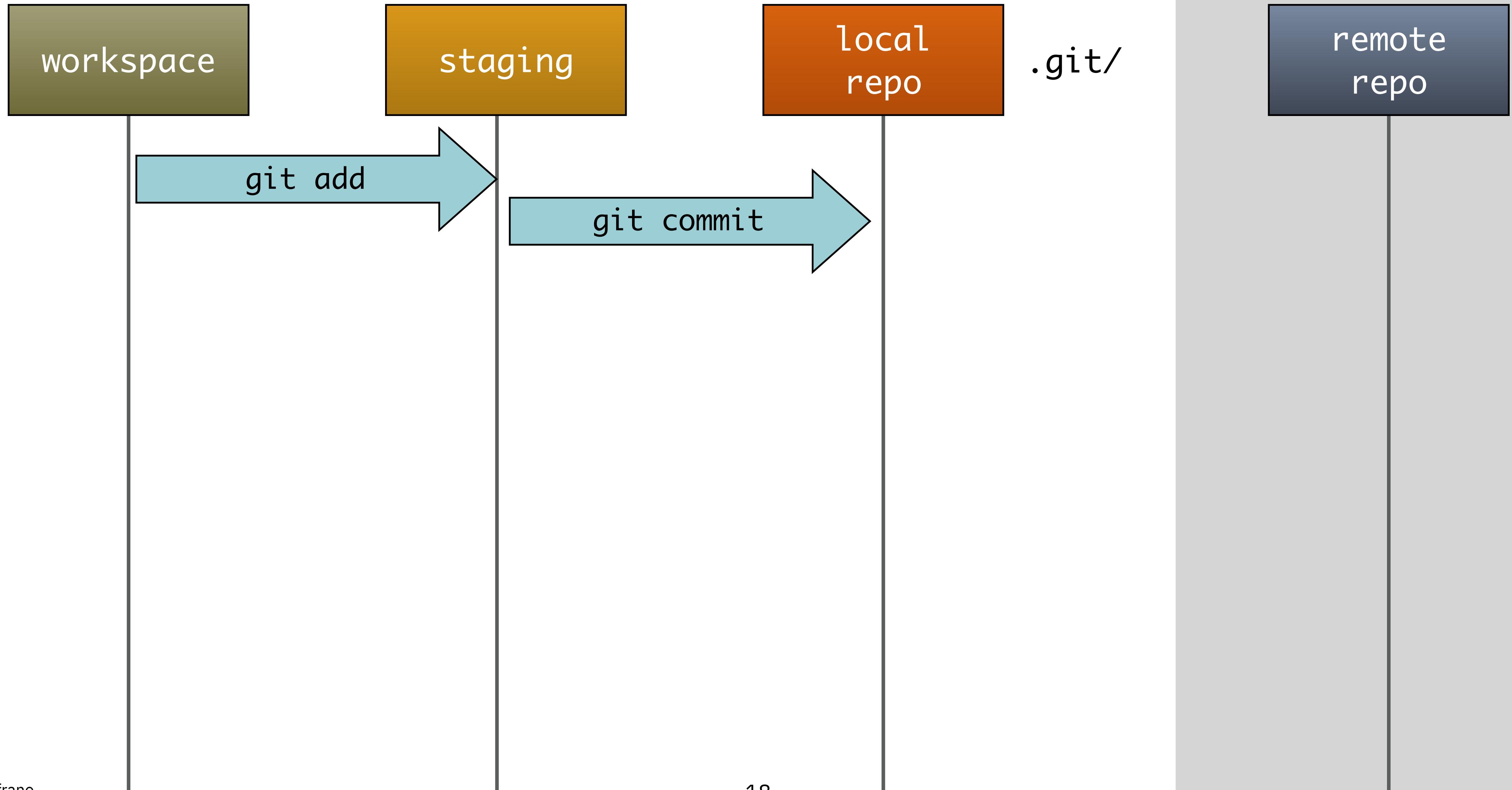


Git Command Workflow



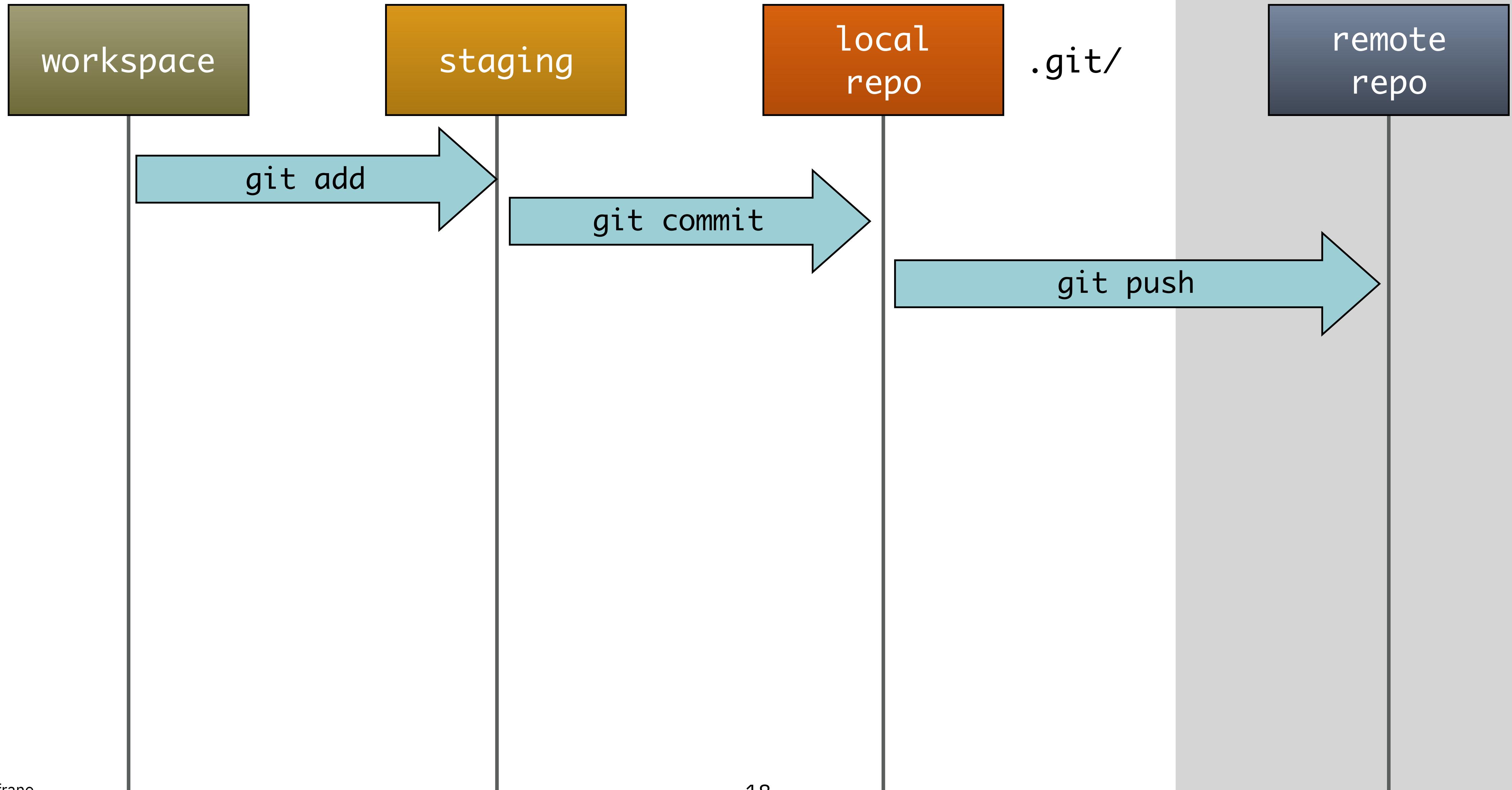


Git Command Workflow



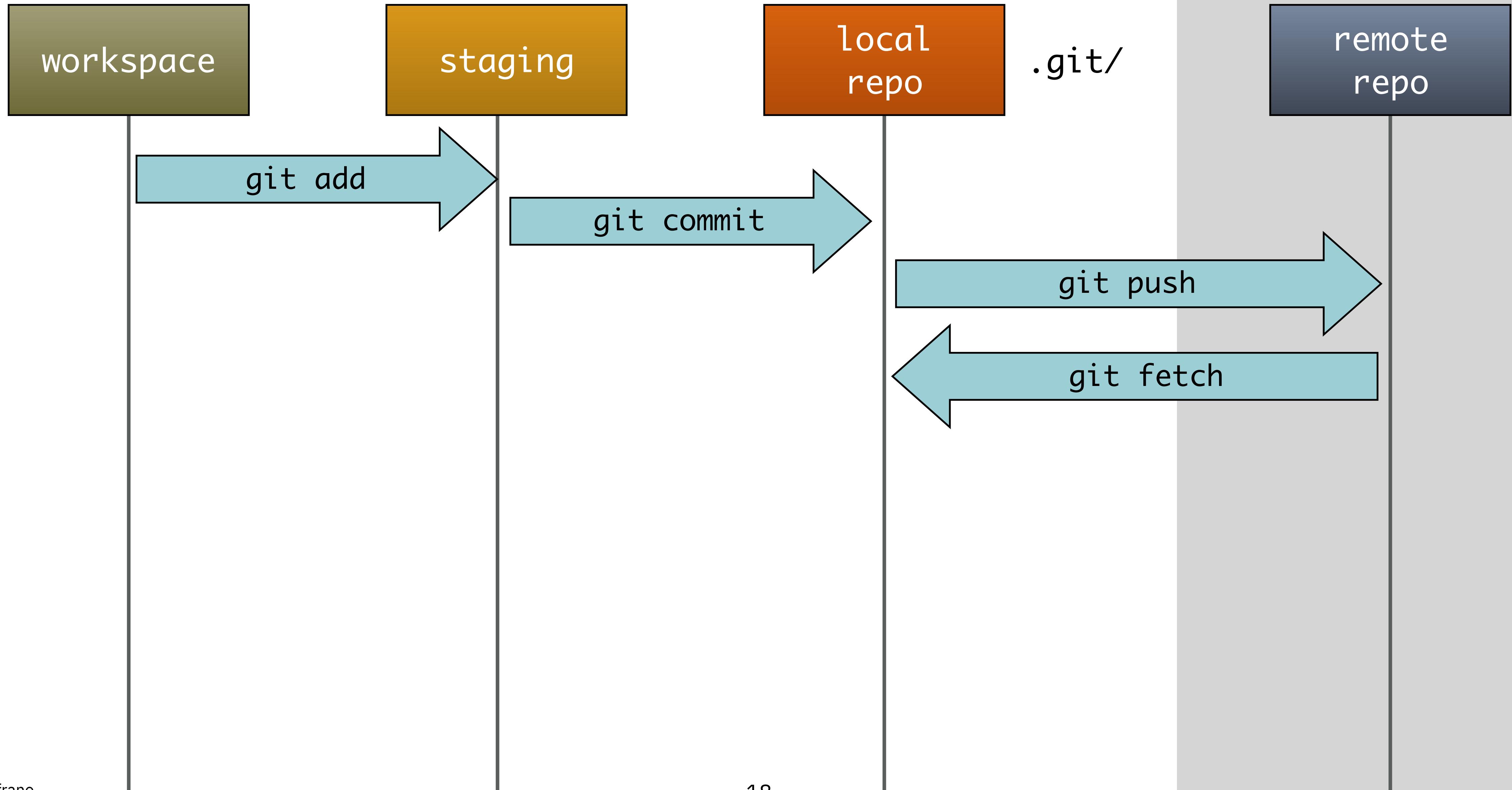


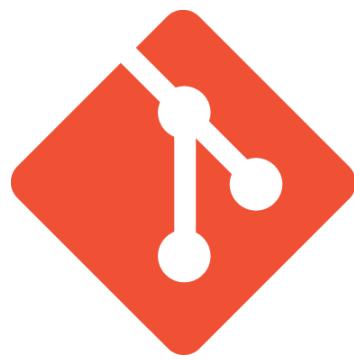
Git Command Workflow



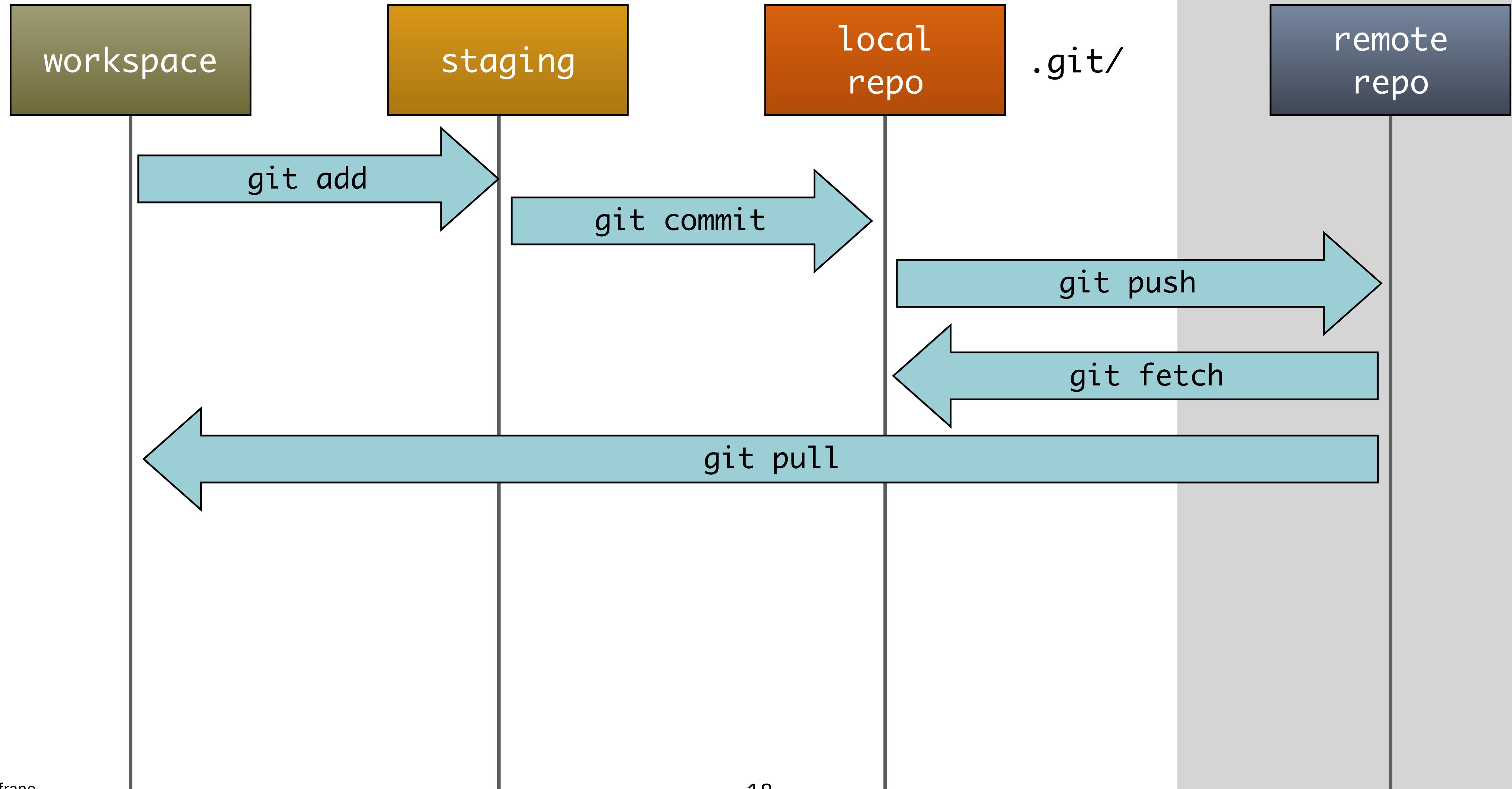


Git Command Workflow



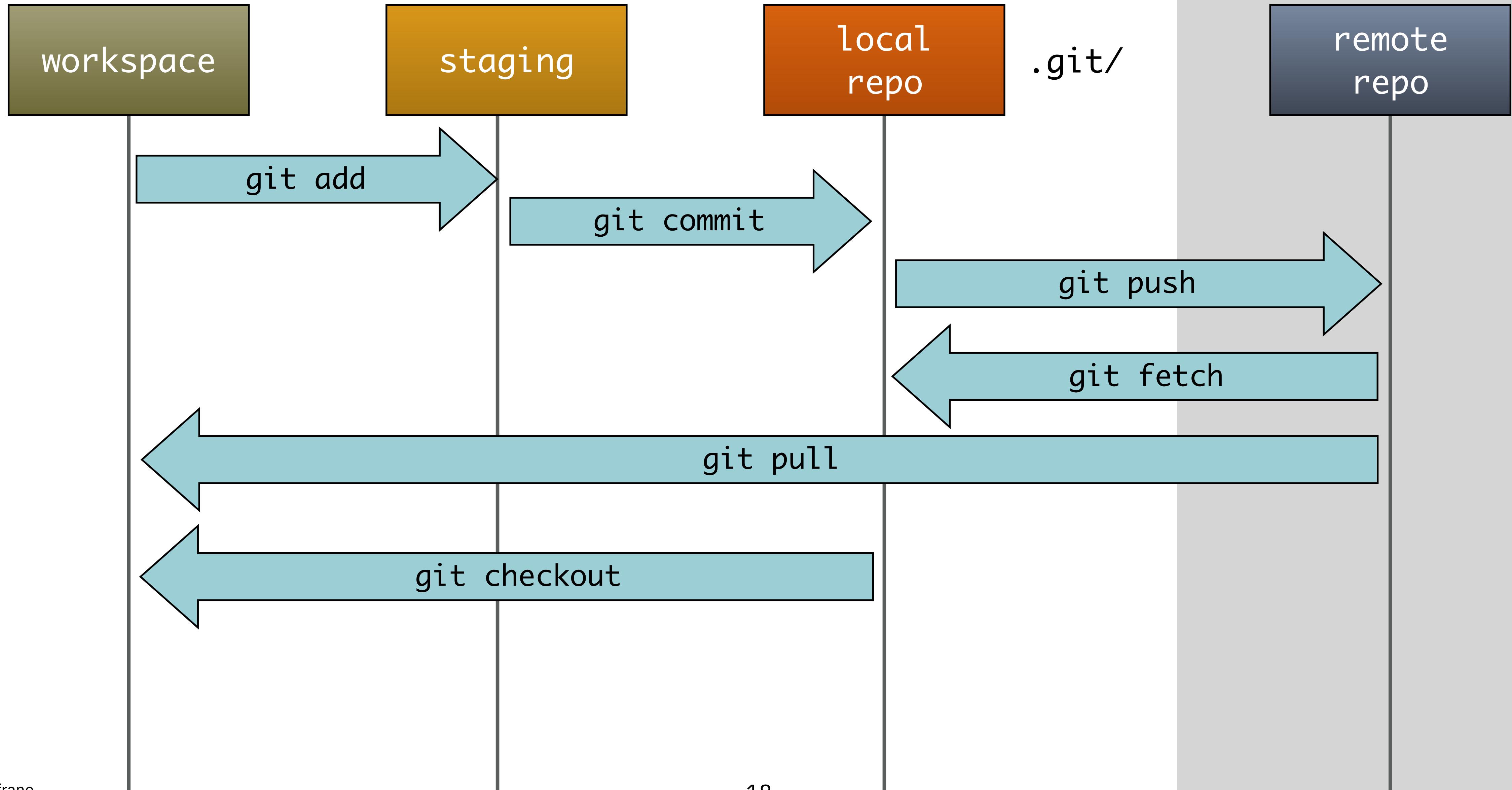


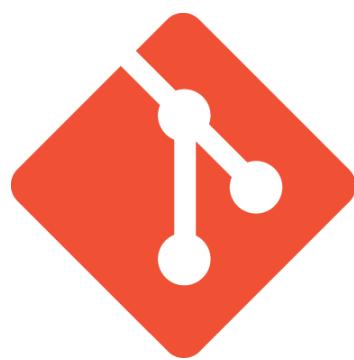
Git Command Workflow



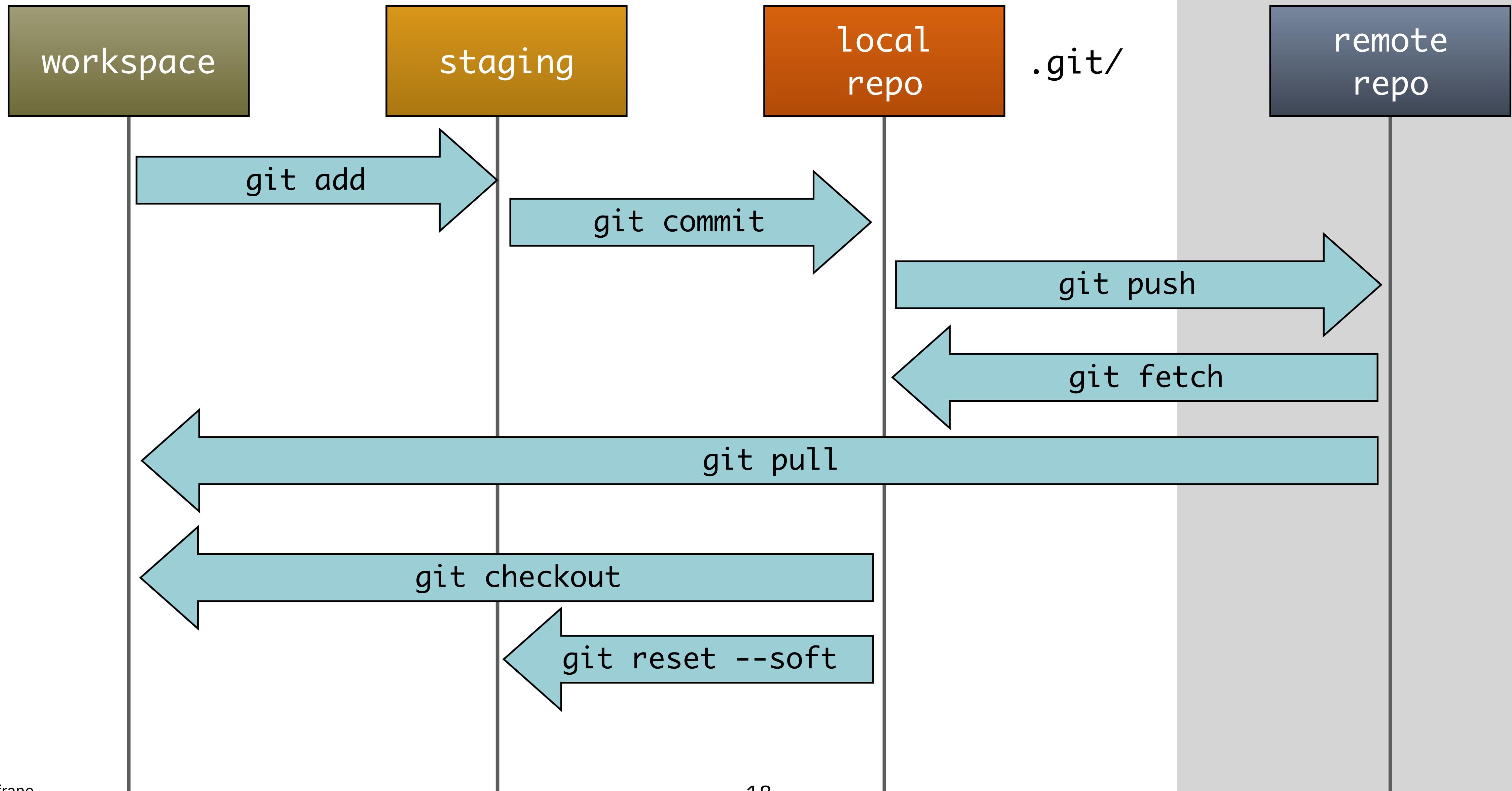


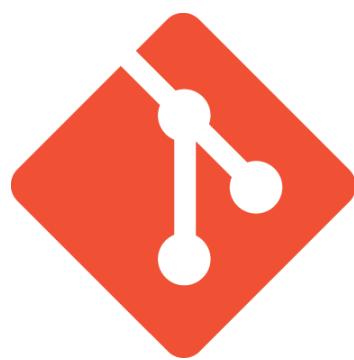
Git Command Workflow



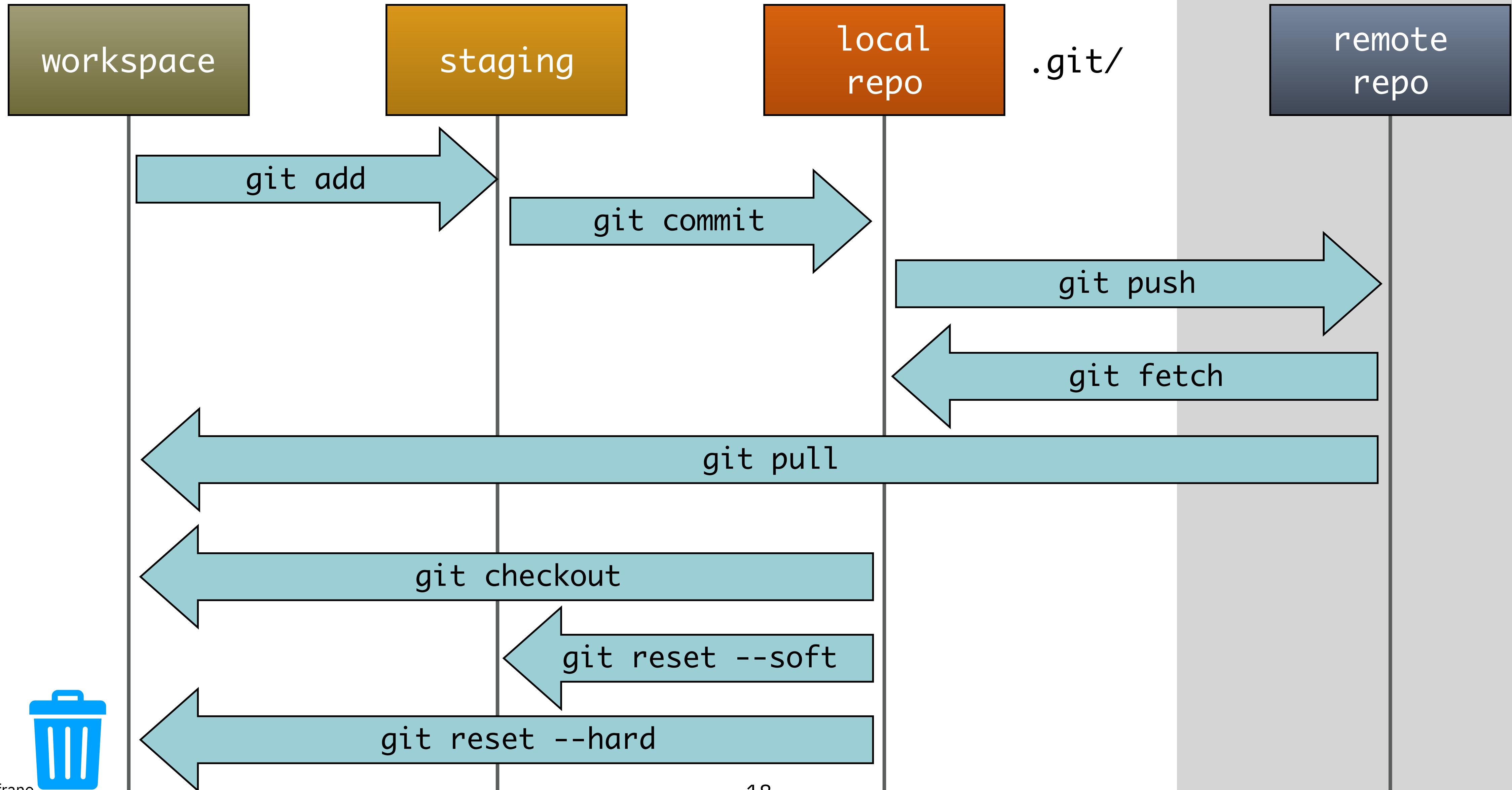


Git Command Workflow





Git Command Workflow



github

SOCIAL CODING

BUILD SOFTWARE BETTER
TOGETHER



What is Social Coding?

- In the past developers worked on **private** repositories and you had to be a member of the team to contribute
- With Social Coding, repositories are **public** and everyone is encouraged to Fork the code and contribute
- You would think that anarchy would ensue but it actually works quite well because it is controlled by the repository owner



Code reuse Dilemma

- You see a project that is 80% of what you need but there are some missing features
- You feel that if you make a feature request of the project owner, your request will be at the bottom of their priorities
 - If they get funding cuts, you know that your feature request will be cut
- So you rebuild 100% of what you need so as not to have a dependency on another project



Social Coding Solution

- Discuss the new feature with the repo owner and agree to develop it
- Open an **Issue** and assign it to yourself so that everyone knows what you are working on
- **Fork** the code, create a **branch**, and make your changes
- Issue a **Pull Request** when you are ready to review and merge your work back into the main project



Style Guide for Python Code

- PEP 8 -- Style Guide for Python Code
 - <https://www.python.org/dev/peps/pep-0008/>
- Coding Standards are critical so that the code looks consistent even though it is written by multiple contributors
- PyLint can be used to enforce PEP8 plus more

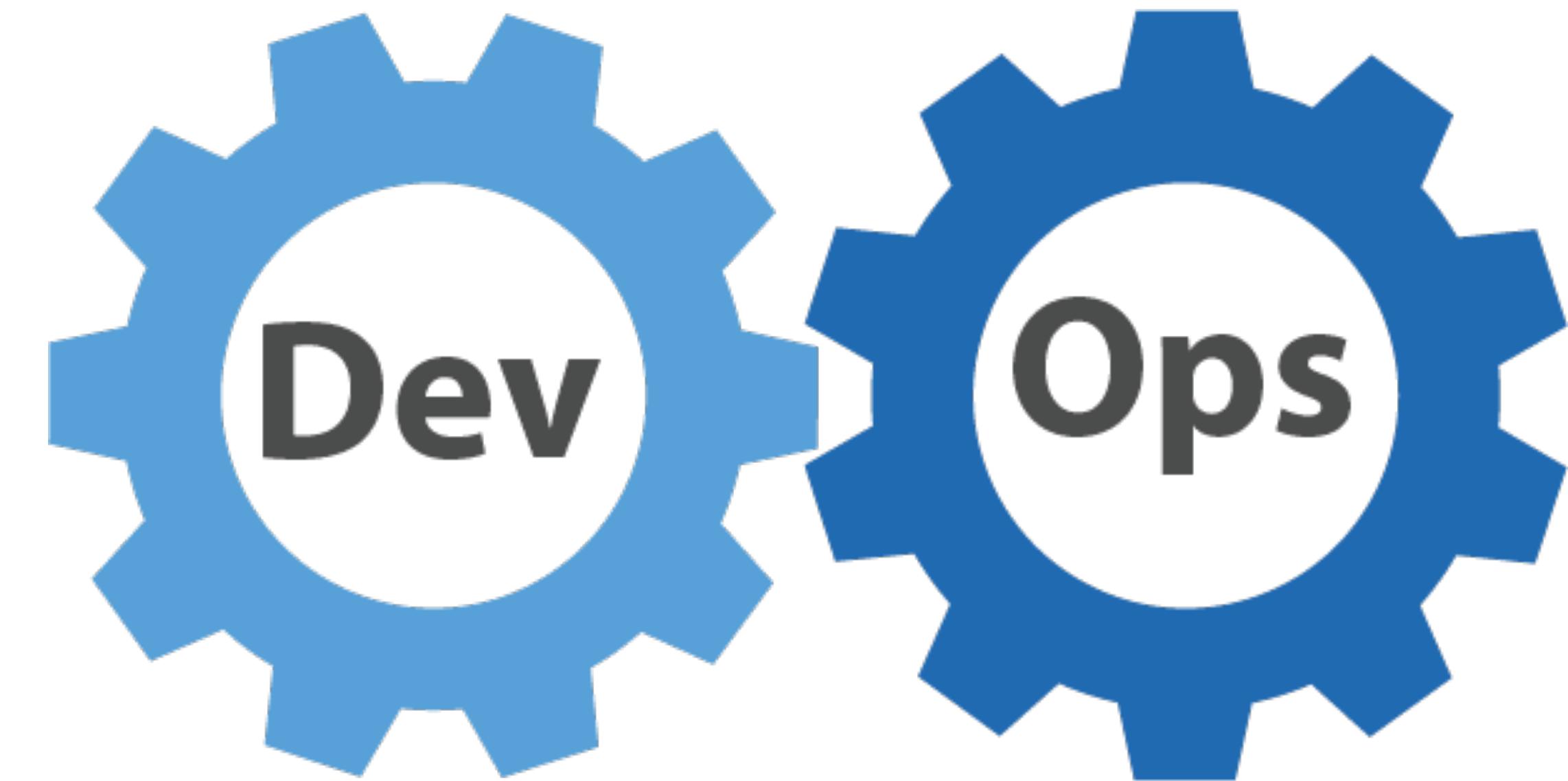


Why Git and DevOps?



GitHub Is A DevOps Enabler

- Open Culture
 - ✓ Forking and Pull Requests enable openness
- Emphasize Collaboration and Sharing (no Silos)
 - ✓ Issues and Comments invite Collaboration and Planning
- Infrastructure as Code
 - ✓ Git is the perfect place to store all of your infrastructure scripts
- Automate Everything
 - ✓ Git works well with Travis CI, Jenkins, and other automation tools



Git Overview

- Git is a Decentralized Source Code Management (SCM) System
- Developers work in their own BRANCHES (even in FORKS)
- The MASTER branch should always be ready to deploy
- PULL REQUESTS are used to MERGE code BRANCHES into MASTER



Huh?

Branches?

Forks?

Merge?

Spoons?

Master?

Pull Requests?

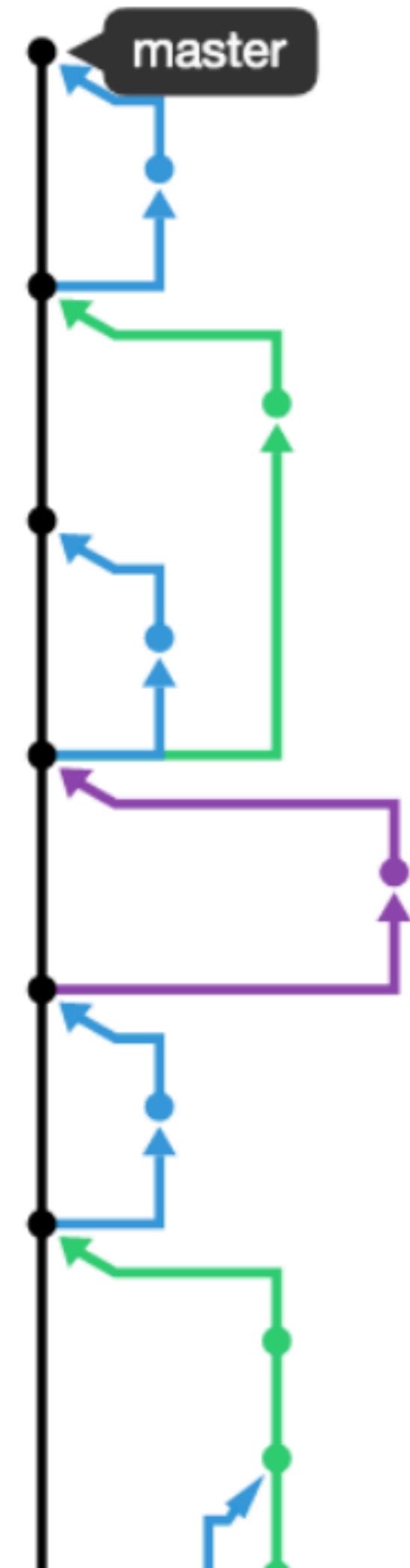
Pulling hair?

What?

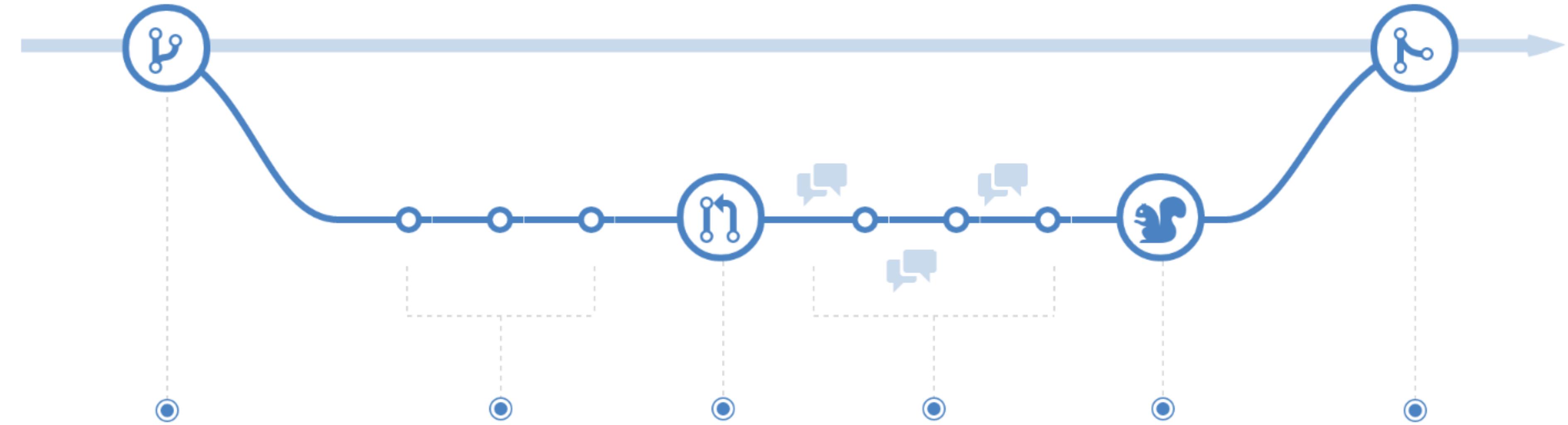


Git Feature Branch Workflow

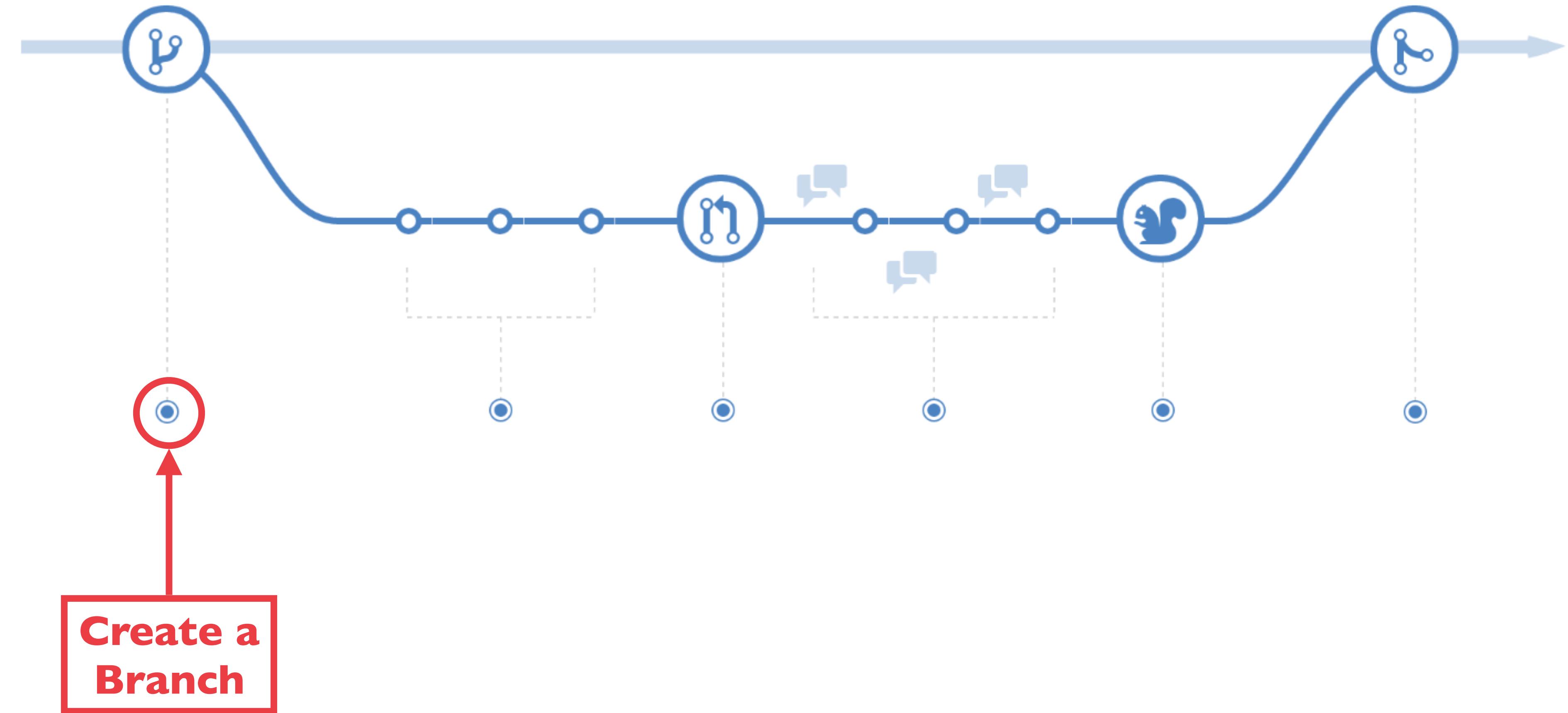
1. CREATE a new repository or FORK an existing repository
2. CLONE it to your local system
3. Create a BRANCH to work on your ISSUE
4. COMMIT changes to that branch
5. PUSH your changes to the Remote branch
6. Issue a PULL REQUEST to have your work reviewed
7. MERGE your code to master and close the ISSUE



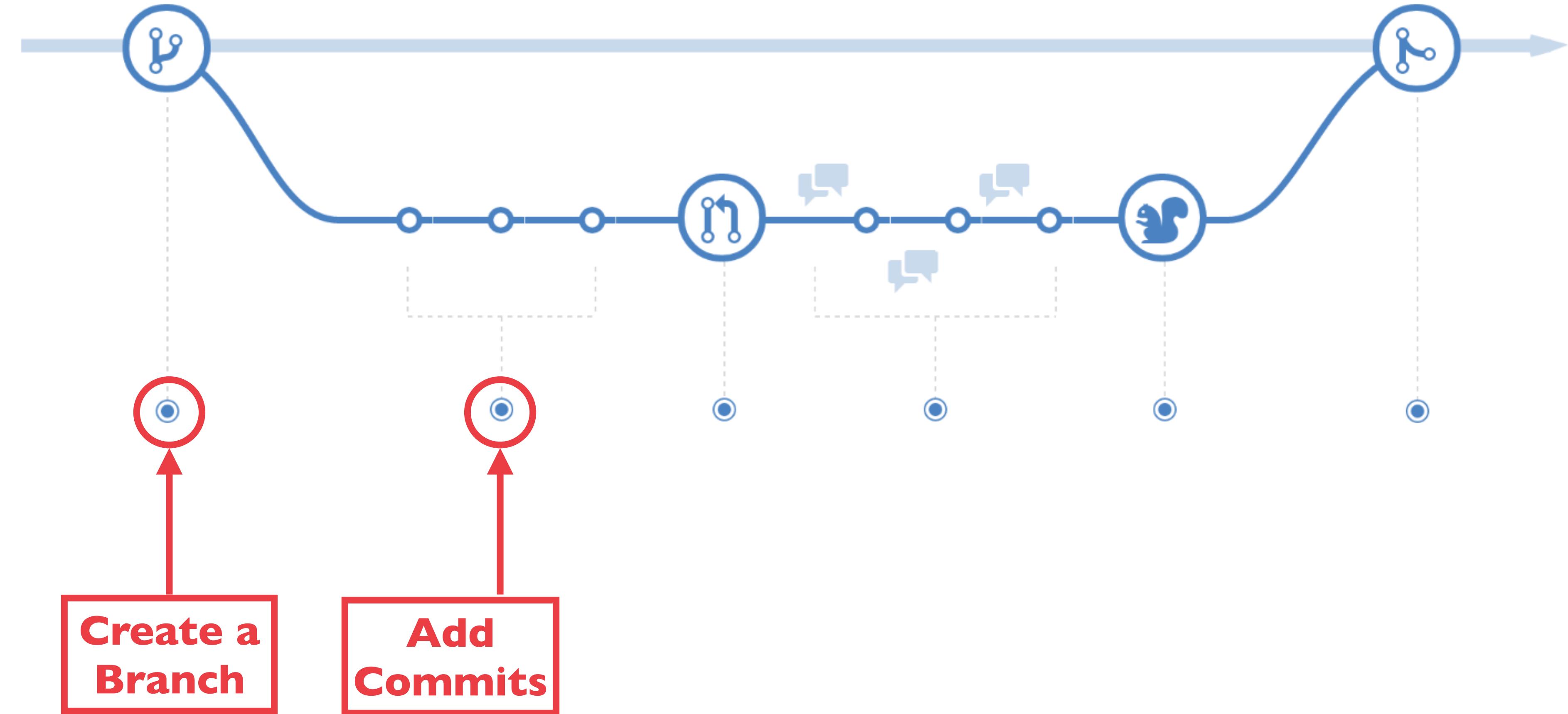
Git Feature Branch Workflow



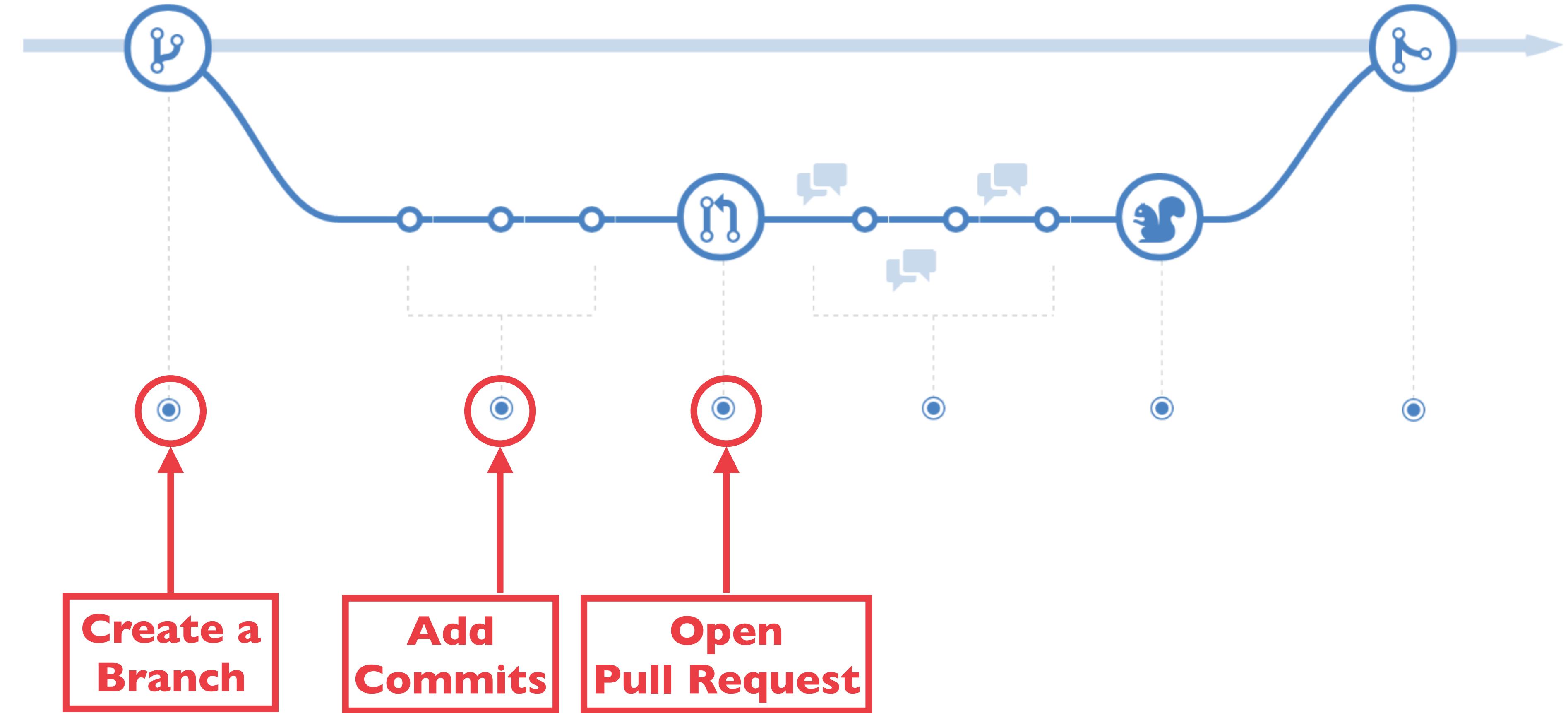
Git Feature Branch Workflow



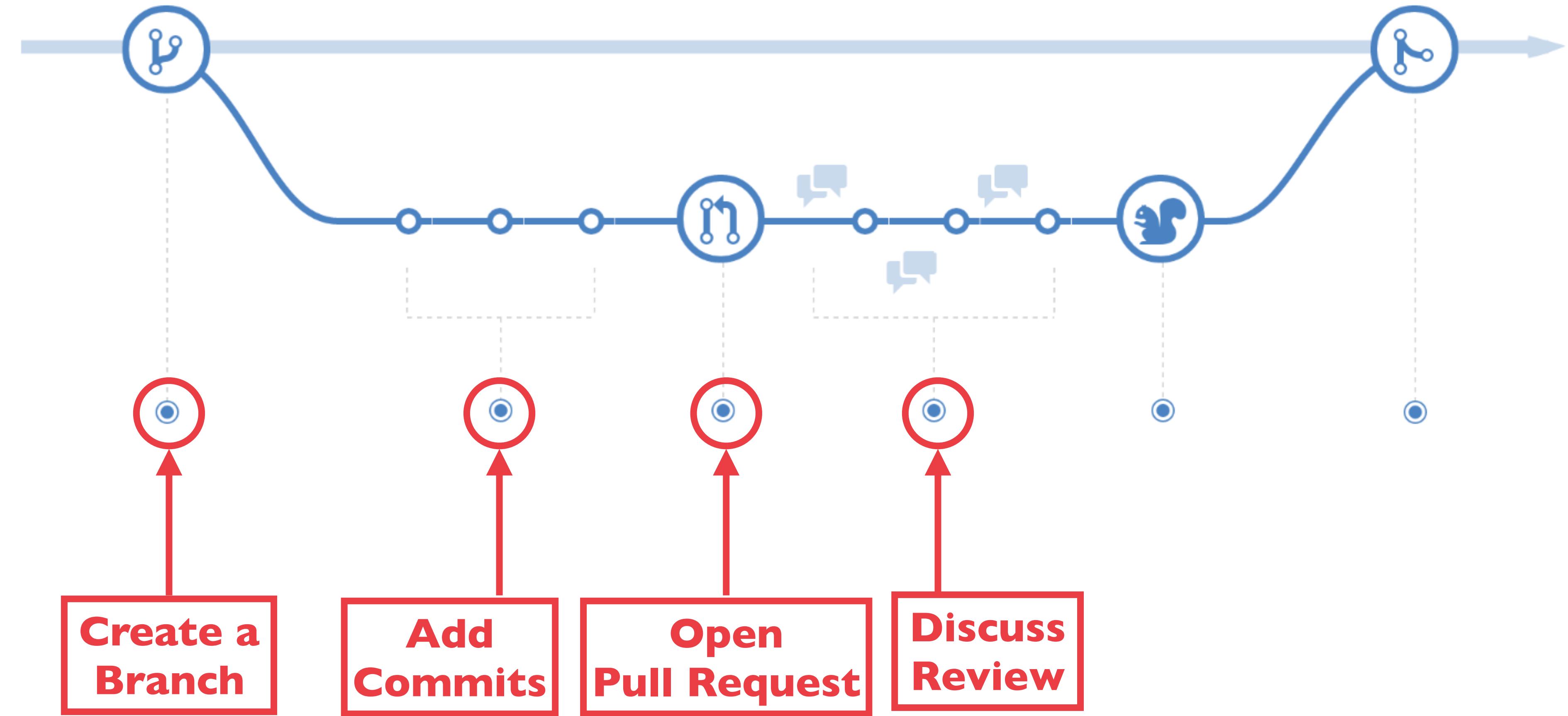
Git Feature Branch Workflow



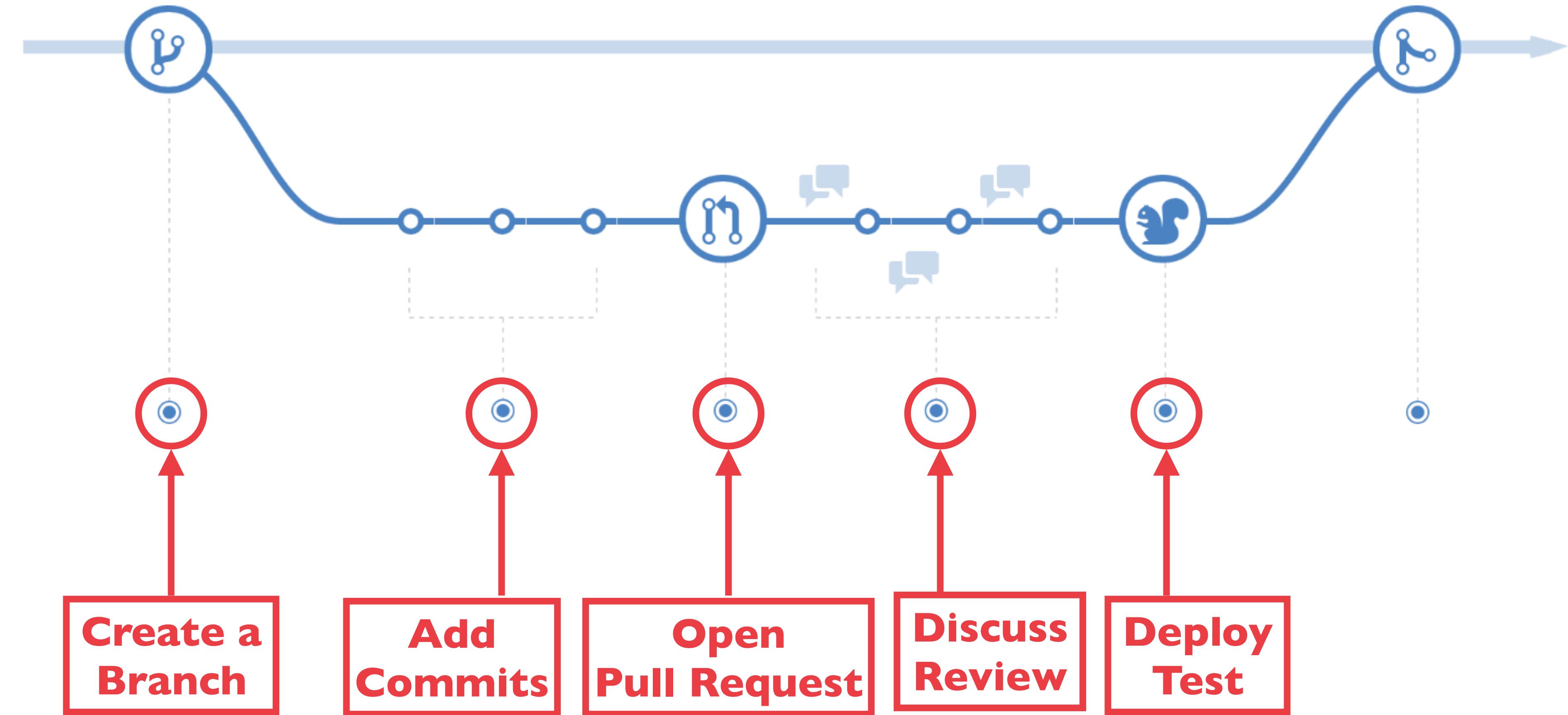
Git Feature Branch Workflow



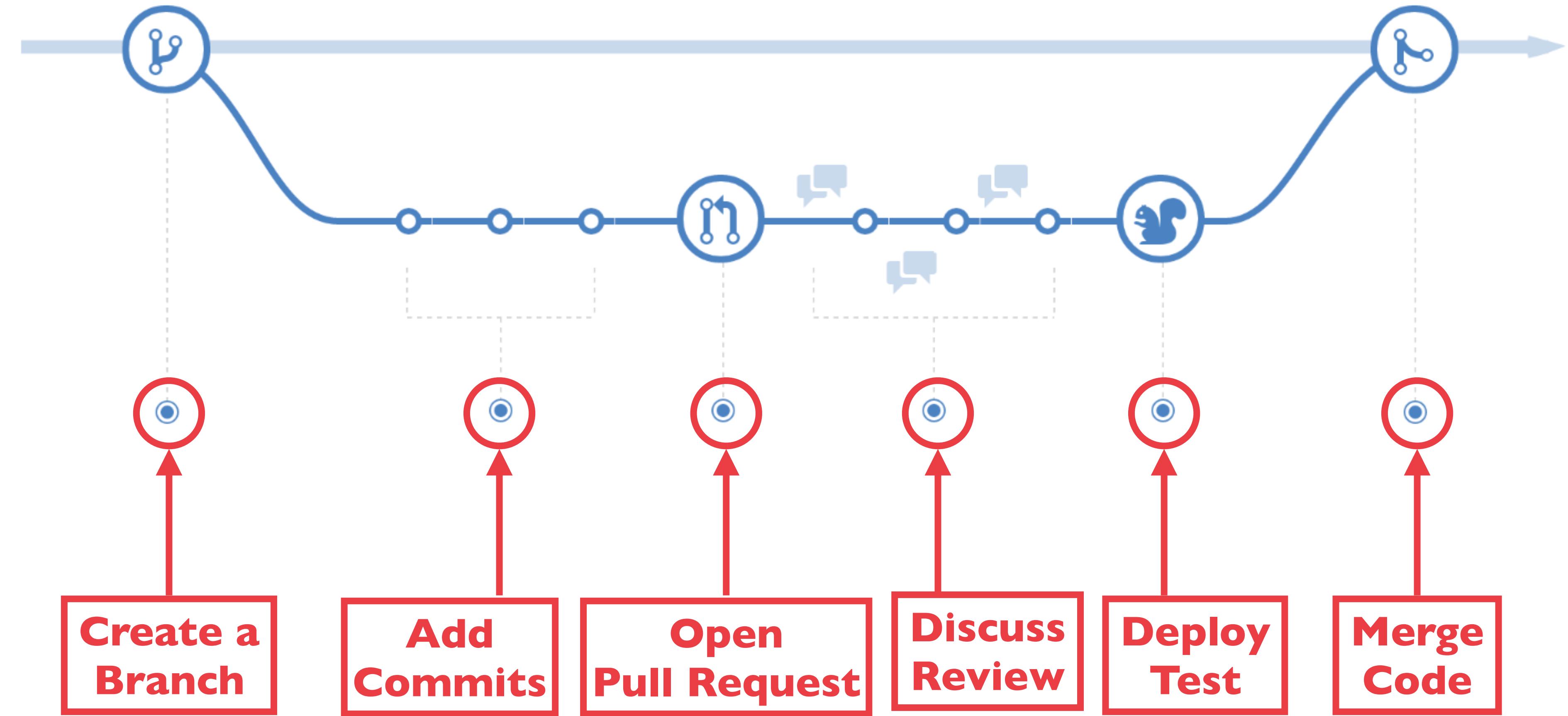
Git Feature Branch Workflow



Git Feature Branch Workflow

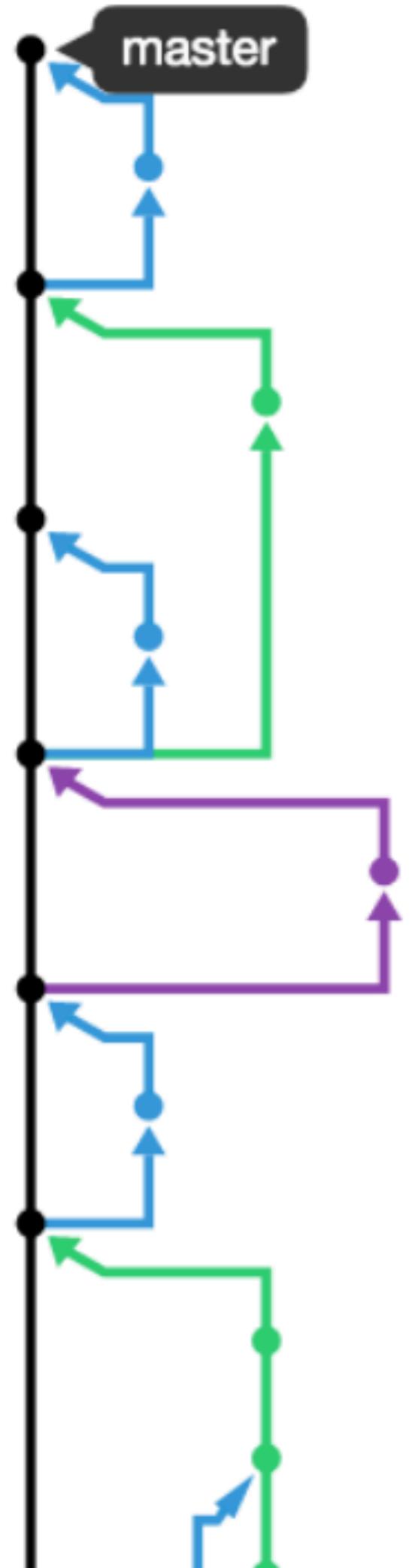


Git Feature Branch Workflow

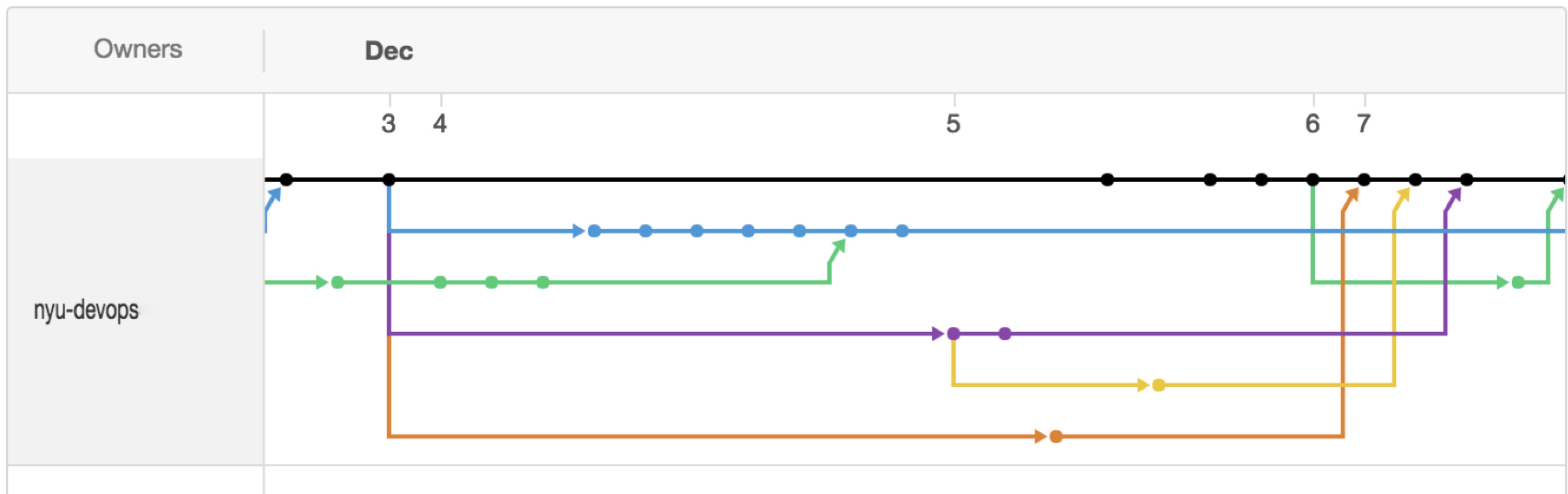


Detailed Feature Branch Workflow

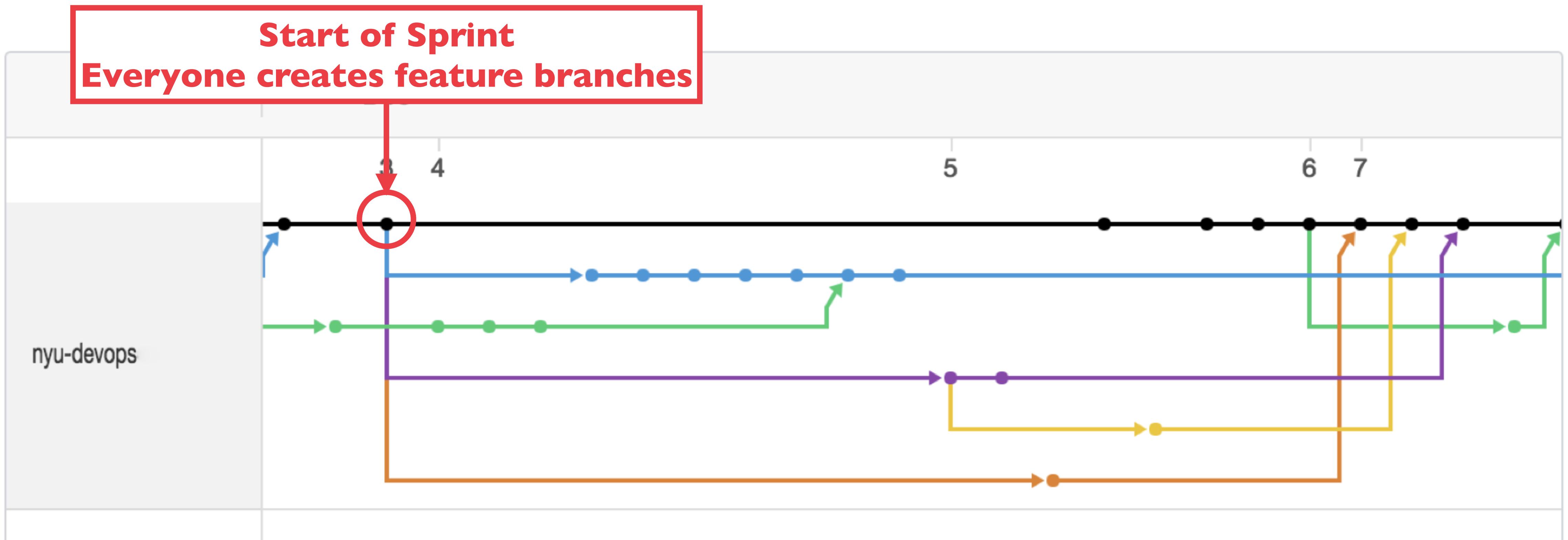
- CLONE a Repository (FORK first if not part of Dev Team)
- Assign the ISSUE for your work to yourself and place it in working status
- Create a BRANCH to work on an ISSUE
- Run the Test suite to make sure you can run the code
- Make changes to code and test cases and COMMIT to local BRANCH
- Run the Test suite early and often to make sure you didn't break anything
- PUSH changes to remote BRANCH
- Did we mention testing the code early and often?
- Create PULL REQUEST when all tests pass and code is ready for review / MERGE



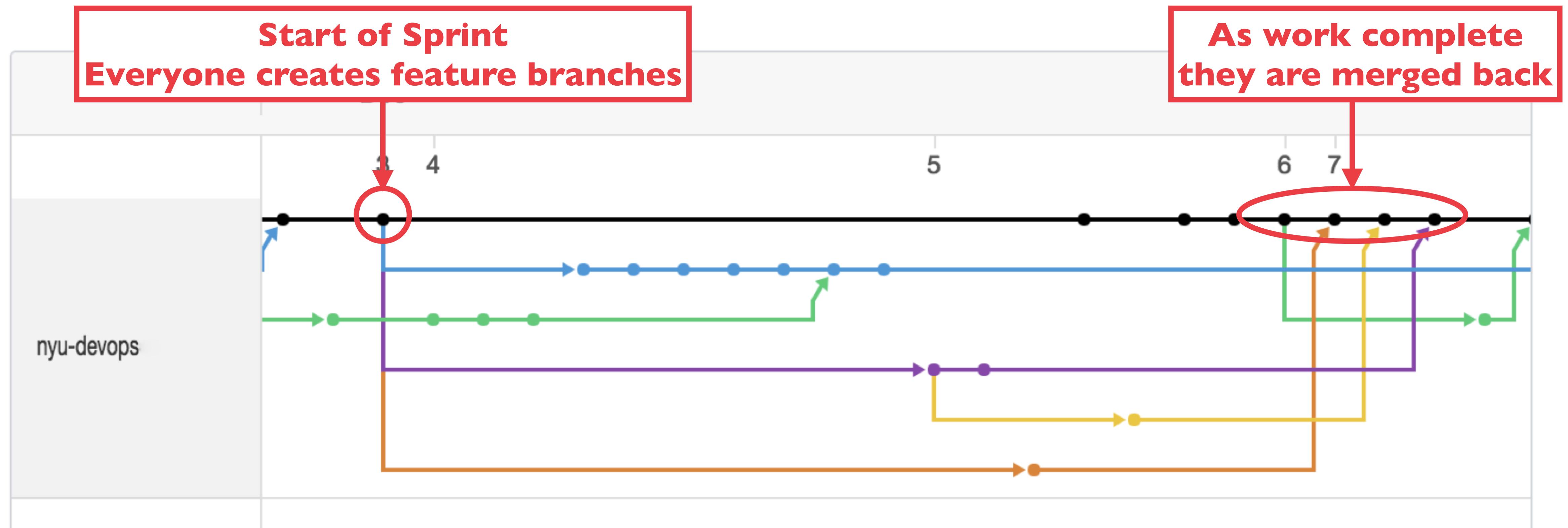
Branching example



Branching example



Branching example



Example workflow

- Your initial workflow should set up a remote branch that is tracked

```
git checkout -b my-new-feature
... write some code here ...
git add .
git commit -m 'initial working version of my new feature'
git push -u origin my-new-feature
```

- The next day you add more code and push to your remote branch

```
... write some code here ...
git add .
git commit -m 'added this cool code...'
git push
```

Example workflow

- Your initial workflow should set up a remote branch that is tracked

```
git checkout -b my-new-feature  
... write some code here ...  
git add .  
git commit -m 'initial working version of my new feature'  
git push -u origin my-new-feature
```

Sets up a remote branch and tracks it

- The next day you add more code and push to your remote branch

```
... write some code here ...  
git add .  
git commit -m 'added this cool code...'  
git push
```

Example workflow

- Your initial workflow should set up a remote branch that is tracked

```
git checkout -b my-new-feature  
... write some code here ...  
git add .  
git commit -m 'initial working version of my new feature'  
git push -u origin my-new-feature
```

Sets up a remote branch and tracks it

- The next day you add more code and push to your remote branch

```
... write some code here ...  
git add .  
git commit -m 'added this cool code'  
git push
```

No need to use the branch name because it is tracked

Why is Testing So Important?

- Automated testing must pass before your Pull Request will be accepted
- Automated testing is your guarantee that you didn't break anything
- Without automated testing you cannot fully automate the DevOps pipeline



Hands-On

“live follow-along session”

Some Assembly Required

- Tools you will need to complete this lab:
 - GitHub Account
 - <http://github.com>
 - Git command line client
 - <https://git-scm.com/downloads>



.gitconfig

- It's important to make sure that you have git configured correctly with your `user.name` and `user.email`
- You can use `git config -l` to list the values
- You can use `git config name value` to change them

```
git config --global user.name "Your Name Goes Here"
```

```
git config --global user.email you@example.com
```



Windows Users

- This will ensure that Windows doesn't use crlf or flip the execute bit on all of your files (because Windows doesn't support an execute bit so it doesn't care)

```
$ git config --global core.fileMode false  
$ git config --global core.eol lf  
$ git config --global core.autocrlf input
```

Git Credentials

- If you are going to use HTTPS URLs when cloning Git repos, then you should setup the git command line to use a credential store
- On Windows

```
$ git config --global credential.helper wincred
```

- On MacOs you can use the OS X Keychain.

```
$ git config --global credential.helper osxkeychain
```



SSH Keys



- If you haven't set up an ssh key to work with GitHub then you should do this now.
In your Terminal window type the following:

```
$ ssh-keygen -t rsa -b 4096 -C "youremail@nyu.edu"
```

- Just press ENTER at all of the prompts. This will generate a private and public ssh key that you can use with GitHub

Add SSH key to GitHub

The screenshot shows a web browser window with the GitHub URL in the address bar. The main content area displays the profile of the user 'rofrano' under the organization 'NYU Devops'. The user has 18 repositories, 1 person, 0 teams, and 0 projects. A red box highlights the 'Settings' link in the user dropdown menu. A red arrow points from the text 'Select Settings' to the 'Settings' button in the dropdown menu. The dropdown also includes links for 'Your profile', 'Your repositories', 'Your stars', and 'Your gists'. The bottom right corner of the screen shows the GitHub logo.

NYU Devops

DevOps course at NYU Courant Institute of Mathematical Sciences

New York

Repositories 18

People 1

Teams 0

Projects 0

Settings

Find a repository...

Type: All

Select Settings

Customize pinned

Settings

Sign out

Top languages

Python Shell

Most used topics

flask python vagrant

Display a menu for "https://github.com/settings/profile"

New SSH key

The screenshot shows the GitHub 'Personal settings' page. On the left, a sidebar lists options: Personal settings (selected), Profile, Account, Emails, Notifications, Billing, **SSH and GPG keys**, Security, Sessions, Blocked users, Repositories, and Organizations. A red box highlights 'SSH and GPG keys', and a red arrow points from the text 'Select SSH and GPG keys' to this box. The main content area is titled 'SSH keys' and contains a list of associated keys. One key is shown: 'John's MacBook Pro' (key icon), 1b:1c:4f:4d:2a:e8:b7:87:c9:f2:11:61:d0:a2:dc:e5, Added on Jul 27, 2013, Last used within the last 2 weeks — Read/write. A 'Delete' button is next to the key entry. A green 'New SSH key' button is located at the top right of the 'SSH keys' section. Below it, another section titled 'GPG keys' states 'There are no GPG keys associated with your account.' with a 'New GPG key' button.

SSH and GPG keys

Select SSH and GPG keys

SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

John's MacBook Pro
1b:1c:4f:4d:2a:e8:b7:87:c9:f2:11:61:d0:a2:dc:e5
Added on Jul 27, 2013
Last used within the last 2 weeks — Read/write

Delete

New SSH key

GPG keys

New GPG key

New SSH key

The screenshot shows the GitHub 'Personal settings' page. On the left, a sidebar lists options: Personal settings (selected), Profile, Account, Emails, Notifications, Billing, SSH and GPG keys (highlighted with a red box), Security, Sessions, Blocked users, Repositories, and Organizations. The main content area is titled 'SSH keys' and contains the message: 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' Below this, there is a card for 'John's MacBook Pro' with the key fingerprint '1b:1c:4f:4d:2a:e8:b7:87:c9:f2:11:61:d0:a2:dc:e5', added on Jul 27, 2013, and last used within the last 2 weeks — Read/write. A 'Delete' button is on the right. To the right of the SSH keys section is a 'GPG keys' section with the message: 'There are no GPG keys associated with your account. Learn how to generate a GPG key and add it to your account.' A 'New GPG key' button is at the bottom right of this section. At the top right of the main content area is a green 'New SSH key' button. Red boxes and arrows highlight the 'SSH and GPG keys' link in the sidebar, the 'Select SSH and GPG keys' link above the GPG keys section, and the 'New SSH key' button.

SSH and GPG keys

Select SSH and GPG keys

Select New SSH key

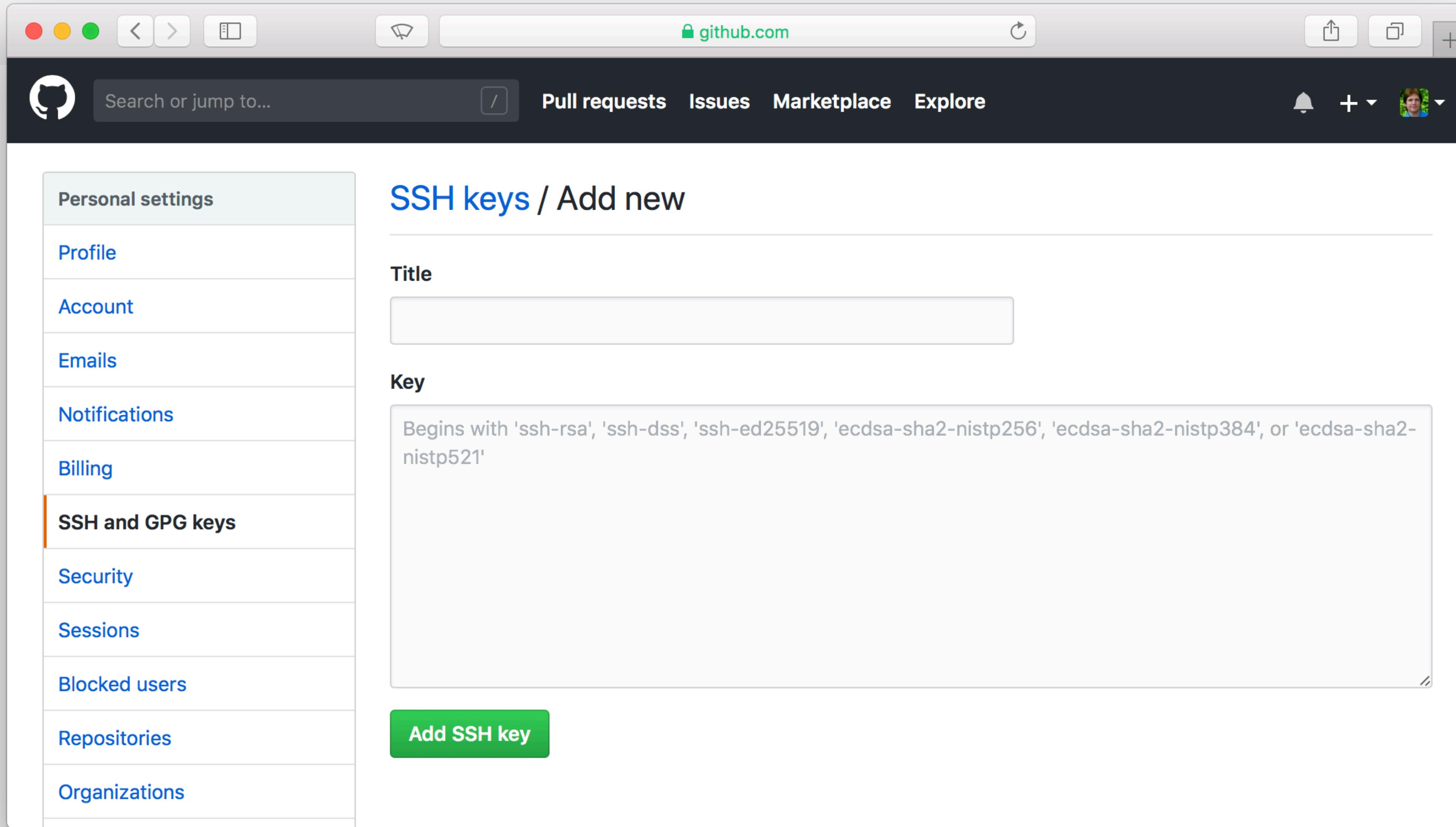
New SSH key

Copy Key to Clipboard

- Copy the key to your clipboard to paste into GitHub

```
$ pbcopy < ~/.ssh/id_rsa.pub
```

Add SSH key



The screenshot shows a Mac OS X browser window displaying the GitHub 'Add SSH key' page. The URL in the address bar is `github.com`. The GitHub logo is in the top-left corner of the header. The header also includes a search bar, a pull requests button, an issues button, a marketplace button, an explore button, a notifications bell icon, a plus sign for creating new items, and a user profile picture.

The left sidebar, titled 'Personal settings', contains the following options: Profile, Account, Emails, Notifications, Billing, **SSH and GPG keys**, Security, Sessions, Blocked users, Repositories, and Organizations. The 'SSH and GPG keys' option is currently selected, indicated by an orange vertical bar on its left.

The main content area is titled 'SSH keys / Add new'. It features two input fields: 'Title' and 'Key'. A note below the 'Key' field states: 'Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521''. At the bottom of the form is a green 'Add SSH key' button.

Add SSH key

The screenshot shows the GitHub 'Personal settings' page with the 'SSH and GPG keys' section selected. The main heading is 'SSH keys / Add new'. A red box highlights the 'Name it' input field, and a red arrow points down to the 'Title' input field, which is also highlighted by a red box. The 'Key' text area contains placeholder text: 'Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521''. At the bottom is a green 'Add SSH key' button.

SSH keys / Add new

Name it

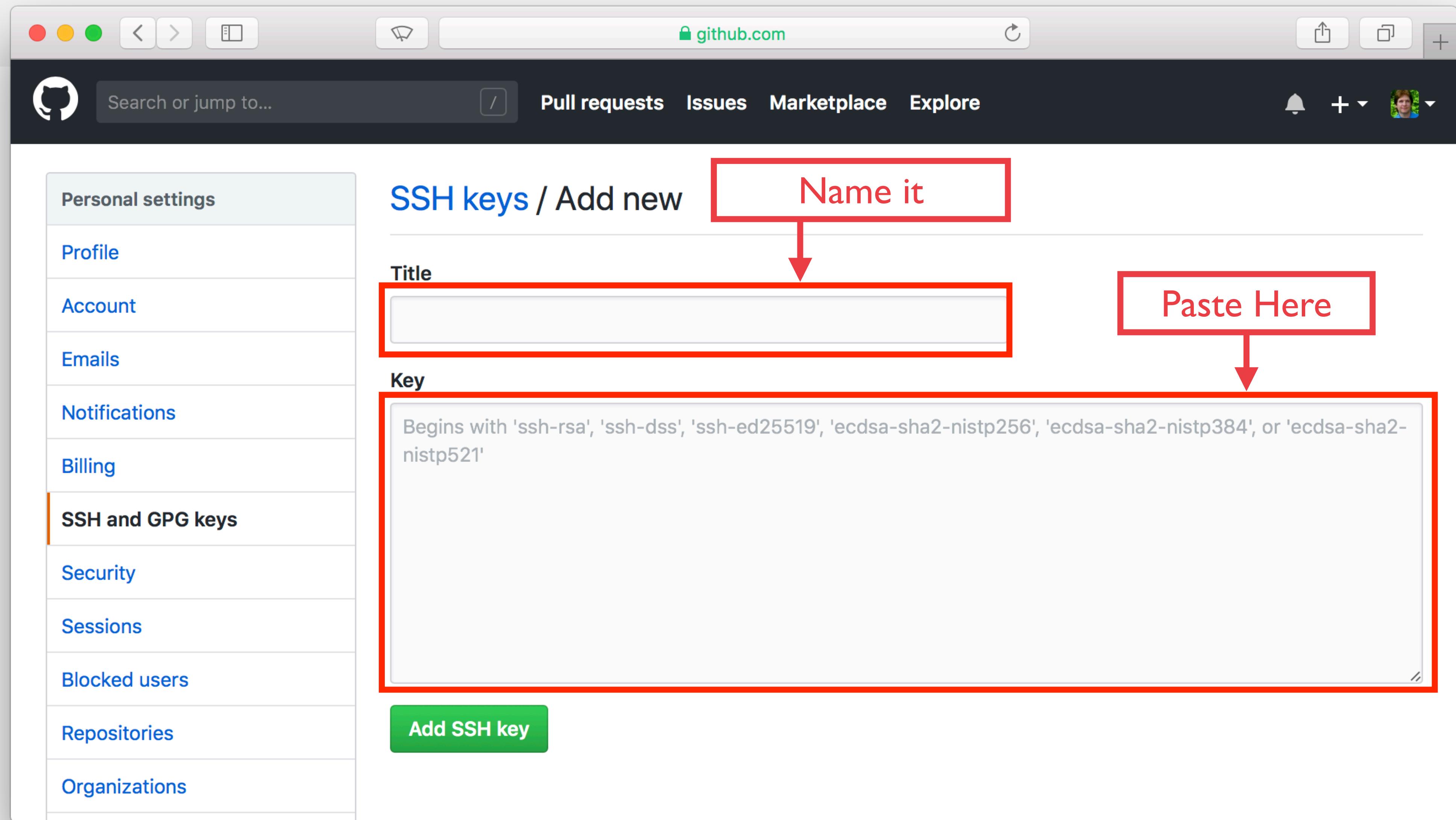
Title

Key

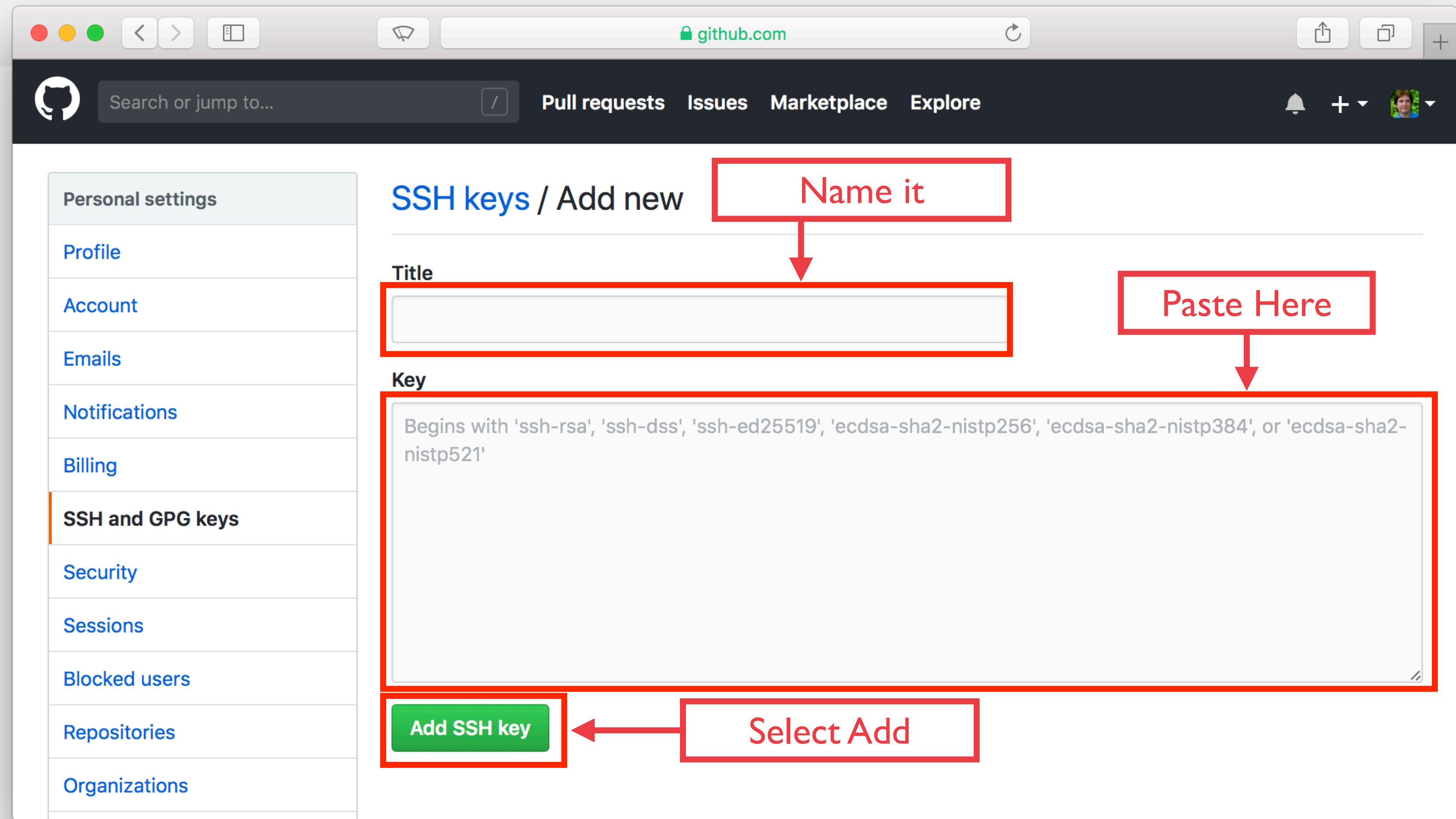
Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

Add SSH key



Add SSH key

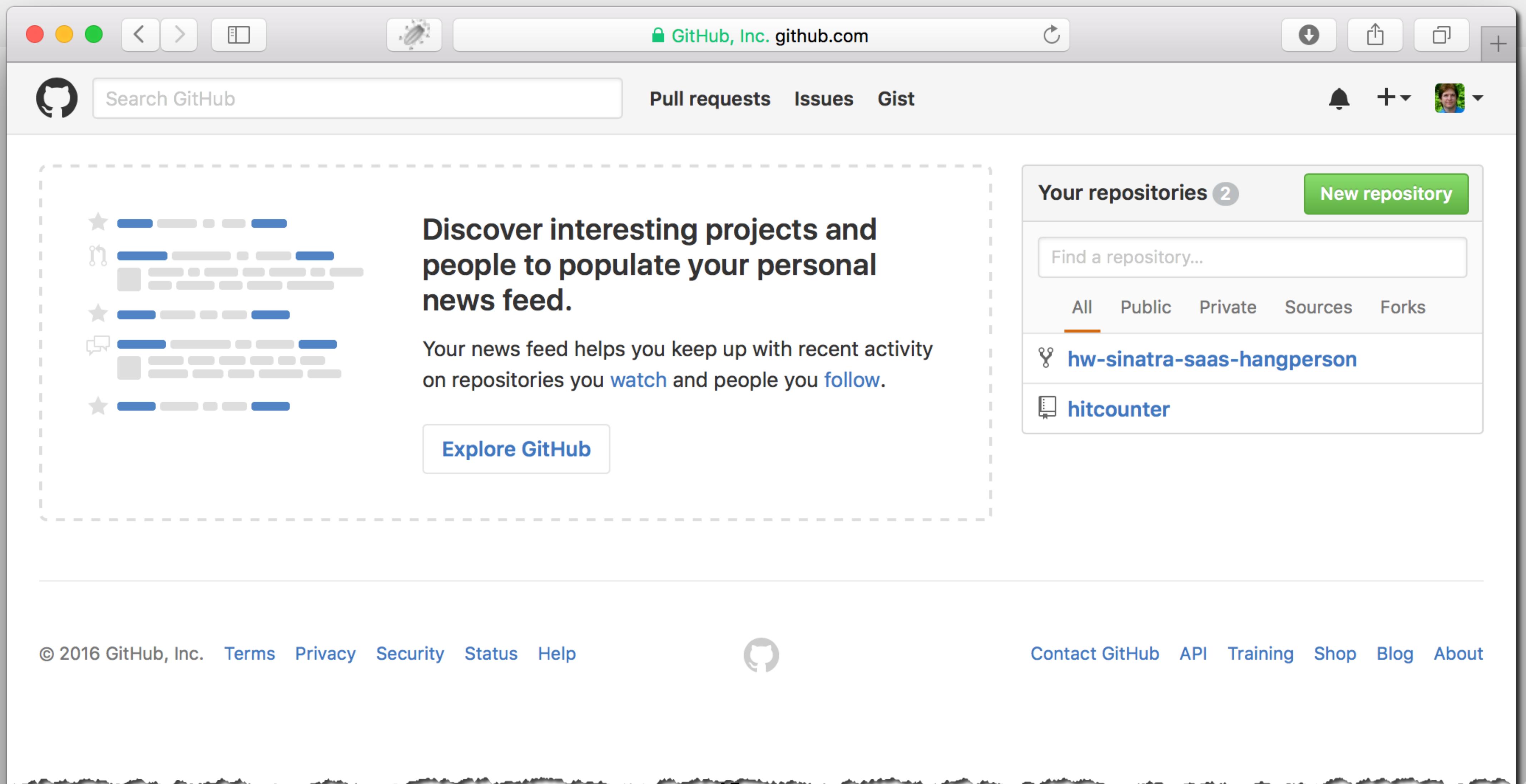


CREATE and FORK

- The Git Repository holds your code, documentation, etc.
- You need to CREATE one at github.com for new work or...
- Find one you want to contribute to and FORK it
- You only need to FORK a repo if you plan to contribute back
 - Otherwise you can just CLONE it



Create a New Repository



The screenshot shows the GitHub homepage with a focus on creating a new repository. At the top right, there is a green "New repository" button. The main content area features a section titled "Discover interesting projects and people to populate your personal news feed." Below this, there is a "Explore GitHub" button. On the right side, under "Your repositories", two repositories are listed: "hw-sinatra-saas-hangperson" and "hitcounter". The footer contains links for Terms, Privacy, Security, Status, Help, Contact GitHub, API, Training, Shop, Blog, and About.

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you [watch](#) and people you [follow](#).

[Explore GitHub](#)

Your repositories 2

[New repository](#)

Find a repository...

All Public Private Sources Forks

[hw-sinatra-saas-hangperson](#)

[hitcounter](#)

© 2016 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)

[Contact GitHub](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)

Create a New Repository

The screenshot shows the GitHub homepage. On the left, there's a section titled "Discover interesting projects and people to populate your personal news feed." It features five small cards with icons: a star, a person, a gear, a speech bubble, and another star. Below this is a "Explore GitHub" button. On the right, there's a sidebar with "Your repositories 2" and a "New repository" button, which is highlighted with a red box and an arrow pointing to it. A red callout box also points to this button with the text "Create a New Repository". The GitHub logo is at the bottom center.

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with changes on repositories you [watch](#) and people you [follow](#).

[Explore GitHub](#)

Your repositories 2

New repository

Find a repository...

All Public Private Sources Forks

hitcounter

© 2016 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)

Contact GitHub API Training Shop Blog About

New Repository Page

The screenshot shows the GitHub interface for creating a new repository. At the top, the URL is github.com/organizations/nyu-devops/repositories. The main title is "Create a new repository". Below it, a sub-instruction says "A repository contains all the files for your project, including the revision history." The "Owner" field is set to "nyu-devops". The "Repository name" field is empty and highlighted with a blue border. A suggestion "supreme-octo-funicular" is shown below. The "Description (optional)" field is empty. Under "Visibility", "Public" is selected, with the note "Anyone can see this repository. You choose who can commit.". The "Private" option is also available. There is a checkbox for "Initialize this repository with a README", which is unchecked. Below this, there are dropdown menus for ".gitignore" (set to "None") and "Add a license" (set to "None"). At the bottom is a large green "Create repository" button.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner /

Great repository names are short and memorable. Need inspiration? How about [supreme-octo-funicular](#).

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

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

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: [None](#) | Add a license: [None](#)

Create repository

New Repository Filled In

The screenshot shows the GitHub interface for creating a new repository. The URL in the address bar is github.com/organizations/nyu-devops/repositories. The page title is "Create a new repository".

Owner: nyu-devops

Repository name: lab-git-intro

Description (optional): This repo is for the lab in NYU class Intro to git

Visibility: Public (selected) / Private

Initialize this repository with a README: checked

Additional settings: Add .gitignore: Python, Add a license: Apache License 2.0

Create repository button at the bottom.

New Repository Filled In

A screenshot of a web browser showing the GitHub 'Create a new repository' page. The URL in the address bar is `GitHub, Inc. github.com/organizations/nyu-devops/repositories`. The page title is 'Create a new repository'. A red callout box with the text 'Name it: lab-git-intro' has an arrow pointing to the 'Repository name' input field, which contains the value 'lab-git-intro'. The 'Owner' dropdown is set to 'nyu-devops'. The 'Description (optional)' text area contains the placeholder 'This repo is for the lab in NYU class Intro to git'. Below the description are two radio button options: 'Public' (selected) and 'Private'. Underneath is a checked checkbox for 'Initialize this repository with a README'. At the bottom are buttons for 'Add .gitignore: Python' and 'Add a license: Apache License 2.0', followed by a large green 'Create repository' button.

Create a new repository

A repository contains all the files for your project, including

Owner nyu-devops Repository name lab-git-intro ✓

Great repository names are short and memorable. Need inspiration? How about [probable-rotary-phone](#).

Description (optional)

This repo is for the lab in NYU class Intro to git

Public
Anyone can see this repository. You choose who can commit.

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

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Python Add a license: Apache License 2.0

Create repository

Name it: lab-git-intro

New Repository Filled In

The screenshot shows the GitHub interface for creating a new repository. The URL in the address bar is `GitHub, Inc. github.com/organizations/nyu-devops/repositories`. The main title is "Create a new repository". Below it, a sub-instruction says "A repository contains all the files for your project, including".

Name it: lab-git-intro

The "Repository name" field contains "lab-git-intro" and has a red box around it. A red arrow points from this box to the text "Name it: lab-git-intro".

Give it a Description

The "Description (optional)" field contains "This repo is for the lab in NYU class Intro to git" and has a red box around it. A red arrow points from this box to the text "Give it a Description".

Owner: nyu-devops

Repository name: lab-git-intro

Description (optional): This repo is for the lab in NYU class Intro to git

Public (selected): Anyone can see this repository. You choose who can commit.

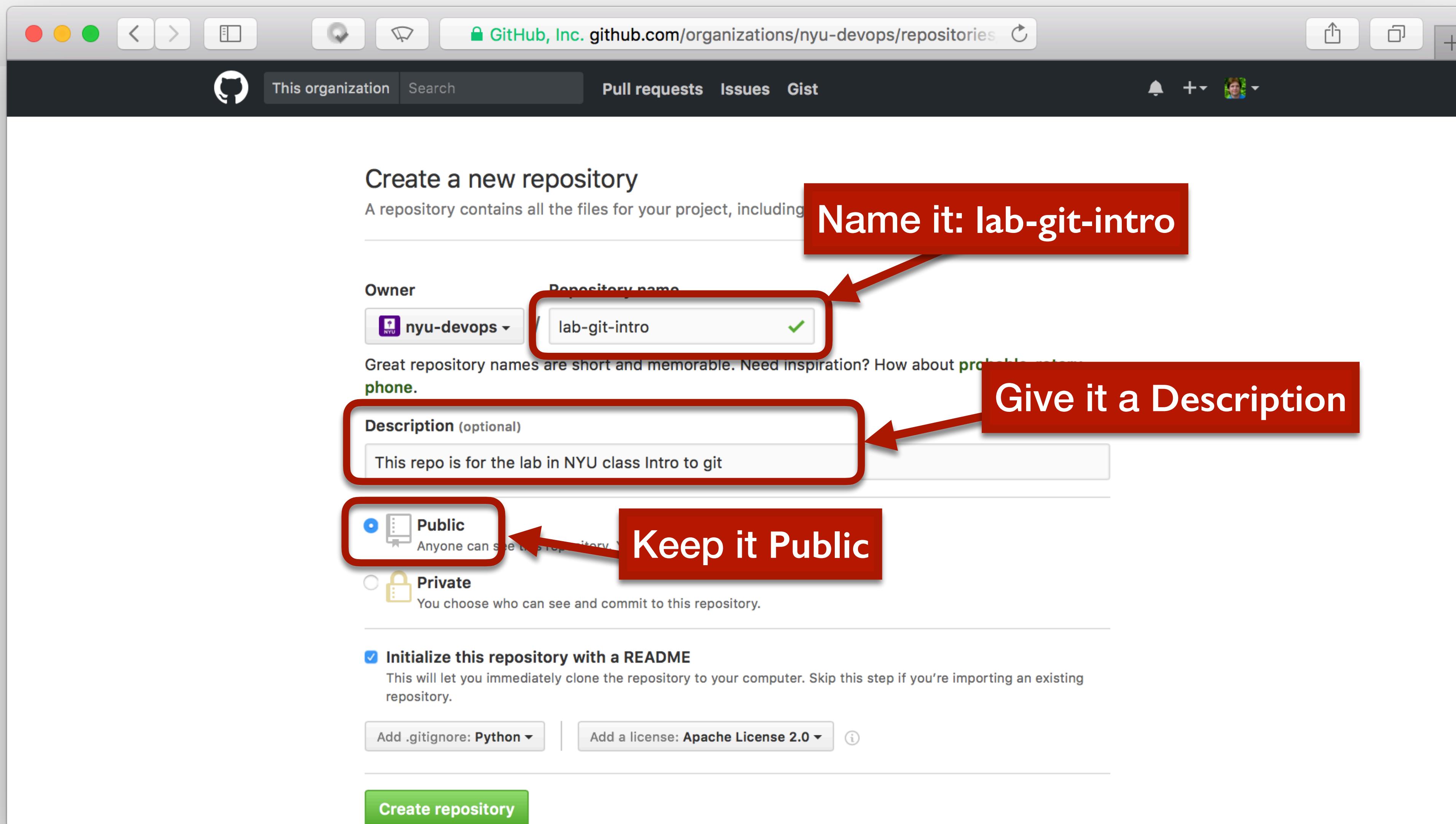
Private: You choose who can see and commit to this repository.

Initialize this repository with a README (checked): This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Python | Add a license: Apache License 2.0

Create repository

New Repository Filled In



New Repository Filled In

The screenshot shows the GitHub interface for creating a new repository. The URL in the address bar is `github.com/organizations/nyu-devops/repositories`. The page title is "Create a new repository". The "Repository name" field contains "lab-git-intro". The "Description (optional)" field contains "This repo is for the lab in NYU class Intro to git". The "Visibility" section shows "Public" selected (radio button is blue). The "Initialize this repository with a README" checkbox is checked. A red box labeled "Name it: lab-git-intro" points to the repository name field. A red box labeled "Give it a Description" points to the description field. A red box labeled "Keep it Public" points to the public visibility option. A red box labeled "Add a README, .gitignore, and License so that we have something to change" points to the "Initialize this repository with a README" checkbox.

Name it: lab-git-intro

Give it a Description

Keep it Public

Add a README, .gitignore, and License so that we have something to change

Create repository

New Repository Filled In

The screenshot shows the GitHub 'Create a new repository' interface. A red box highlights the 'Repository name' field containing 'lab-git-intro'. Another red box highlights the 'Description (optional)' field with the text 'This repo is for the lab in NYU class Intro to git'. A third red box highlights the 'Public' radio button. A fourth red box highlights the 'Initialize this repository with a README' checkbox. A fifth red box highlights the 'Create repository' button at the bottom.

Name it: lab-git-intro

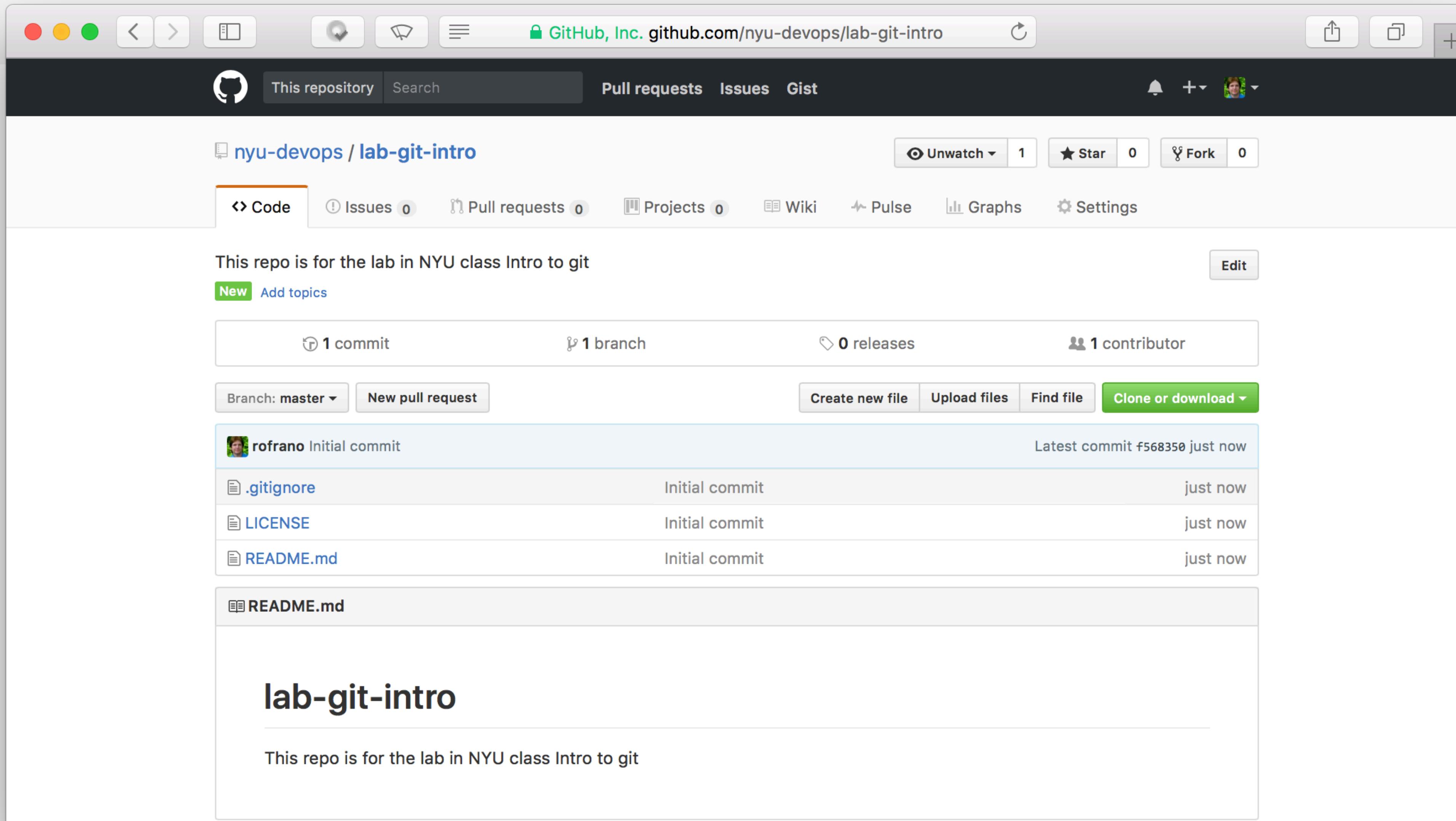
Give it a Description

Keep it Public

Add a README, .gitignore, and License so that we have something to change

Go for it! (Create repository)

You Now Have A Repository



Guidelines For Contributing

- To help your project contributors do good work, you can add a file with contribution guidelines to the root of your project's repository
- Then, whenever someone opens a pull request or creates an issue, they will see a link to that file.
- Good Social Coding requires that everyone know and follow the guidelines.



Consistent Contributions

- As you can imagine, with outside contributors there must be coding standards
- Use a **CONTRIBUTION.md** file to document how to contribute
- It's a good idea to publish your **coding standards** on your GitHub Wiki or other public place and point to them in your **CONTRIBUTION.md** file.
- When a Pull Request is made, you should also check that the standards have been followed



Clone Options

The screenshot shows a GitHub repository page for 'nyu-devops / lab-git-intro'. The browser window has a title bar with standard OS X icons and a URL bar showing 'github.com/nyu-devops/lab-git-intro/tree/master'. The main header includes the GitHub logo, 'This repository' button, search bar, and navigation links for 'Pull requests', 'Issues', and 'Gist'. On the right side of the header are 'Unwatch' (1), 'Star' (0), 'Fork' (0) buttons, and a user profile icon.

The repository name 'nyu-devops / lab-git-intro' is displayed prominently. Below it is a navigation bar with tabs: 'Code' (selected), 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. A note below the repository name states 'This repo is for the lab in NYU class Intro to git' with an 'Edit' button.

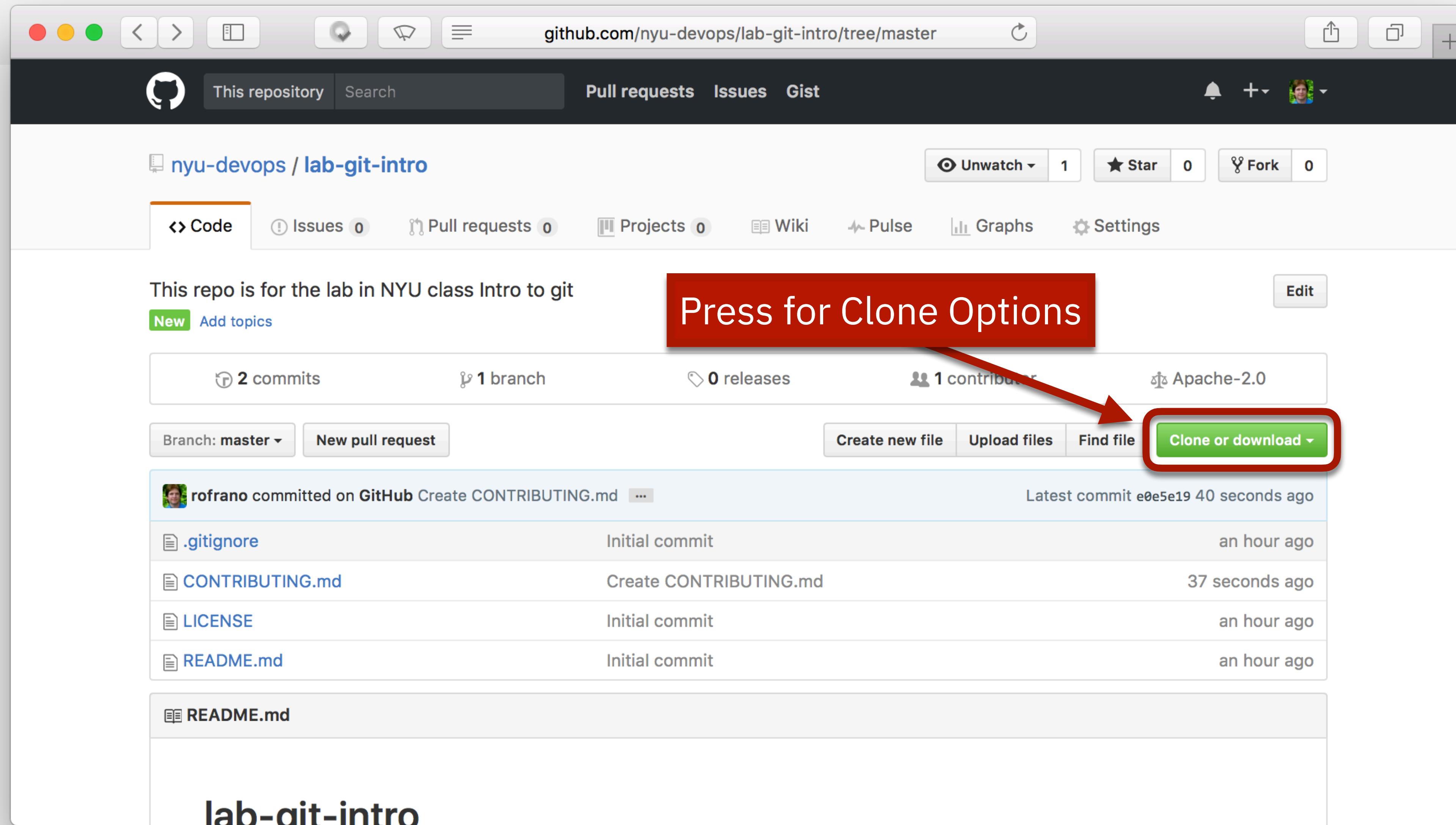
Key statistics are shown: 2 commits, 1 branch, 0 releases, 1 contributor, and Apache-2.0 license. Below these are buttons for 'Branch: master ▾', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a large green 'Clone or download ▾' button.

The commit history lists the following entries:

- rofrano committed on GitHub Create CONTRIBUTING.md ... Latest commit e0e5e19 40 seconds ago
- .gitignore Initial commit an hour ago
- CONTRIBUTING.md Create CONTRIBUTING.md 37 seconds ago
- LICENSE Initial commit an hour ago
- README.md Initial commit an hour ago

A large 'lab-ait-intro' watermark is visible at the bottom of the page.

Clone Options



Clone Your Repo

This repo is for the lab in NYU class Intro to git

New Add topics

2 commits 1 branch 0 releases 1 contributor Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

rofrano committed on GitHub Create CONTRIBUTING.md ...

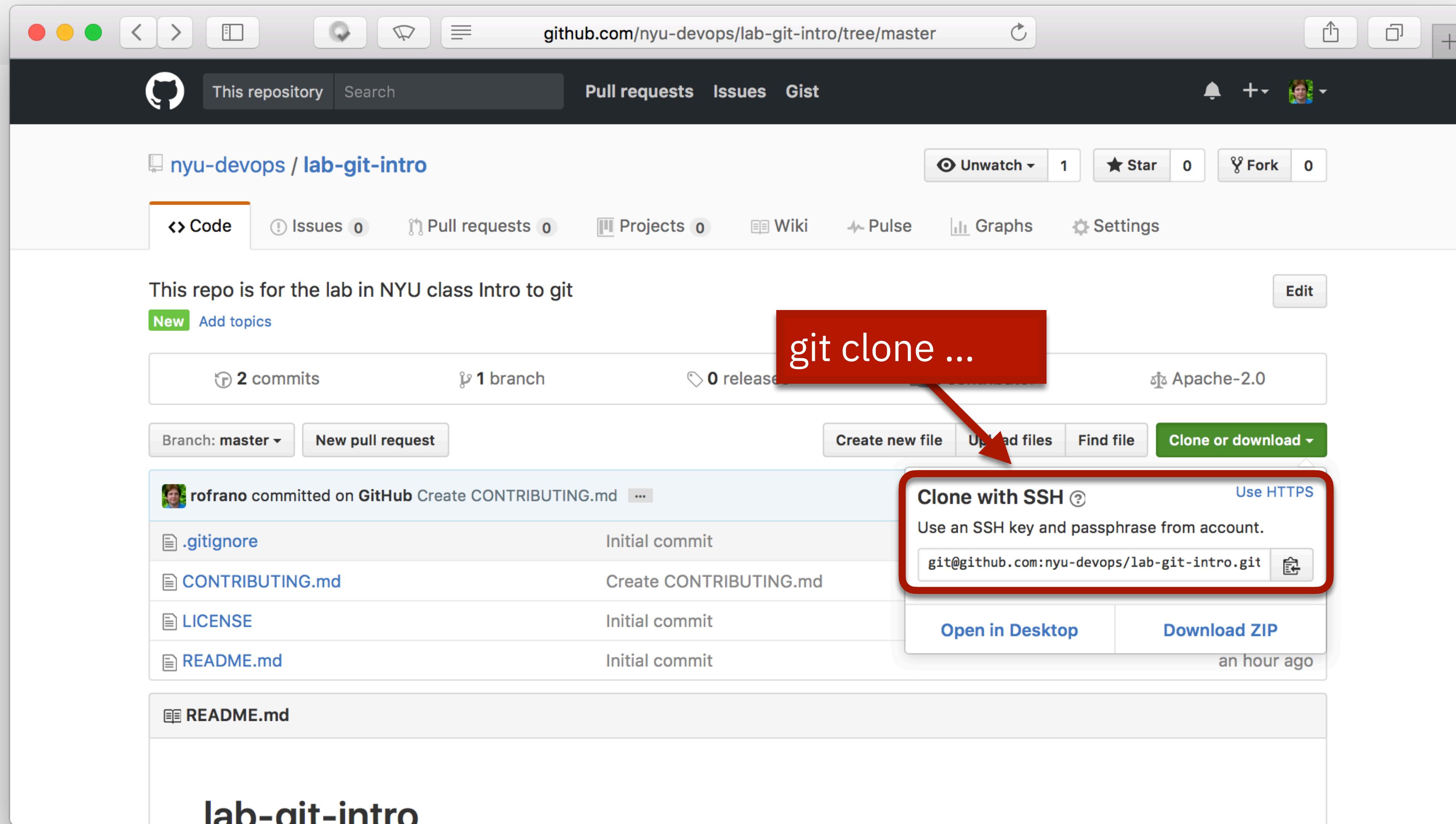
File	Commit
.gitignore	Initial commit
CONTRIBUTING.md	Create CONTRIBUTING.md
LICENSE	Initial commit
README.md	Initial commit

lab-git-intro

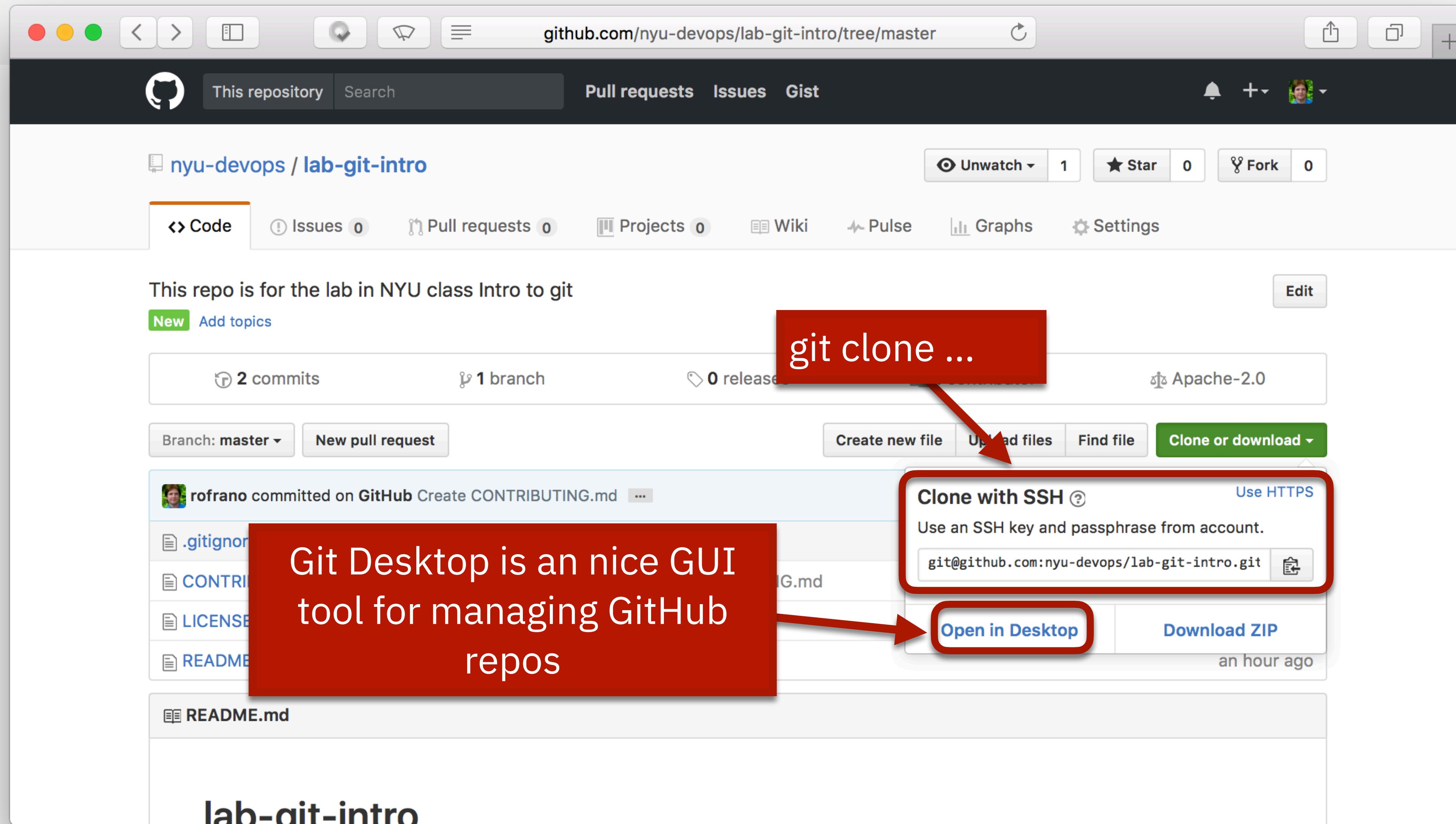
Clone with SSH Use HTTPS
git@github.com:nyu-devops/lab-git-intro.git

Open in Desktop Download ZIP an hour ago

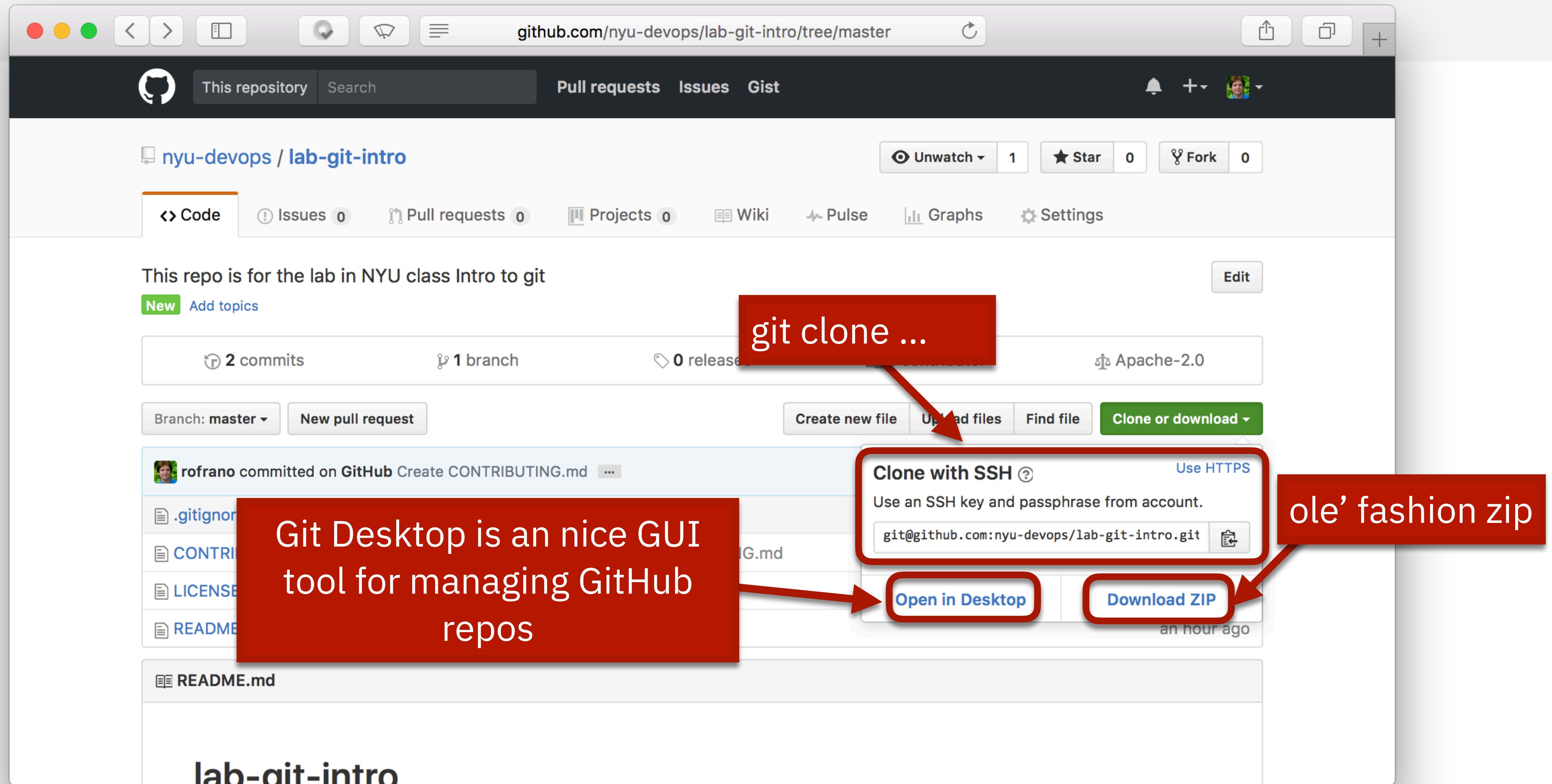
Clone Your Repo



Clone Your Repo



Clone Your Repo



CLONE

- Let's CLONE the repository to get a local copy

```
$ git clone git@github.com:rofrano/nyu-lab-git-intro.git
```

\

CLONE

- Cloning copies the entire repository locally

```
$ git clone git@github.com:rofrano/nyu-lab-git-intro.git
Cloning into 'nyu-lab-git-intro'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 4.79 KiB | 0 bytes/s, done.
Checking connectivity... done.
$
```

CLONE

- Change into the cloned directory

```
$ git clone git@github.com:rofrano/nyu-lab-git-intro.git
Cloning into 'nyu-lab-git-intro'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 4.79 KiB | 0 bytes/s, done.
Checking connectivity... done.
$ cd nyu-lab-git-intro/
$
```

CLONE

- List the files to see what was cloned

```
$ git clone git@github.com:rofrano/nyu-lab-git-intro.git
Cloning into 'nyu-lab-git-intro'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 4.79 KiB | 0 bytes/s, done.
Checking connectivity... done.
$ cd nyu-lab-git-intro/
$ ls -la
total 40
drwxr-xr-x  6 rofrano  staff   204B Sep 12 15:46 .
drwxr-xr-x  3 rofrano  staff   102B Sep 12 15:46 ../
drwxr-xr-x 13 rofrano  staff  442B Sep 12 15:46 .git/
-rw-r--r--  1 rofrano  staff   1.0K Sep 12 15:46 .gitignore
-rw-r--r--  1 rofrano  staff   11K Sep 12 15:46 LICENSE
-rw-r--r--  1 rofrano  staff   71B Sep 12 15:46 README.md
$
```

CLONE

- List the files to see what was cloned

```
$ git clone git@github.com:rofrano/nyu-lab-git-intro.git
Cloning into 'nyu-lab-git-intro'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 4.79 KiB | 0 bytes/s, done.
Checking connectivity... done.
$ cd nyu-lab-git-intro/
$ ls -la
total 40
drwxr-xr-x  6 rofrano  staff   204B Sep 12 15:46 .
drwxr-xr-x  3 rofrano  staff   102B Sep 12 15:46 ../
drwxr-xr-x 13 rofrano  staff   442B Sep 12 15:46 .git/
-rw-r--r--  1 rofrano  staff   1.0K Sep 12 15:46 .gitignore
-rw-r--r--  1 rofrano  staff   11K Sep 12 15:46 LICENSE
-rw-r--r--  1 rofrano  staff   71B Sep 12 15:46 README.md
$
```

This is the local repository

CLONE

- List the files to see what was cloned

```
$ git clone git@github.com:rofrano/nyu-lab-git-intro.git
Cloning into 'nyu-lab-git-intro'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 4.79 KiB | 0 bytes/s, done.
Checking connectivity... done.
$ cd nyu-lab-git-intro/
$ ls -la
total 40
drwxr-xr-x  6 rofrano  staff   204B Sep 12 15:46 .
drwxr-xr-x  3 rofrano  staff   102B Sep 12 15:46 ../
drwxr-xr-x 13 rofrano  staff  442B Sep 12 15:46 .git/
-rw-r--r--  1 rofrano  staff   1.0K Sep 12 15:46 .gitignore
-rw-r--r--  1 rofrano  staff   11K Sep 12 15:46 LICENSE
-rw-r--r--  1 rofrano  staff   71B Sep 12 15:46 README.md
$
```

This is the local repository

.git/
.gitignore
LICENSE
README.md

These are the optional files
we asked to have created

GIT STATUS is Your Friend

- Git Status will tell you the status of your work
- You'll be using it a lot!

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
$
```

Issues

- ISSUES help you track what everyone is doing
- ISSUES can be features or defects or pull requests or ...
- You should not be working on code without an open ISSUE
- Always assign the ISSUE to yourself before working on it



We Need an Issue to Work On

The screenshot shows a GitHub repository page for 'nyu-lab-git-intro'. The repository was created by 'rofrano' and has 1 commit, 1 branch, 0 releases, 1 contributor, and is licensed under Apache-2.0. The latest commit was made 6 hours ago. The repository contains files: .gitignore, LICENSE, and README.md. The README.md file is displayed with the title 'nyu-lab-git-intro'.

This repo is for the lab in NYU class Intro to git — Edit

1 commit 1 branch 0 releases 1 contributor Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

rofrano Initial commit Latest commit 907cda6 6 hours ago

.gitignore Initial commit 6 hours ago

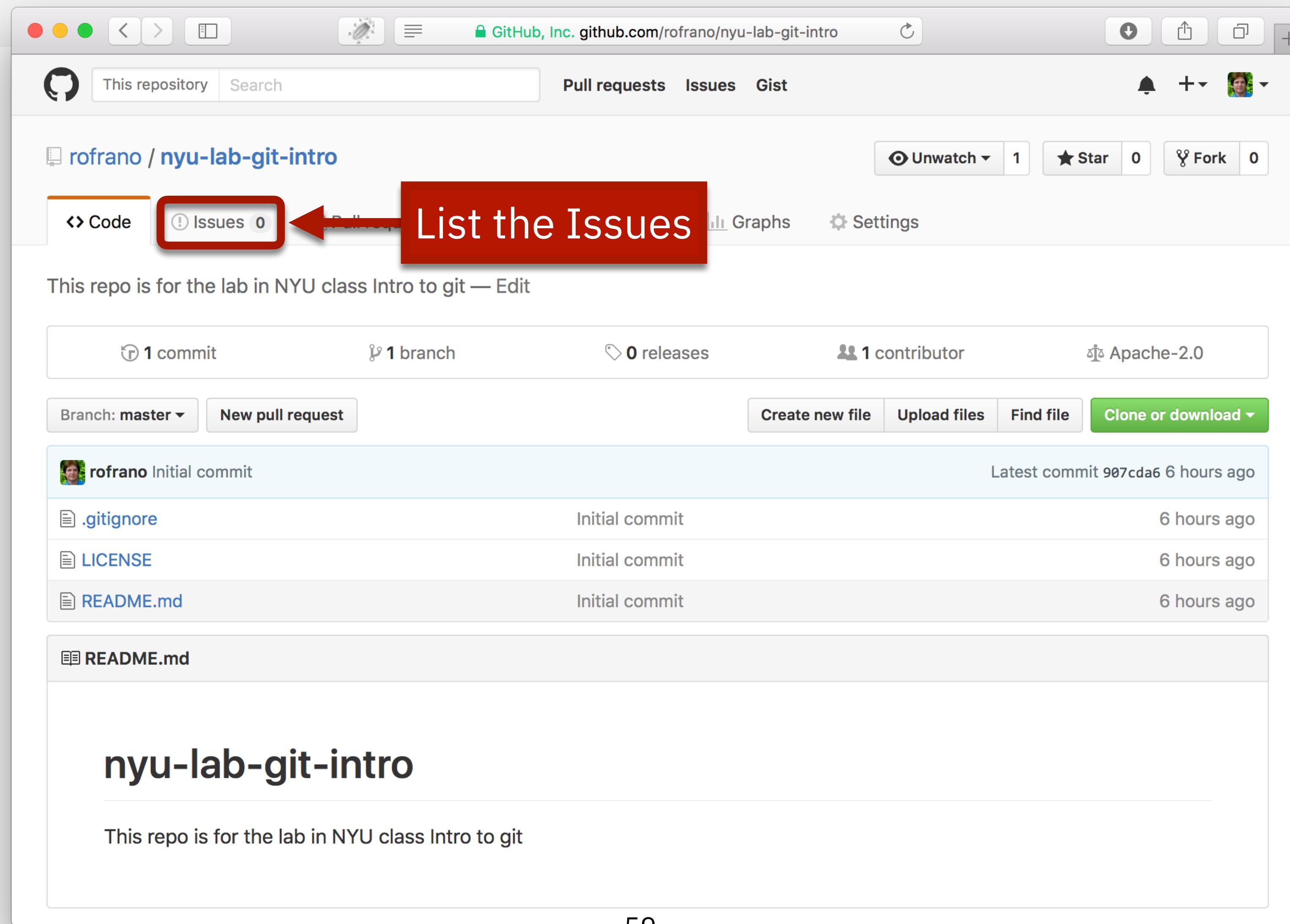
LICENSE Initial commit 6 hours ago

README.md Initial commit 6 hours ago

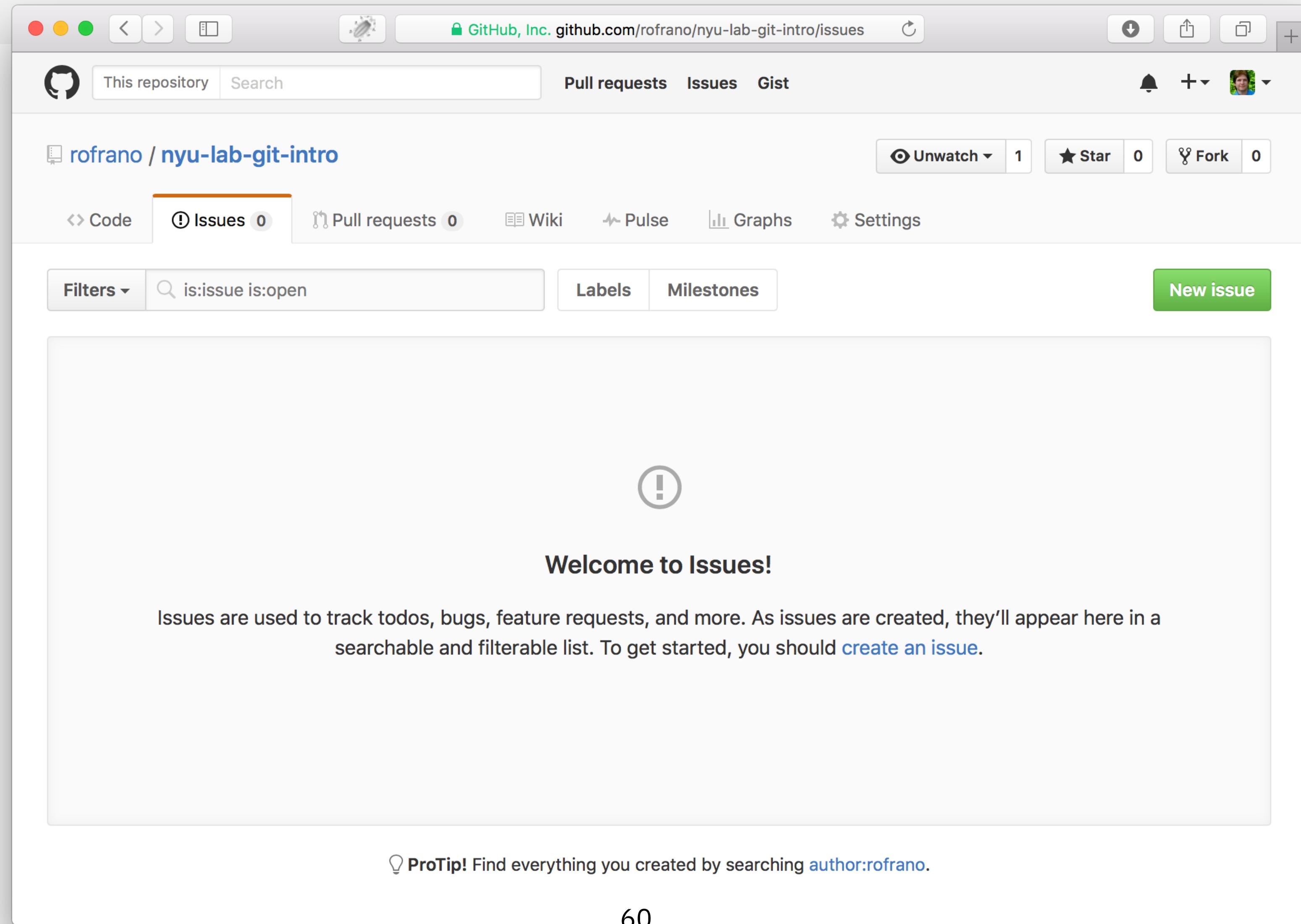
nyu-lab-git-intro

This repo is for the lab in NYU class Intro to git

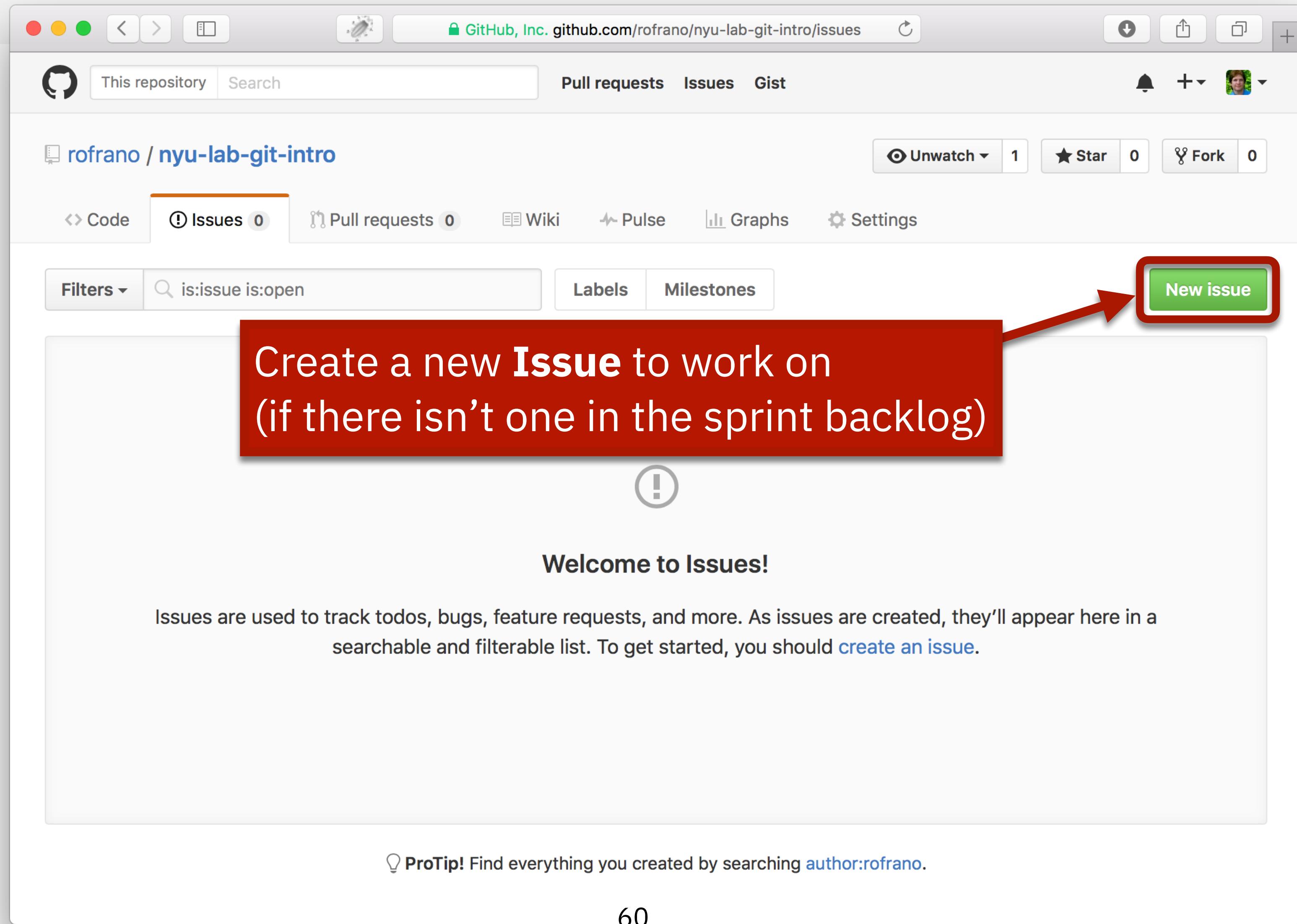
We Need an Issue to Work On



Create a New Issue



Create a New Issue



Create a new **Issue** to work on
(if there isn't one in the sprint backlog)

New issue

Welcome to Issues!

Issues are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list. To get started, you should [create an issue](#).

ProTip! Find everything you created by searching [author:rofrano](#).

New Issue Page

The screenshot shows the GitHub New Issue Page for the repository `rofrano/nyu-lab-git-intro`. The page has a clean, modern design with a light gray background. At the top, there's a header bar with the GitHub logo, a search bar, and navigation links for "Pull requests", "Issues", and "Gist". Below the header, the repository name `rofrano / nyu-lab-git-intro` is displayed, along with statistics: 1 unwatched, 0 stars, and 0 forks. A navigation bar below the repository name includes links for "Code", "Issues 0", "Pull requests 0", "Wiki", "Pulse", "Graphs", and "Settings". The main content area is a large form for creating a new issue. It features a title input field, a rich text editor with "Write" and "Preview" tabs, and a WYSIWYG toolbar. Below the editor is a text area for "Leave a comment" and a file attachment section with the instruction "Attach files by dragging & dropping or selecting them.". A note at the bottom left says "Styling with Markdown is supported". On the right side of the form, there are three sections: "Labels" (None yet), "Milestone" (No milestone), and "Assignees" (No one—assign yourself). At the bottom right of the form is a green "Submit new issue" button. The footer of the page includes standard GitHub links for "Contact GitHub", "API", "Training", "Shop", "Blog", and "About", along with copyright information for 2016 GitHub, Inc.

Markdown Reference

<https://guides.github.com/features/mastering-markdown/>

Type	Or	... to Get
Italic	_Italic_	<i>Italic</i>
Bold	__Bold__	Bold
# Heading 1	Heading 1 =====	<h1>Heading 1</h1>
## Heading 2	Heading 2 -----	<h2>Heading 2</h2>
[Link](http://a.com)	[Link]	Link
![Image](http://url/a.png)	![Image]	
> Blockquote		Blockquote
* List	- List	• List
* List	- List	• List
1. One	1) One	1. One
2. Two	2) Two	2. Two

Fill Out The Issue

The screenshot shows a GitHub repository named 'rofrano / nyu-lab-git-intro' with a new issue being created. The issue title is 'Add a home page for our web site'. The description contains a user story in Markdown:

```
**As a** Retailer
**I need** a home page for web site
**So that** my customers can see that we are open for business

**Assumptions:**
* The home page will be called `index.html`
* We will use `css` style sheets

**Acceptance Criteria:**
```
When I navigate to the base URL for our web site
I should see the home page
```

Attach files by dragging & dropping or selecting them.
```

A large red callout box on the right side of the screen contains the text: 'Give it a good **Title** and **Description** and then **Submit**'. Three red arrows point from the text in the callout box to the following elements: one arrow points to the title field, another points to the description area, and a third points to the green 'Submit new issue' button at the bottom right.

Submit new issue

Issues are Assigned Numbers

The screenshot shows a GitHub repository page for 'rofrano / nyu-lab-git-intro'. The 'Issues' tab is selected, showing one open issue. The issue title is 'Add a home page for our web site #1'. The issue was opened by 'rofrano' just now with 0 comments. A comment from 'rofrano' is displayed, detailing requirements, assumptions, and acceptance criteria for the task.

rofrano commented just now

As a Retailer
I need a home page for web site
So that my customers can see that we are open for business

Assumptions:

- The home page will be called `index.html`
- We will use `css` style sheets

Acceptance Criteria:

When I navigate to the base URL for our web site
I should see the home page

Labels: None yet

Milestone: No milestone

Assignees: No one—assign yourself

1 participant: rofrano

Notifications: You're receiving notifications because you authored the thread. [Unsubscribe](#)

Issues are Assigned Numbers

A screenshot of a GitHub issue page for the repository "rofrano / nyu-lab-git-intro". The page shows one open issue titled "Add a home page for our web site". The issue number "#1" is highlighted with a red square and an arrow points from it to a red callout box containing the text: "Remember this Issue number for later reference in commit messages".

The GitHub interface includes a navigation bar with "Pull requests", "Issues", and "Gist" tabs. Below the title, there are links for "Code", "Issues 1", "Pull requests 0", "Wiki", "Pulse", and "Graphs". On the right side, there are sections for "Labels" (None yet), "Milestone" (No milestone), "Assignees" (No one—assign yourself), and "1 participant" (a small profile picture). A "Notifications" section at the bottom right says "You're receiving notifications because you authored the thread." with a "Unsubscribe" button.

Add a home page for our web site #1

rofrano commented just now

As a Retailer
I need a home page for web site
So that my customers can see that we are open for business

Assumptions:

- The home page will be called `index.html`
- We will use `css` style sheets

Acceptance Criteria:

When I navigate to the base URL for our web site
I should see the home page

Labels: None yet

Milestone: No milestone

Assignees: No one—assign yourself

1 participant:

Notifications: **Unsubscribe**

You're receiving notifications because you authored the thread.

Assign Issue to Yourself

The screenshot shows a GitHub issue page for a repository named "rofrano / nyu-lab-git-intro". The issue is titled "Add a home page for our web site #1" and is currently open. The "Issues" tab is selected, showing 1 issue. The issue details show a comment from "rofrano" stating: "As a Retailer I need a home page for web site So that my customer can easily find us". Below this, there are sections for "Assumptions:" and "Acceptance Criteria". A large red callout box with white text is overlaid on the issue body, containing the instruction: "Assign it to yourself so that the team knows that you are working on it". An arrow points from this callout to the "Assignees" section on the right side of the page. The "Assignees" section shows "rofrano" listed as the assignee, with a note indicating "1 participant". The sidebar on the right also shows "None yet" for Labels and "No milestone" for Milestone.

Add a home page for our web site #1

Open rofrano opened this issue a minute ago · 0 comments

rofrano commented a minute ago

As a Retailer
I need a home page for web site
So that my customer can easily find us

Assumptions:

- The home page will be simple and clean.
- We will use a static site generator like Jekyll.

Acceptance Criteria

When I navigate to the base URL for our web site
I should see the home page

rofrano self-assigned this 14 seconds ago

Assignees

rofrano

1 participant

Notifications

Unsubscribe

You're receiving notifications because you were assigned.

GitHub Collaboration

The screenshot shows a GitHub issue page for a project titled "So that my customers can see that we are open for business". The issue has two sections: "Assumptions:" and "Acceptance Criteria:". The "Assumptions:" section lists:

- The home page will be called `index.html`
- We will use `css` style sheets

The "Acceptance Criteria:" section contains a single item:

When I navigate to the base URL for our web site
I should see the home page

A self-assigned note from user `rofrano` states: "rofrano self-assigned this 14 seconds ago".

On the right side of the page, there are "Labels" (None yet) and "Milestone" (No milestone) sections. A red callout box with white text is overlaid on the right side, containing the following message:

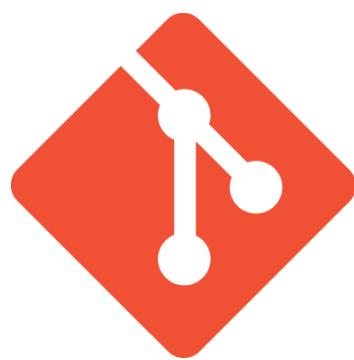
All of your collaboration should be done
on GitHub where others can follow the
conversation.
(i.e., STOP SENDING EMAILS!!!)

A red arrow points from the bottom of the callout box towards the "Comment" button at the bottom of the issue editor.

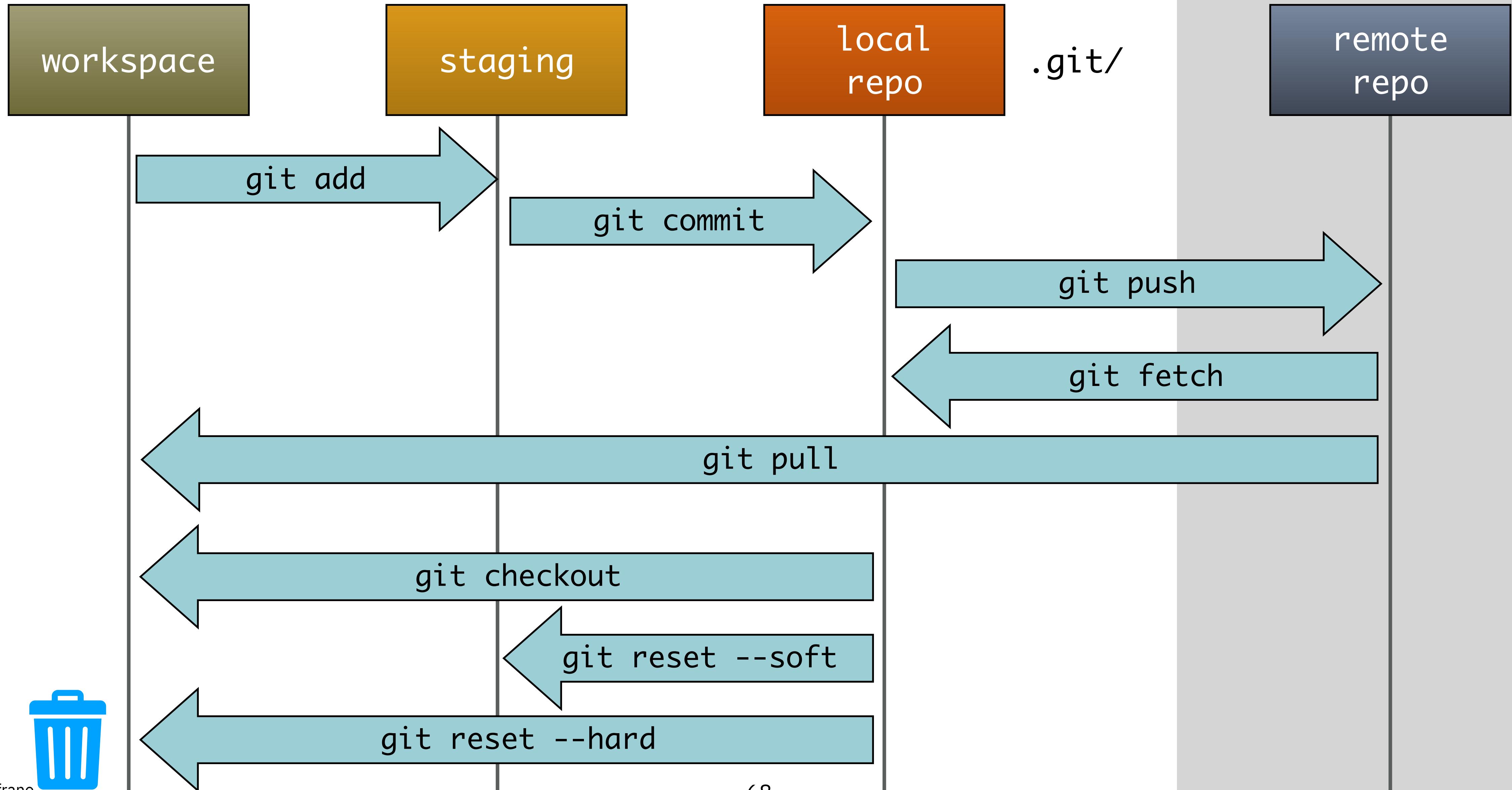
At the bottom of the page, there are links for "Write" and "Preview", a rich text editor toolbar, and a comment input field with placeholder text "Leave a comment". Below the input field is a note: "Attach files by dragging & dropping or selecting them.". At the bottom right are "Close issue" and "Comment" buttons. The footer includes copyright information: "© 2016 GitHub, Inc. Terms Privacy Security Status Help" and navigation links: "Contact GitHub API Training Shop Blog About".

IT'S TIME TO PLAY





Git Command Workflow



Branches

- Branching is a core concept in Git
 - The entire GitHub Flow is based upon branches
- There's only one rule:
 - *Anything in the **master** branch must always be deployable.*
- It's extremely important that your new branch is created off of master when working on a feature or a fix

BRANCH

- All development is done in a BRANCH

```
git checkout -b add-home-page
```

- You can switch between BRANCHES

```
git checkout other-branch
```

- You can see what branches exist (-a returns all both local and remote)

```
git branch -a
```

- You should delete the BRANCH after it has been MERGED

```
git branch -d add-home-page
```

Create A Branch

- Create a BRANCH to add the home page

```
$ git checkout -b add-home-page
Switched to a new branch 'add-home-page'
$
```

Add Some Files

- Add your working files and edit them as needed

```
$ git checkout -b add-home-page
Switched to a new branch 'add-home-page'
$ touch index.html
$ touch index.css
$
```

Check Your Status

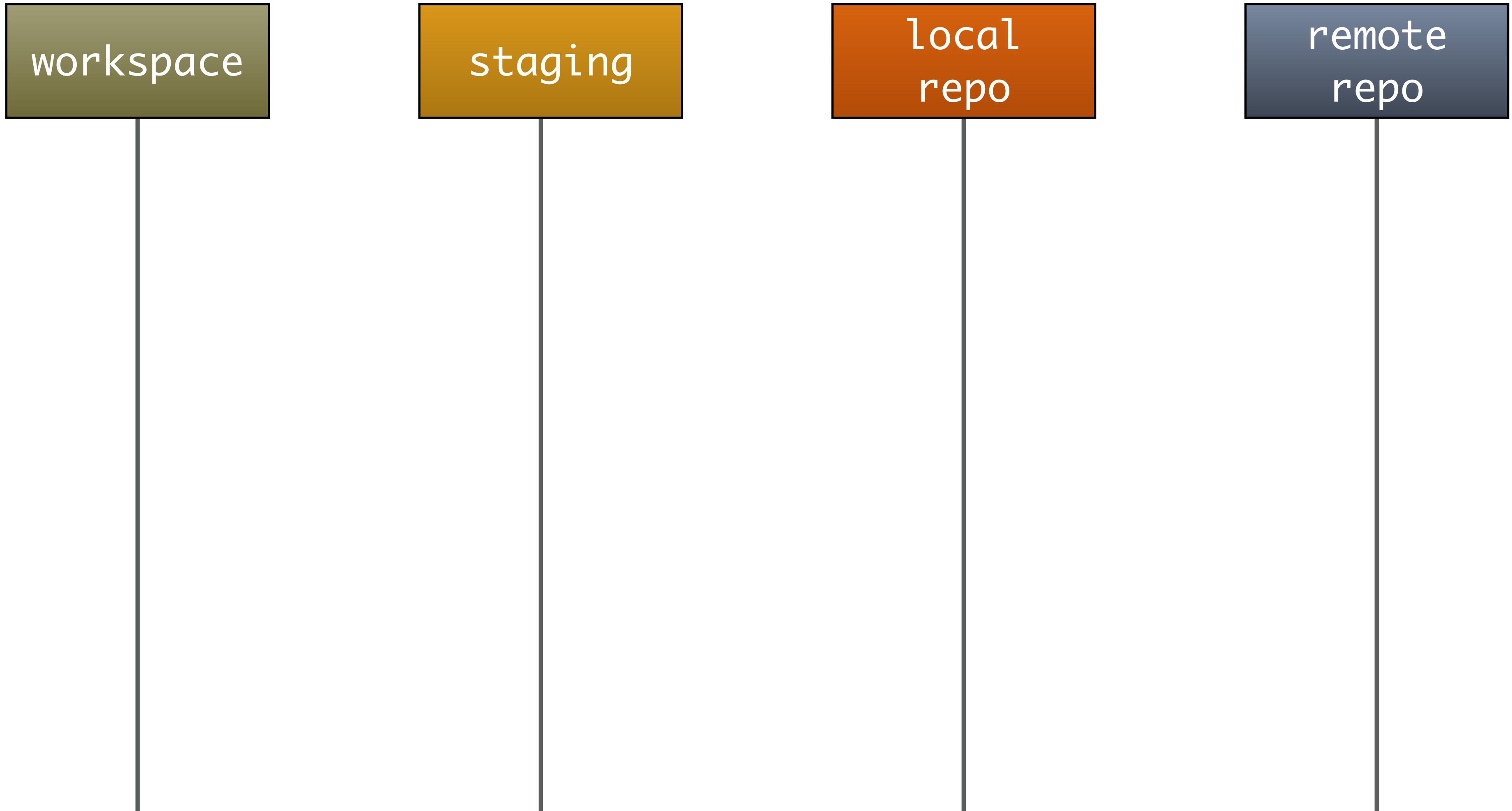
- Git Status shows that the files are not yet Tracked

```
$ git checkout -b add-home-page
Switched to a new branch 'add-home-page'
$ touch index.html
$ touch index.css
$ git status
On branch add-home-page
Untracked files:
  (use "git add <file>..." to include in what will be committed)

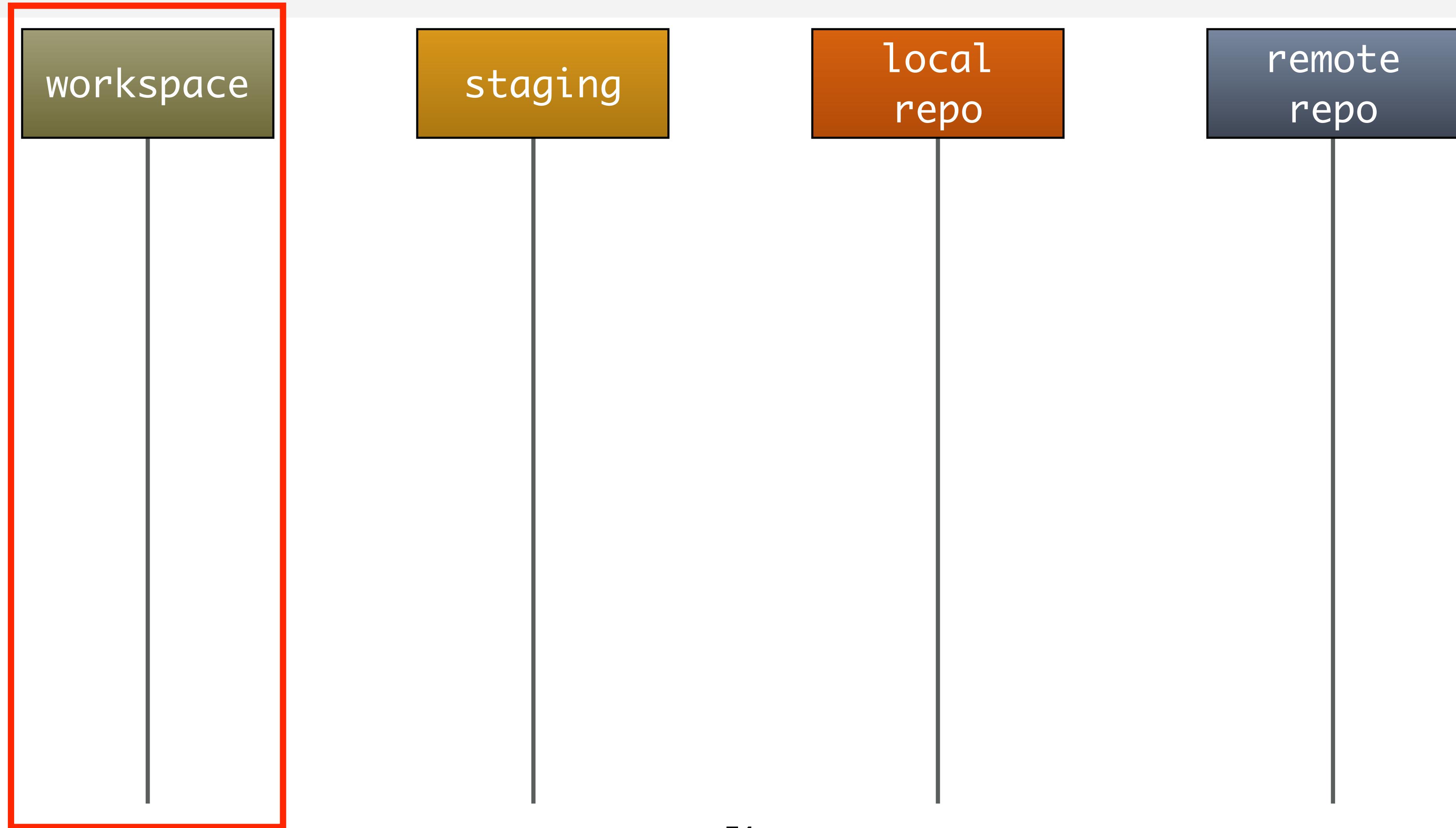
    index.css
    index.html

nothing added to commit but untracked files present (use "git add" to track)
$
```

Where Are Your Changes Now?



Where Are Your Changes Now?



ADD

- Use ADD to stage your files for a COMMIT
 - You can use individual filenames or wild cards
 - `git add .` ← will add everything! (use with extreme caution)

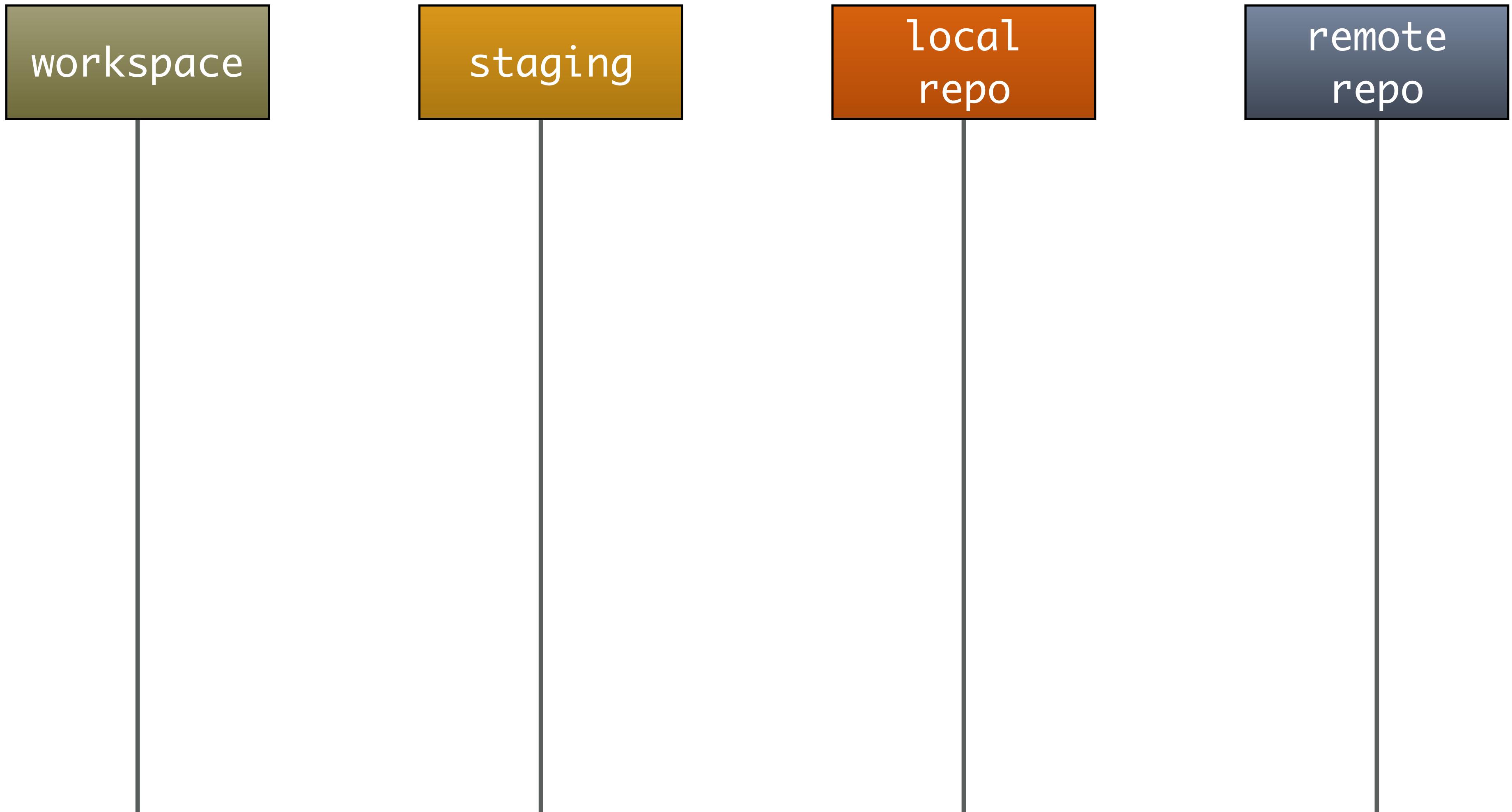
```
$ git add index.*  
$
```

Check Your Status

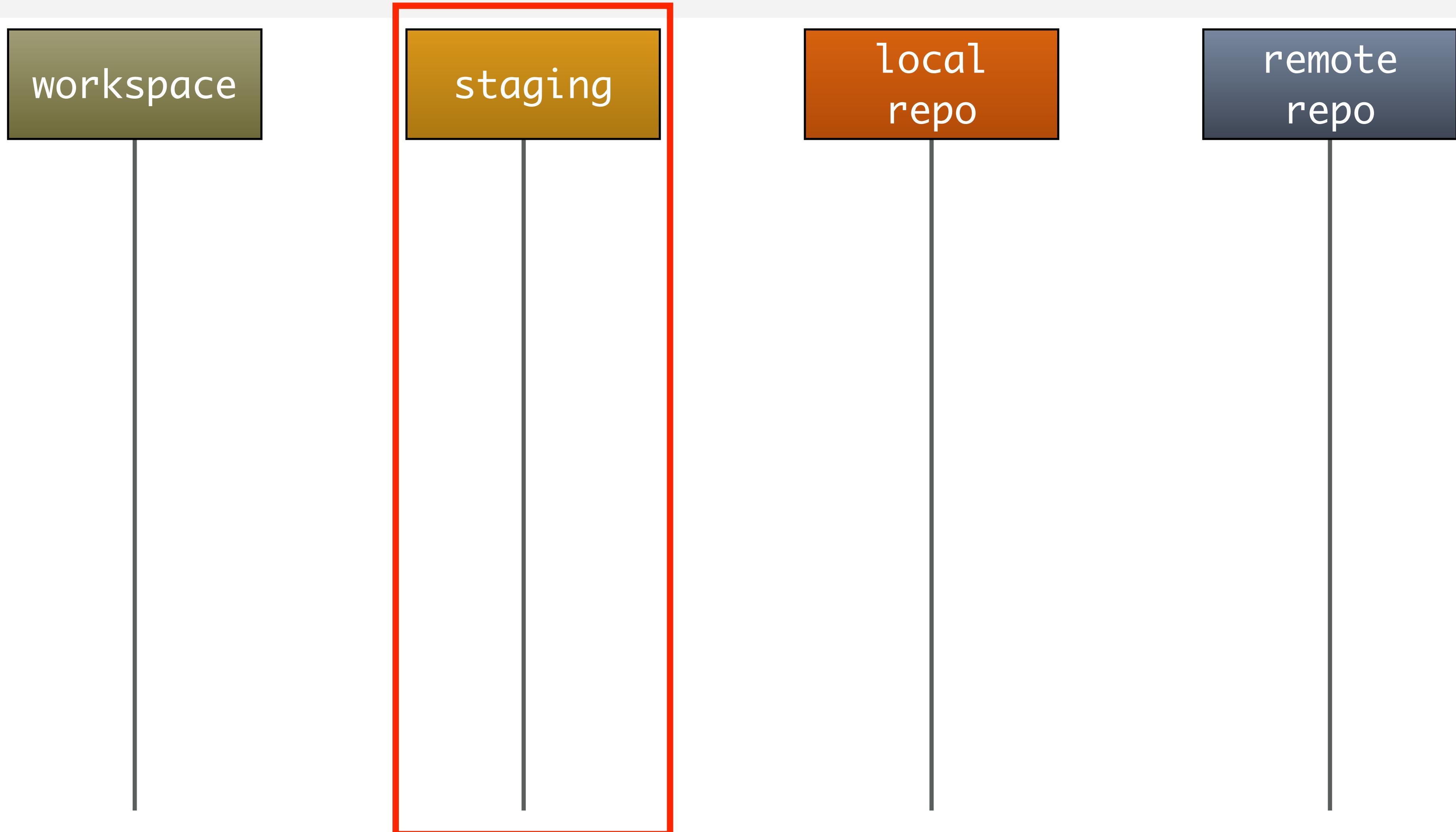
- Check your STATUS
 - You now have two new files staged for the next commit

```
$ git add index.*  
$ git status  
On branch add-home-page  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
    new file:   index.css  
    new file:   index.html  
  
$
```

Where Are Your Changes Now?



Where Are Your Changes Now?



Commit

- COMMIT your changes adding a meaningful message

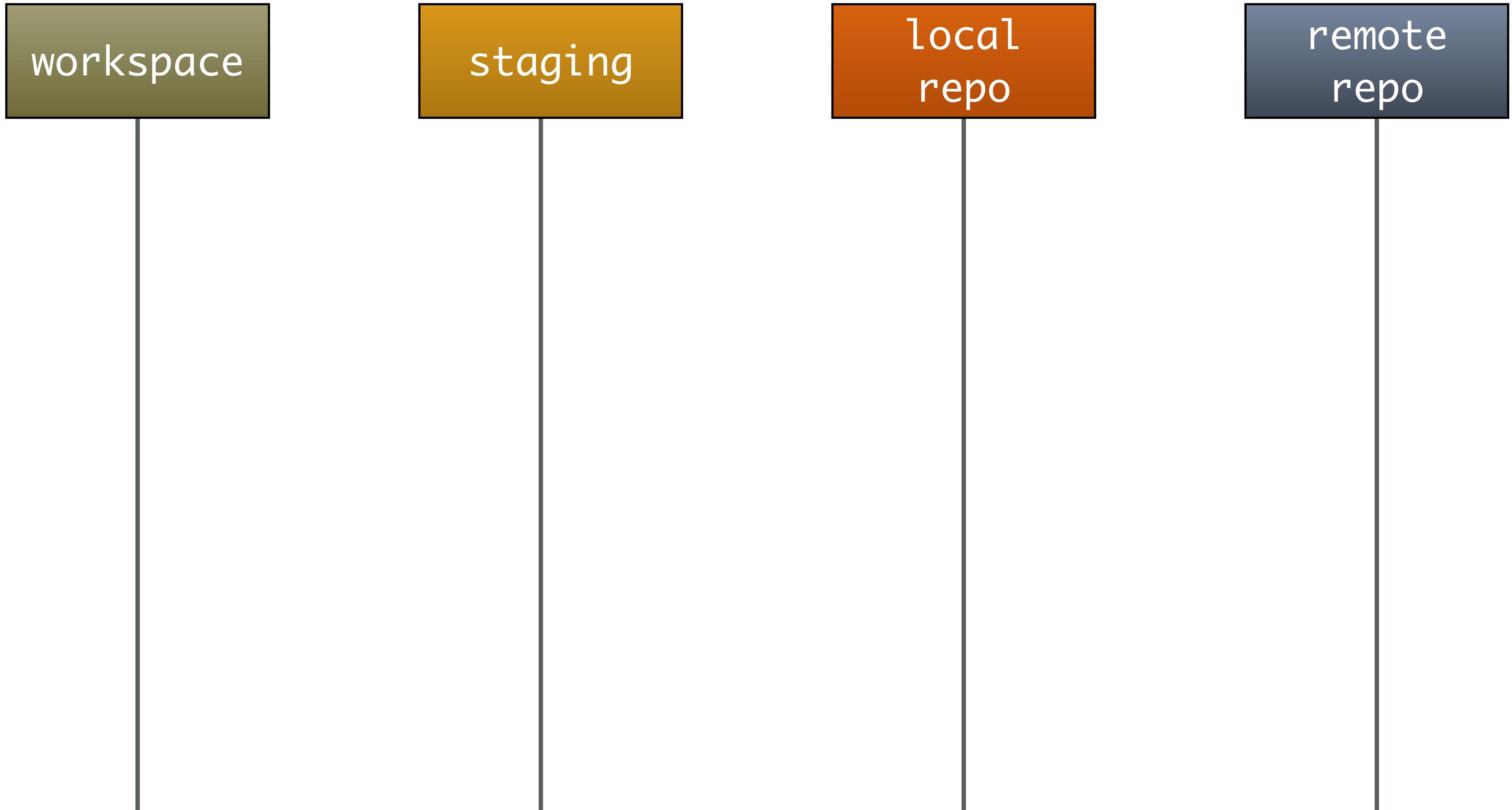
```
$ git commit -m 'added home page and style sheet'  
[add-home-page ab9d8be] added home page and style sheet  
 2 files changed, 0 insertions(+), 0 deletions(-)  
  create mode 100644 index.css  
  create mode 100644 index.html  
$
```

Commit

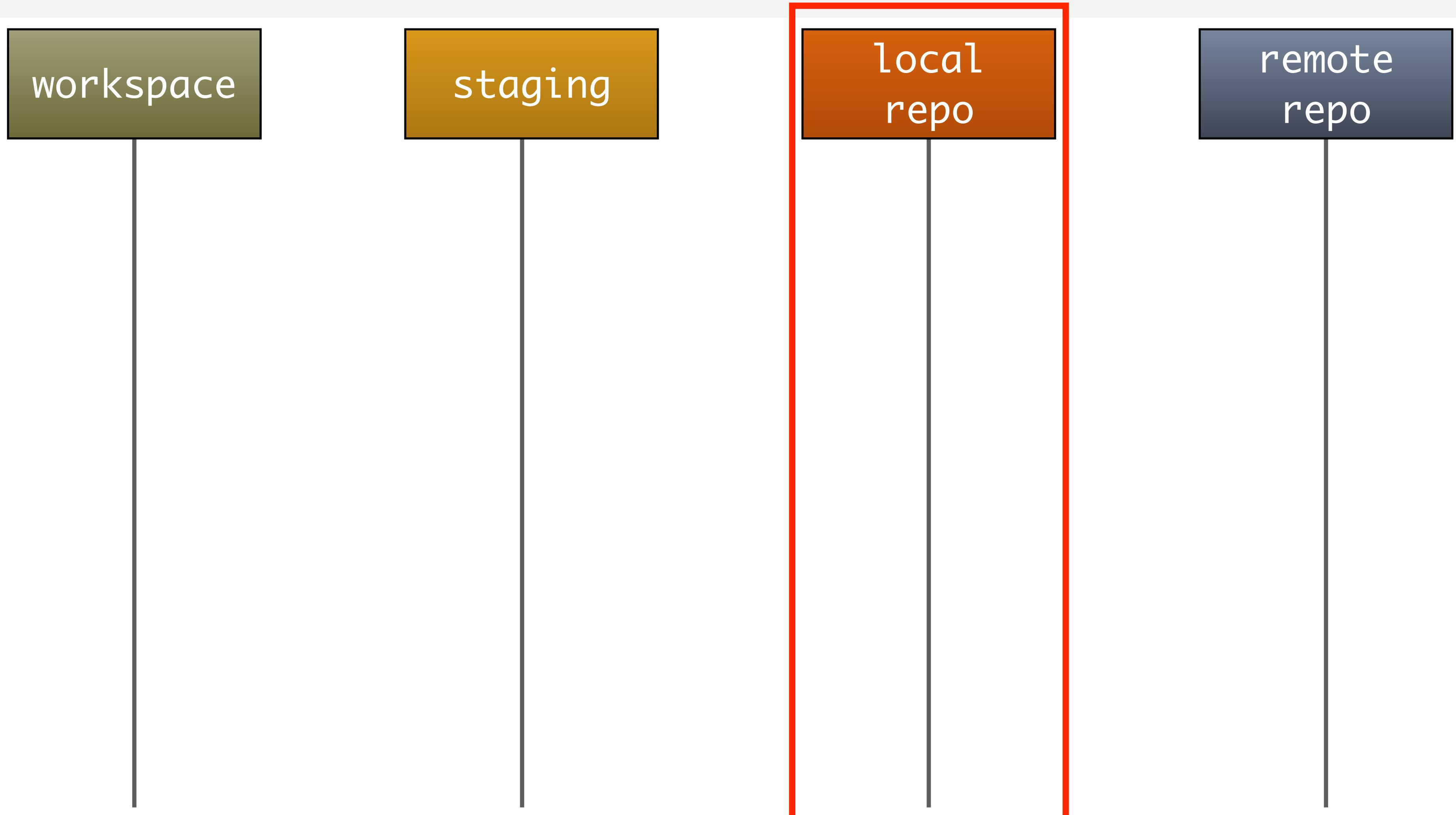
- COMMIT your changes adding a meaningful message
- Check your STATUS and note your clean directory

```
$ git commit -m 'added home page and style sheet'  
[add-home-page ab9d8be] added home page and style sheet  
 2 files changed, 0 insertions(+), 0 deletions(-)  
   create mode 100644 index.css  
   create mode 100644 index.html  
$ git status  
On branch add-home-page  
nothing to commit, working directory clean  
$
```

Where Are Your Changes Now?



Where Are Your Changes Now?



CHECKOUT

- Now that you have committed your changes you have a safe place to return to
- If you do more with on index.html and you want to revert back to the last commit, just use:

```
git checkout -- index.html
```

LOG

- We can use GIT LOG to see where we are
- But the defaults are pretty useless :(

```
$ git log  
commit ab9d8be47269d76c319246bebb17f73d7dffebda  
Author: John Rofrano <johnnyroy@johnrofrano.com>  
Date: Mon Sep 12 22:26:03 2016 -0400
```

added home page and style sheet

```
commit 907cda61614c76194512458247fc59a8f696d5b5  
Author: John Rofrano <johnnyroy@johnrofrano.com>  
Date: Mon Sep 12 15:35:05 2016 -0400
```

Initial commit

Much Improved LOG

- **--oneline** makes the log compact
- **--decorate** adds branch information
- **--all** shows all branches
- **--graph** creates a nice graph of the branches and merges

```
$ git log --oneline
ab9d8be added home page and style sheet
907cda6 Initial commit

$ git log --oneline --decorate --all --graph
* ab9d8be (HEAD -> add-home-page) added home page and style sheet
* 907cda6 (origin/master, origin/HEAD, master) Initial commit

$
```

You Can Make Aliases

- You can make an alias for any git command to make typing easier
- The syntax is: `git config --global alias.<name> "<command>"`
- To make an alias called 'lg' for this log command use:

```
$ git config --global alias.lg "log --oneline --decorate --all --graph"
```

You Can Make Aliases

- You can make an alias for any git command to make typing easier
- The syntax is: `git config --global alias.<name> "<command>"`
- To make an alias called 'lg' for this log command use:

```
$ git config --global alias.lg "log --oneline --decorate --all --graph"
```

This is the name of the alias

You Can Make Aliases

- You can make an alias for any git command to make typing easier
- The syntax is: `git config --global alias.<name> "<command>"`
- To make an alias called 'lg' for this log command use:

```
$ git config --global alias.lg "log --oneline --decorate --all --graph"
```

This is the name of the alias

This is the command it will invoke

DIFF

- We can use GIT DIFF to see the differences with another branch
- e.g. git diff master will show what is changed in our branch from the master branch

```
$ git diff master
diff --git a/index.css b/index.css
new file mode 100644
index 000000..e69de29
diff --git a/index.html b/index.html
new file mode 100644
index 000000..e69de29
```

More Git Diff

```
git diff
```

Shows what changes you've made from the index. This shows what changes are NOT committed if you

```
git diff master
```

Shows diff from workspace to master commit

```
git diff --staged  
git diff --cached
```

Shows what would be git commit'ed

```
git diff --name-only <branch-name>
```

Show only the names of the files that have changed

PUSH

- To push your COMMITs to your remote BRANCH

```
git push -u origin add-home-page
```

- This saves you if your hard drive crashes!!!
- Do this at the end of every day, or after important changes
 - Note: After initial -u you only need to use `git push`

Push to Remote Branch

- PUSH your changes to a new remote branch

```
$ git push -u origin add-home-page
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 368 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:rofrano/nyu-lab-git-intro.git
 * [new branch]      add-home-page -> add-home-page
Branch add-home-page set up to track remote branch add-home-page from origin.
$
```

Push to Remote Branch

- PUSH your changes to a new remote branch
- Check your STATUS and note you are up-to-date

```
$ git push -u origin add-home-page
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 368 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:rofrano/nyu-lab-git-intro.git
 * [new branch]      add-home-page -> add-home-page
Branch add-home-page set up to track remote branch add-home-page from origin.
$ git status
On branch add-home-page
Your branch is up-to-date with 'origin/add-home-page'.
nothing to commit, working directory clean
$
```

How Has Status Changed?

- Status now has information about the remote branch

Before

```
$ git status
On branch add-home-page
nothing to commit, working directory clean
$
```

After

```
$ git status
On branch add-home-page
Your branch is up-to-date with 'origin/add-home-page'.
nothing to commit, working directory clean
$
```

How Has Status Changed?

- Status now has information about the remote branch

Before

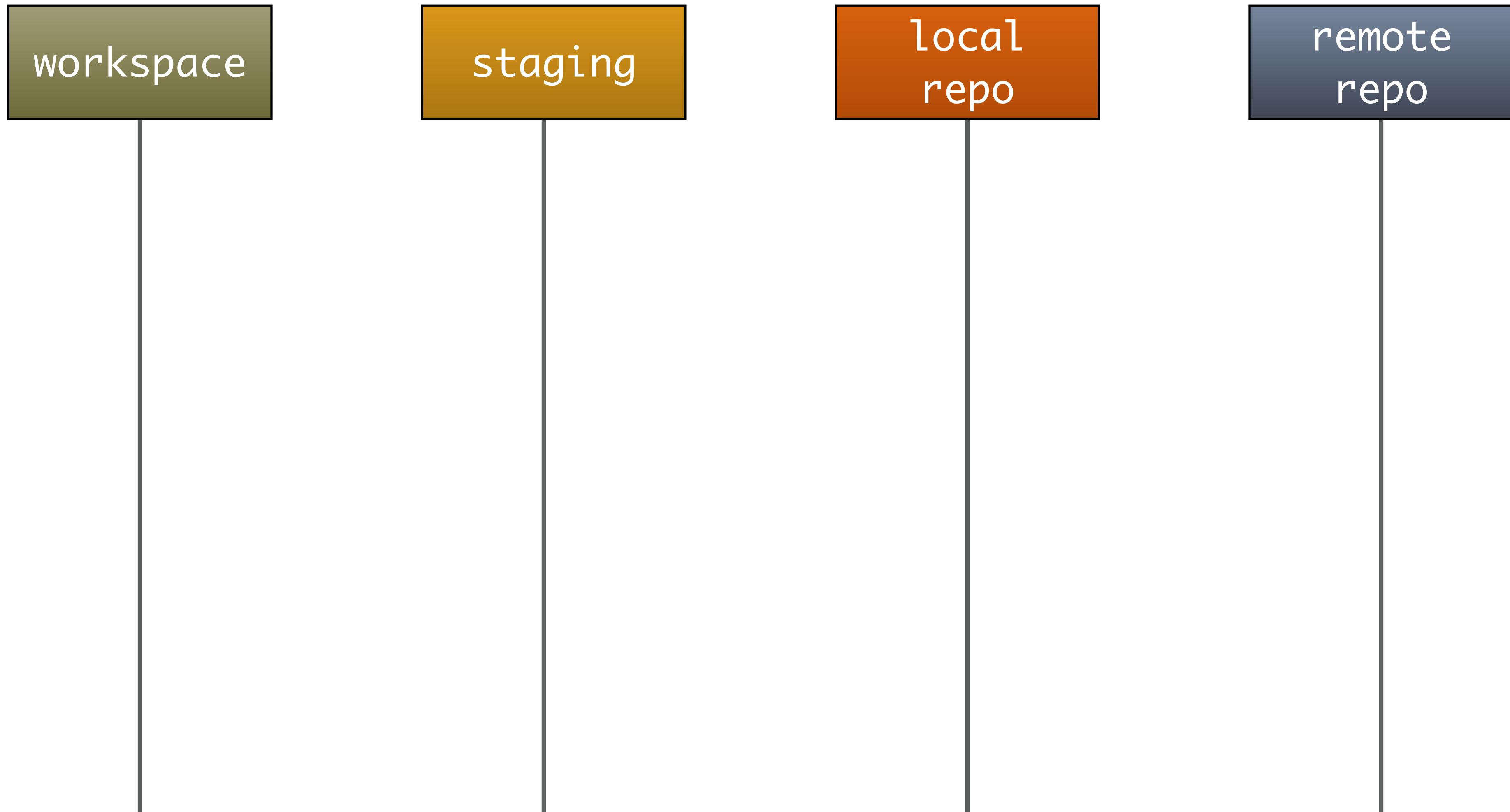
```
$ git status  
On branch add-home-page  
nothing to commit, working directory clean  
$
```

After

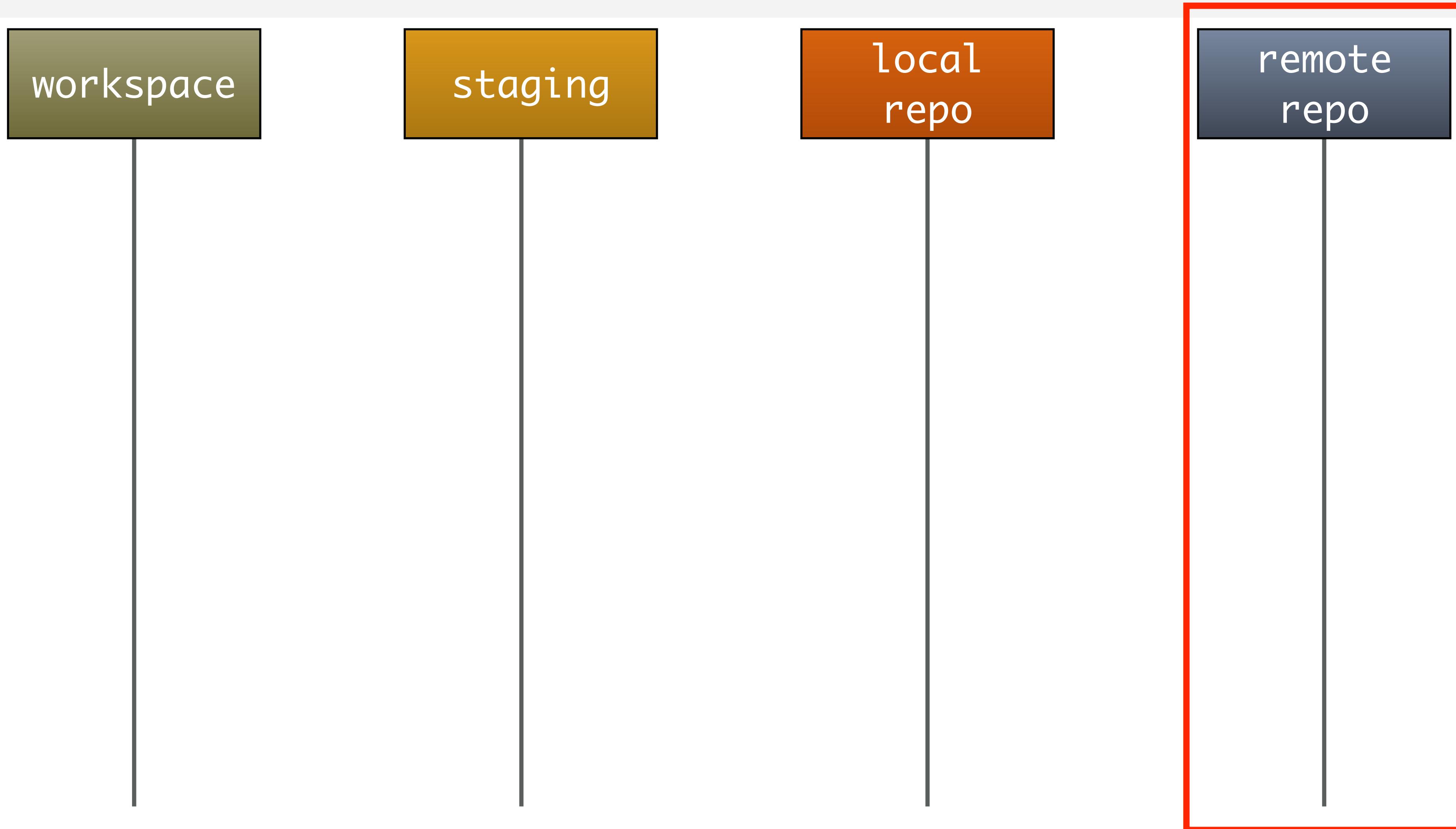
```
$ git status  
On branch add-home-page  
Your branch is up-to-date with 'origin/add-home-page'.  
nothing to commit, working directory clean  
$
```

Information about the remote branch

Where Are Your Changes Now?



Where Are Your Changes Now?



Check the LOG

- We can view the logs to see where we are at now as well
- HEAD points to both the remote and local branch add-home-page

```
$ git log --oneline --decorate --all --graph
* ab9d8be (HEAD -> add-home-page, origin/add-home-page) added home page and style sheet
* 907cda6 (origin/master, origin/HEAD, master) Initial commit
$
```

Check the LOG

- We can view the logs to see where we are at now as well
- HEAD points to both the remote and local branch add-home-page

```
$ git log --oneline --decorate --all --graph
* ab9d8be (HEAD -> add-home-page, origin/add-home-page) added home page and style sheet
* 907cda6 (origin/master, origin/HEAD, master) Initial commit
$
```

HEAD is the most recent commit in the history

Pull request workflow

- The pull request workflow helps us to share information. It gives the following benefits:
- **A second set of eyes:** Sometimes you may have solved a problem, but when someone reviews it they pointed out a subtle flaw, or perhaps a way you could solve it more elegantly. It also helps identify things like security exposures. This results in a much cleaner codebase.
- **Knowledge sharing:** It is dangerous to only have one developer that understands the code. Having another developer review the code allows for shared understanding and protects against one developer leaving the project.
- **Allows automation of unit testing:** You setup your github project to run unit tests on pull requests so that we can verify that everything works before merging.

Prepare for Pull Request

- You want to make sure that you are using the latest code before you make a pull request
- The remote master branch has probably moved on since you created your branch
- Therefore it is necessary "rebase to" or "merge with" the current master



Fetch and Pull

- Fetch will bring down all of the changes from the remote repository
- Pull will merge the changes from the current branch with your local workspace
 - Note: Pull does a fetch so if you just want to pull there is no need to fetch
- We do this on the master branch



Merge

- Before you issue a **Pull Request**, it's a good idea to merge from master
- Merge joins two branches together, usually your branch and master but it can be any two branches only changing the checked out branch
- If there are no conflicts and the current HEAD is an ancestor of the merge branch, git will fast-forward the merge by moving the pointer forward
- Merge retains all the history as it happened chronologically



Refreshing Master Before PR

- Switch to the master branch
- Pull the latest changes
- In this case we were already up to date

```
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
$ git pull
Already up-to-date.
$
```

Merge from Current Master

- Change to your branch to prepare to merge

```
$ git checkout add-home-page
Switched to branch 'add-home-page'
Your branch is up-to-date with 'origin/add-home-page'.
$
```

Merge from Master

- Issue the `merge` command to merge any the new code

```
$ git checkout add-home-page
Switched to branch 'add-home-page'
Your branch is up-to-date with 'origin/add-home-page'.
$ git merge master
Current branch add-home-page is up to date.
$
```

Push to Remote Branch

- Once the merge is complete, push the results to your remote branch

```
$ git checkout add-home-page
Switched to branch 'add-home-page'
Your branch is up-to-date with 'origin/add-home-page'.
$ git merge master
Current branch add-home-page is up to date.
$ git push
Everything up-to-date
$
```

REBASE

- Rebase will replay your changes on top of the current version of master
- It will look as if your branch was created from the current state of master
- You can optionally do this before pushing/creating to remote branch otherwise use merge
- Never rebase once you have pushed to a remote. It will mess everyone else up because it changes history in a way that is incompatible



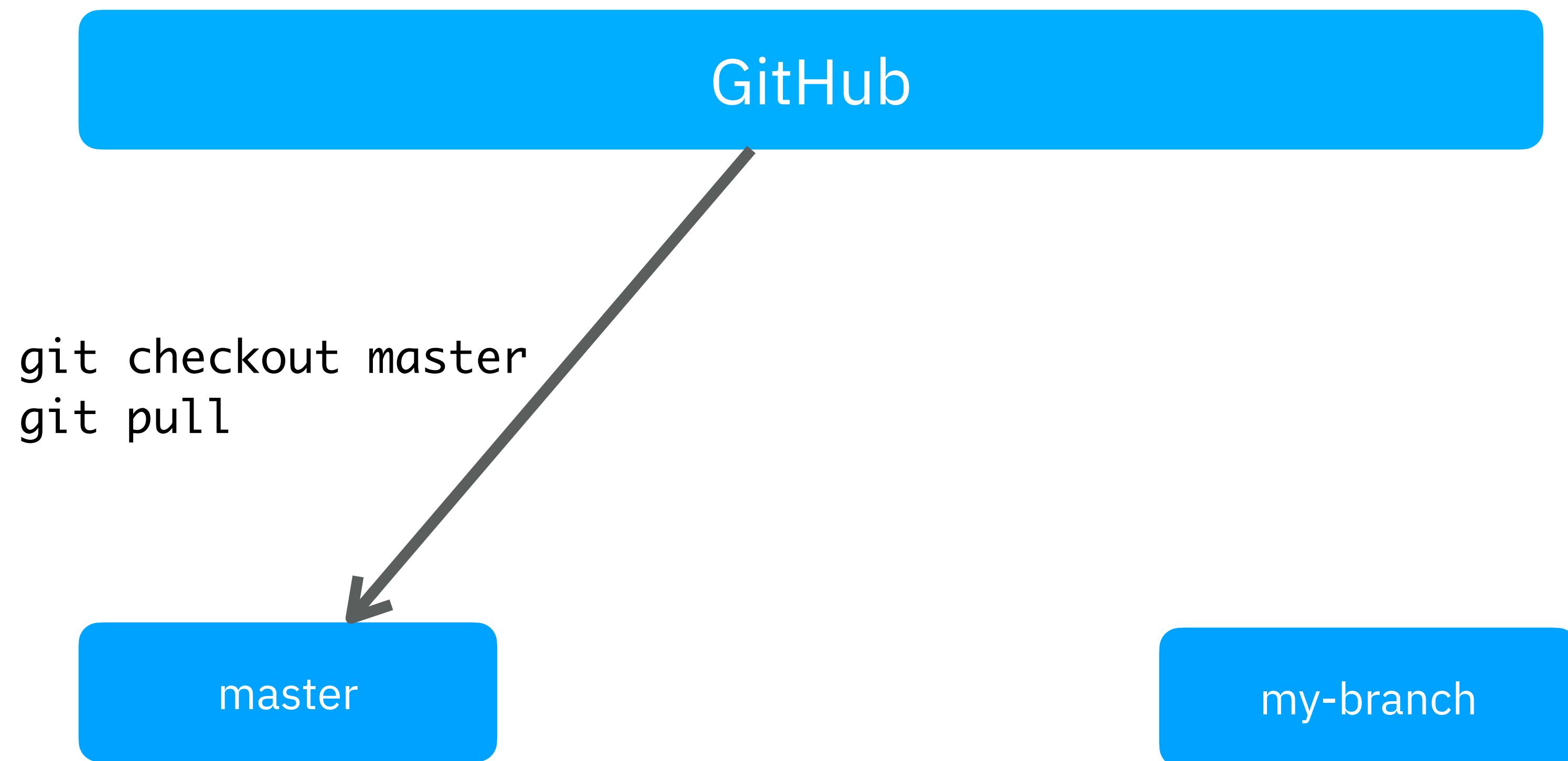
Pull Request Prep Summary

GitHub

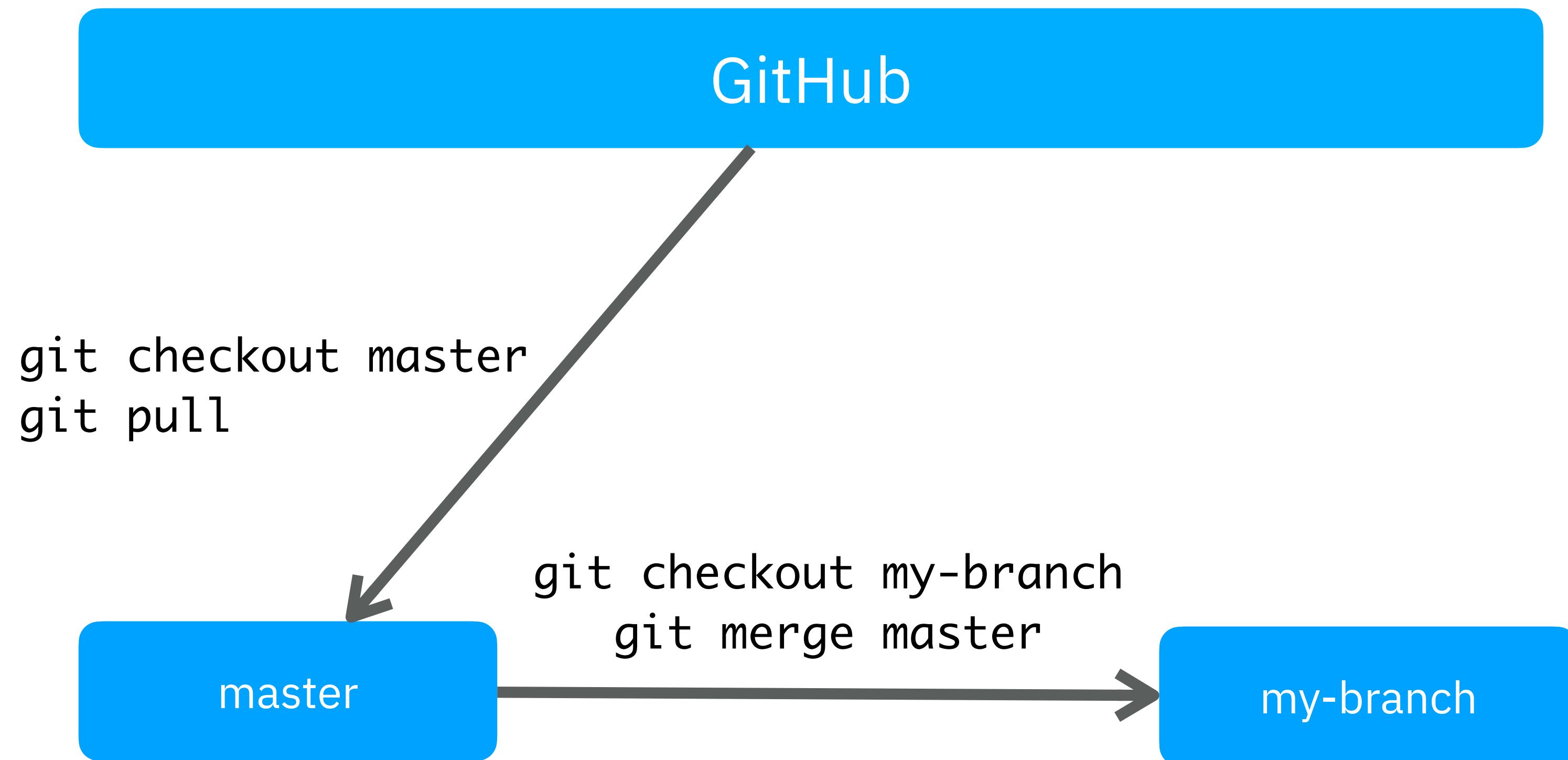
master

my-branch

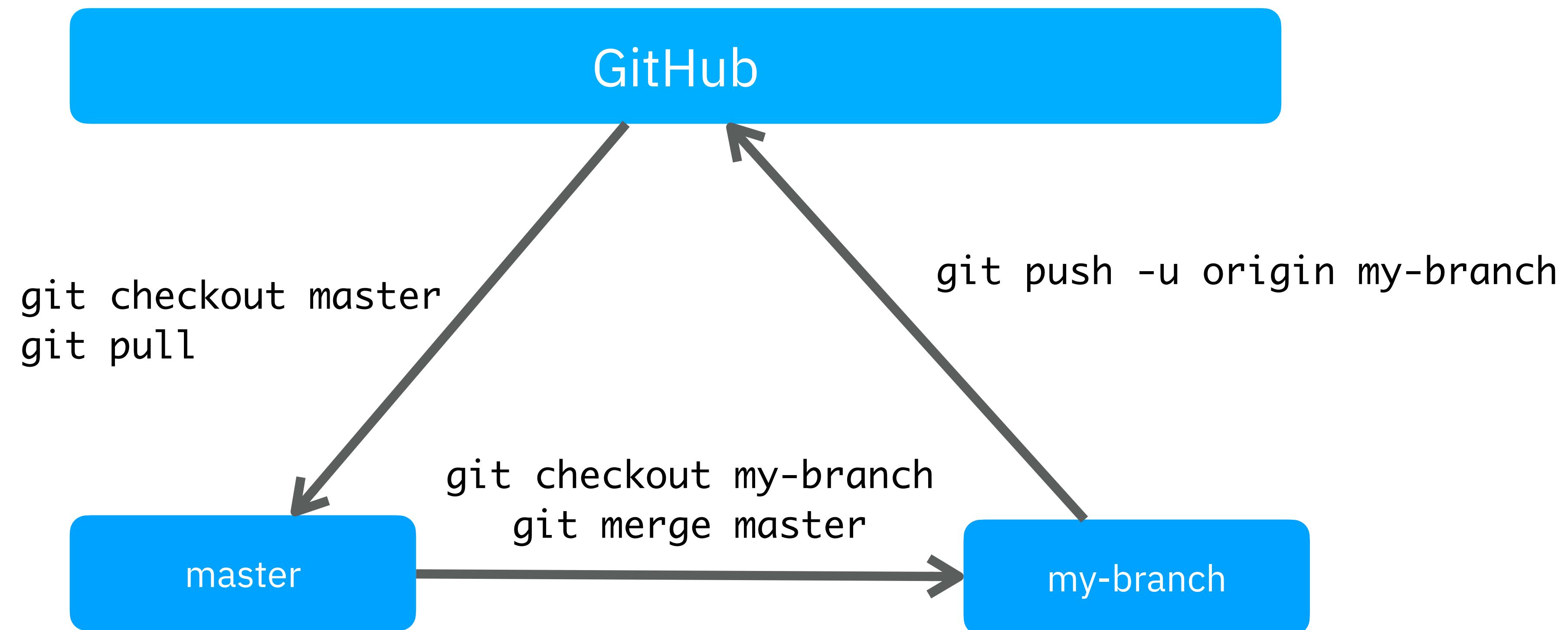
Pull Request Prep Summary



Pull Request Prep Summary

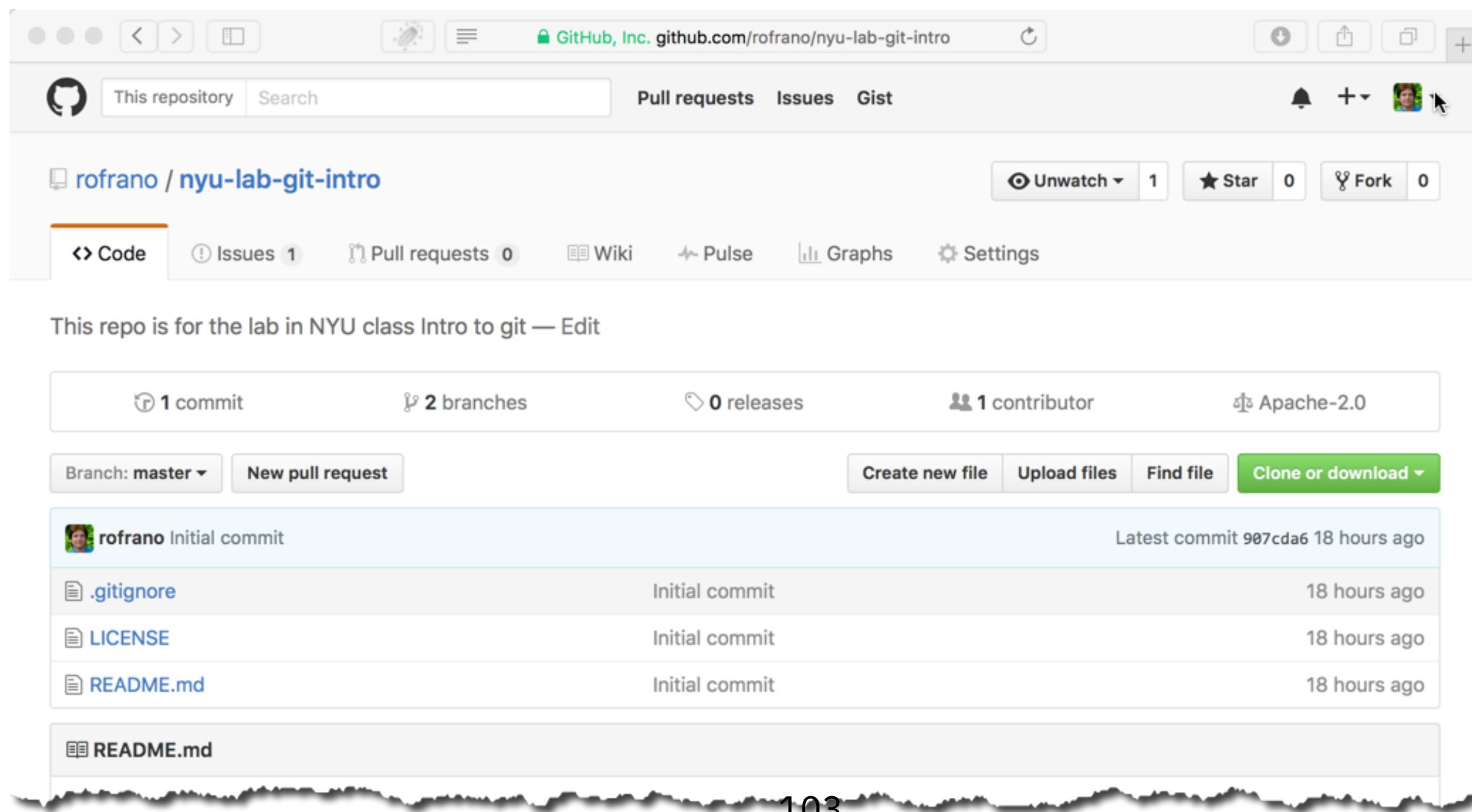


Pull Request Prep Summary



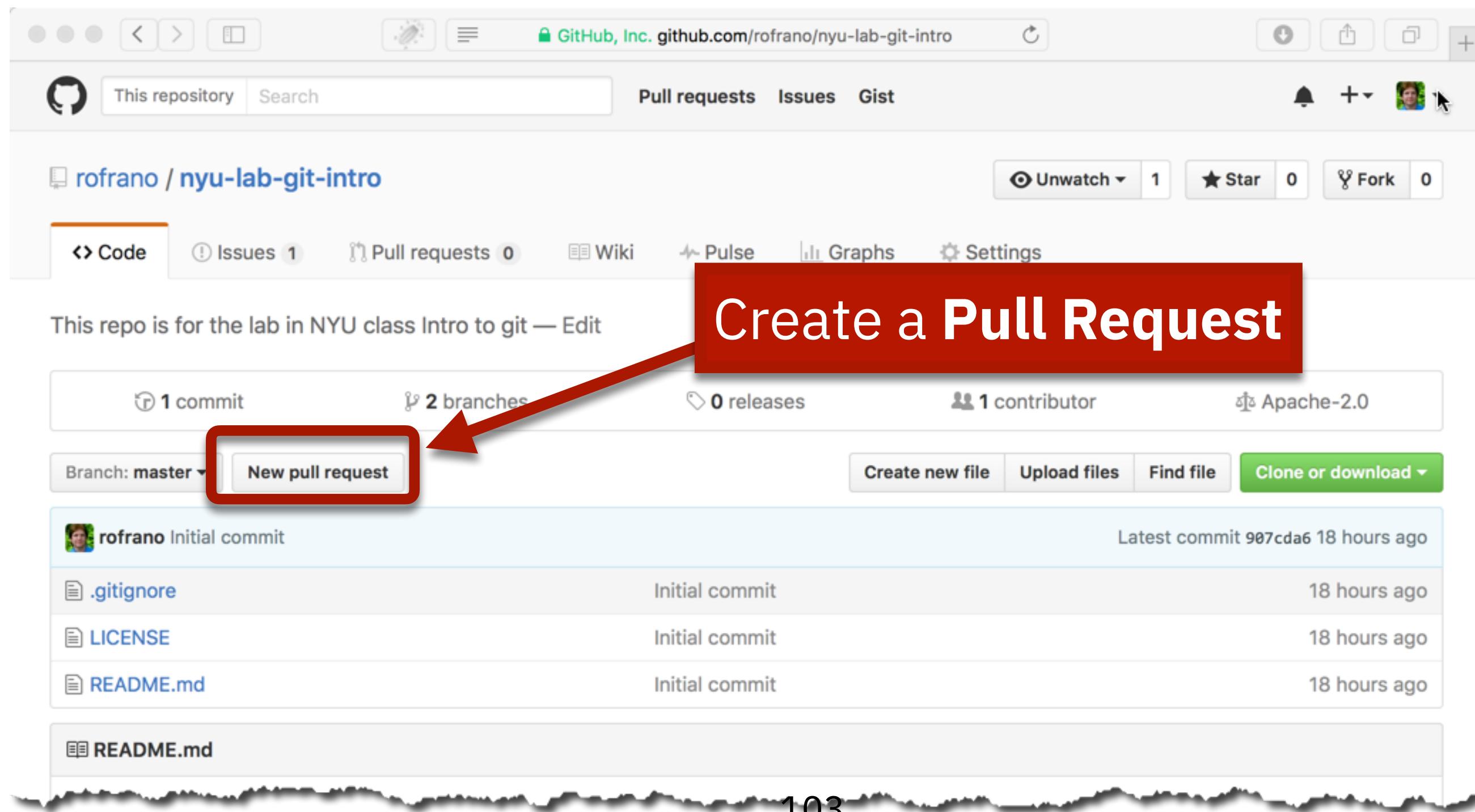
Pull Request

- Use a PULL REQUEST to notify the team that your changes are ready to be reviewed and merged into the MASTER branch
- If you are working on a Fork, initiate the Pull Request from the Forked repository



Pull Request

- Use a PULL REQUEST to notify the team that your changes are ready to be reviewed and merged into the MASTER branch
- If you are working on a Fork, initiate the Pull Request from the Forked repository



Create Pull Request

The screenshot shows the GitHub interface for creating a pull request. At the top, the repository 'rofrano / nyu-lab-git-intro' is selected. The 'Pull requests' tab is active. Below the header, there's a summary bar showing 'Issues 1', 'Pull requests 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The main area is titled 'Open a pull request' and instructs the user to 'Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.' A dropdown menu shows 'base: master' and 'compare: add-home-page', with a note that they are 'Able to merge'. The pull request body contains a message from 'rofrano' stating 'added home page and style sheet' and a detailed description: 'Add the index.html page like we discussed and used a style sheet to style it.' A bold 'Closes Issue #1' is present. On the right, there are sections for 'Labels' (None yet), 'Milestone' (No milestone), and 'Assignees' (No one—assign yourself). At the bottom, there's a note about Markdown support and a large green 'Create pull request' button.

GitHub, Inc. github.com/rofrano/nyu-lab-git-intro/compare/master...add-home-page

This repository Search

Pull requests Issues Gist

rofrano / [nyu-lab-git-intro](#)

Code Issues 1 Pull requests 0 Wiki Pulse Graphs Settings

Unwatch 1 Star 0 Fork 0

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master ... compare: [add-home-page](#) ✓ Able to merge. These branches can be automatically merged.

 added home page and style sheet

Write Preview

Add the index.html page like we discussed and used a style sheet to style it.

Closes Issue #1

Attach files by dragging & dropping or [selecting them](#).

Styling with Markdown is supported

Create pull request

Labels None yet

Milestone No milestone

Assignees No one—assign yourself

Create Pull Request

The screenshot shows the GitHub interface for creating a pull request. At the top, the repository 'rofrano / nyu-lab-git-intro' is selected. The 'Pull requests' tab is active. Below the header, there's a summary: 'Issues 1' and 'Pull requests 0'. A red box highlights the 'base: master' dropdown menu, which is set to 'compare: add-home-page'. To the right of this, a green message says 'Able to merge. These branches can be automatically merged.' A red arrow points from a callout box containing the text 'Refreshing from master ensures you can merge' towards this message. The main body of the page contains a commit message: 'added home page and style sheet' and a description: 'Add the index.html page like we discussed and used a style sheet to style it.' Below this, the text 'Closes Issue #1' is displayed. On the right side, there are sections for 'Labels' (None yet), 'Milestone' (No milestone), and 'Assignees' (No one—assign yourself). At the bottom, there's a note about Markdown support and a large green 'Create pull request' button.

Refreshing from master ensures you can merge

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ... compare: add-home-page ✓ Able to merge. These branches can be automatically merged.

added home page and style sheet

Add the index.html page like we discussed and used a style sheet to style it.

Closes Issue #1

Attach files by dragging & dropping or selecting them.

Styling with Markdown is supported

Create pull request

Create Pull Request

The screenshot shows the GitHub interface for creating a pull request. At the top, there's a red callout box with the text "Refreshing from master ensures you can merge". A red arrow points from this box to the "base: master" dropdown menu, which is highlighted with a red border. Below the dropdown, a green success message says "Able to merge. These branches can be automatically merged." In the main body of the form, there's a title field containing "added home page and style sheet" and a description field with the text "Add the index.html page like we discussed and used a style sheet to style it.". A red callout box on the right side contains the text "Title defaults to commit message. just add a Description and then press Create pull request". Two red arrows point from this callout box to the title and description fields respectively. At the bottom right of the form is a green "Create pull request" button, which is also highlighted with a red border.

Refreshing from master ensures you can merge

base: master ... compare: add-home-page ✓ Able to merge. These branches can be automatically merged.

added home page and style sheet

Add the index.html page like we discussed and used a style sheet to style it.

Closes Issue #1

Attach files by dragging & dropping or selecting them.

Styling with Markdown is supported

Create pull request

Title defaults to commit message. just add a Description and then press Create pull request

Pull Request Details

The screenshot shows a GitHub pull request page for the repository `rofrano / nyu-lab-git-intro`. The pull request is titled "added home page and style sheet #2". It is currently open, with the message: "rofrano wants to merge 1 commit into `master` from `add-home-page`". The commit message is "added home page and style sheet". A comment from `rofrano` says: "Add the index.html page like we discussed and used a style sheet to style it." The pull request has 1 commit and 2 files changed. The merge status indicates "This branch has no conflicts with the base branch" and "Merging can be performed automatically". There is a green "Merge pull request" button. On the right side, there are sections for Labels (None yet), Milestone (No milestone), Assignees (No one—assign yourself), and Notifications (1 participant). The participant is listed as "rofrano".

Pull Request Details

Use Pull Request tab to view requests

The screenshot shows a GitHub repository page for 'rofrano / nyu-lab-git-intro'. A red box highlights the 'Pull requests' tab, which is currently selected and shows 1 pull request. A red arrow points from the 'Use Pull Request tab to view requests' text to this tab. The pull request details are displayed below, showing a commit from 'rofrano' titled 'added home page and style sheet #2'. The commit message includes a comment from 'rofrano' asking to add an index.html page and a style sheet. The pull request has 1 commit and 2 files changed. It is ready to merge automatically. On the right side, there are sections for Labels, Milestone, Assignees, and Notifications.

Pull Request Details

Use Pull Request tab to view requests

The screenshot shows a GitHub repository page for 'rofrano / nyu-lab-git-intro'. A red box highlights the 'Pull requests' tab in the navigation bar, with an arrow pointing from the text 'Use Pull Request tab to view requests'. Another red box highlights the title of a specific pull request, 'added home page and style sheet #2', with an arrow pointing from the text 'Pull Request is a new Issue'. The pull request details show a commit message from 'rofrano' and a merge button at the bottom.

Pull Request is a new Issue

rofrano / nyu-lab-git-intro

Pull requests 1

Issues 1

Code Wiki Pulse Graphs Settings

added home page and style sheet #2

Open rofrano wants to merge 1 commit into master from add-home-page

Conversation 0 Commits 1 Files changed 2

rofrano commented just now

Add the index.html page like we discussed and used a style sheet to style it.

added home page and style sheet ab9d8be

Add more commits by pushing to the add-home-page branch on rofrano/nyu-lab-git-intro.

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Labels None yet

Milestone No milestone

Assignees No one—assign yourself

1 participant

Notifications

Unsubscribe

View Commits

The screenshot shows a GitHub pull request page for the repository `rofrano / nyu-lab-git-intro`. The pull request is titled "added home page and style sheet #2". It is marked as "Open" and shows one commit from the branch `add-home-page` into the `master` branch. The commit message is "added home page and style sheet". It was committed by John J. Rofrano 12 hours ago. The commit hash is `ab9d8be`.

The page includes standard GitHub navigation links like "Code", "Issues 1", "Pull requests 1", "Wiki", "Pulse", "Graphs", and "Settings". It also shows metrics for the pull request: 0 conversations, 1 commit, and 2 files changed. The GitHub footer at the bottom includes links for "Contact GitHub", "API", "Training", "Shop", "Blog", and "About".

View Commits

The screenshot shows a GitHub pull request page for a repository named 'rofrano / nyu-lab-git-intro'. The pull request has been opened by 'rofrano' and is merging 1 commit from the 'add-home-page' branch into the 'master' branch. The 'Pull requests' tab is active. A red box highlights the 'Commits' button, which shows 1 commit. A red arrow points from this button to a callout bubble containing the text 'View the **Commits** that are included'. Below the commits, there is a list of changes: 'added home page and style sheet' by John J. Rofrano committed 12 hours ago. The commit hash is ab9d8be. The bottom of the page includes standard GitHub footer links like Terms, Privacy, Security, Status, Help, Contact GitHub, API, Training, Shop, Blog, and About.

View the **Commits** that are included

added home page and style sheet

rofrano wants to merge 1 commit into master from add-home-page

Commits 1

Conversation 0

Files changed 2

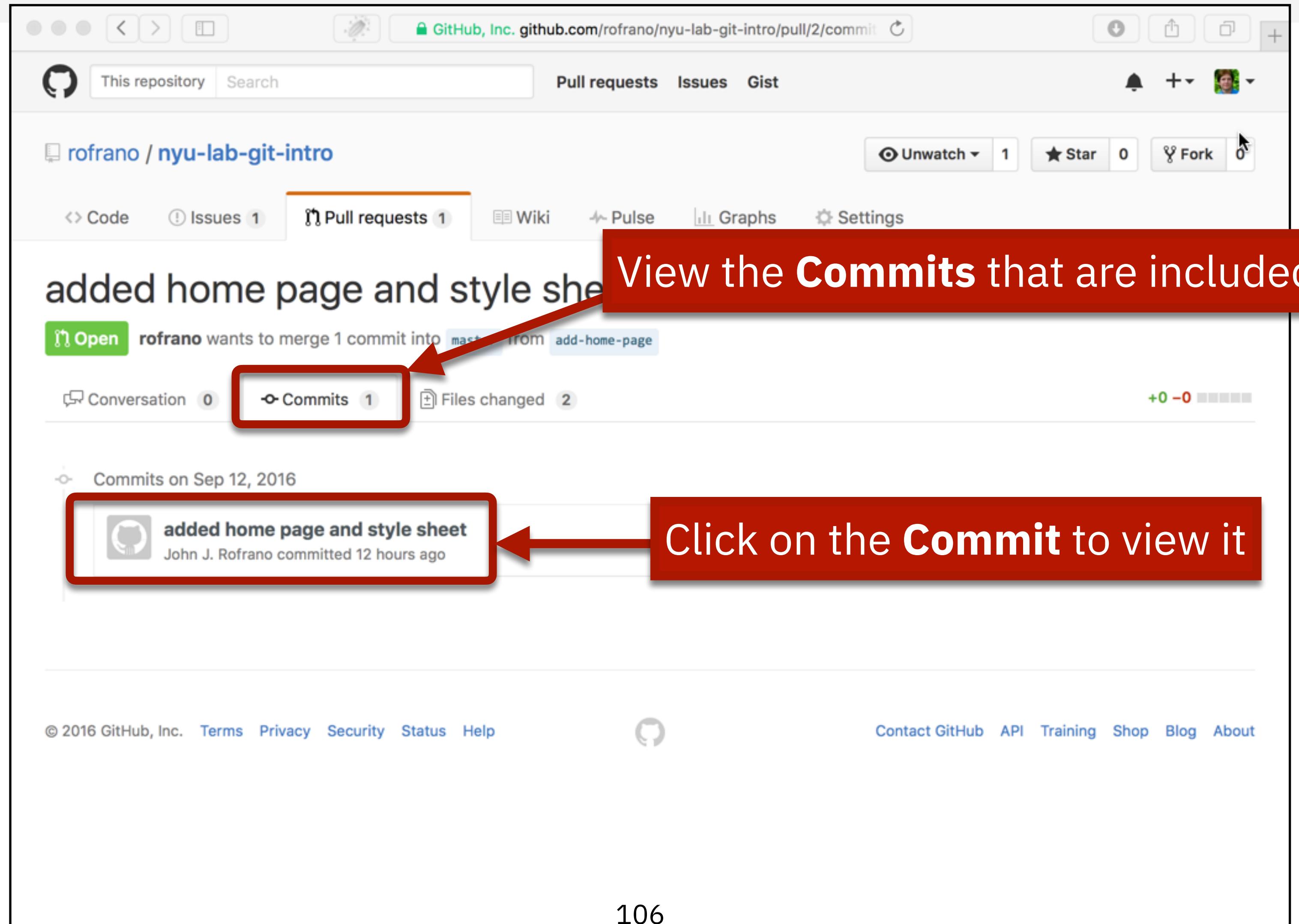
Commits on Sep 12, 2016

added home page and style sheet
John J. Rofrano committed 12 hours ago
ab9d8be

© 2016 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About

View Commits



Files Changed

The screenshot shows a GitHub pull request page for a repository named 'rofrano / nyu-lab-git-intro'. The pull request is titled 'added home page and style sheet #2' and is marked as 'Open'. It shows one commit from 'rofrano' merging into the 'master' branch from the 'add-home-page' branch. The 'Files changed' tab is selected, showing two files: 'index.css' and 'index.html', both of which have 'No changes.'

GitHub, Inc. github.com/rofrano/nyu-lab-git-intro/pull/2/files

This repository Search Pull requests Issues Gist

rofrano / nyu-lab-git-intro Unwatch 1 Star 0 Fork 0

Code Issues 1 Pull requests 1 Wiki Pulse Graphs Settings

added home page and style sheet #2 Edit

Open rofrano wants to merge 1 commit into master from add-home-page

Conversation 0 Commits 1 Files changed 2

Changes from all commits ▾ 2 files ▾ +0 -0 Options ▾

0 index.css View

No changes.

0 index.html View

No changes.

© 2016 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub API Training Shop Blog About

Files Changed

The screenshot shows a GitHub pull request page for a repository named 'rofrano / nyu-lab-git-intro'. The pull request is titled 'added home page and style sheet #2' and is marked as 'Open'. It is merging 1 commit from the 'add-home-page' branch into the 'master' branch. The 'Files changed' tab is highlighted with a red box and a red arrow points to it from the text 'View the files that have changed'. The page displays two files: 'index.css' and 'index.html', both showing 'No changes.'

View the files that have changed

added home page and style sheet #2

Open rofrano wants to merge 1 commit into master from add-home-page

Conversation 0 Commits 1 Files changed 2

Changes from all commits ▾ 2 files ▾

0 index.css View

No changes.

0 index.html View

No changes.

© 2016 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub API Training Shop Blog About

Files Changed

A screenshot of a GitHub pull request page. The URL in the address bar is `github.com/rofrano/nyu-lab-git-intro/pull/2/files`. The repository name is `rofrano / nyu-lab-git-intro`. The pull request title is "added home page and style sheet #2". The "Pull requests" tab is selected. A red box highlights the "Files changed" button, which has a value of "2". A red arrow points from this button to a callout bubble containing the text "View the files that have changed". Below the button, a red box highlights the list of files: "index.css" and "index.html", both showing "No changes." A second red arrow points from this list to another callout bubble containing the text "These are the files that have changed".

added home page and style sheet #2

Open rofrano wants to merge 1 commit into master from add-home-page

Conversation 0 Commits 1 Files changed 2

Changes from all commits ▾ 2 files ▾

0 index.css
No changes.

0 index.html
No changes.

View

View

© 2016 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About

View the files that have changed

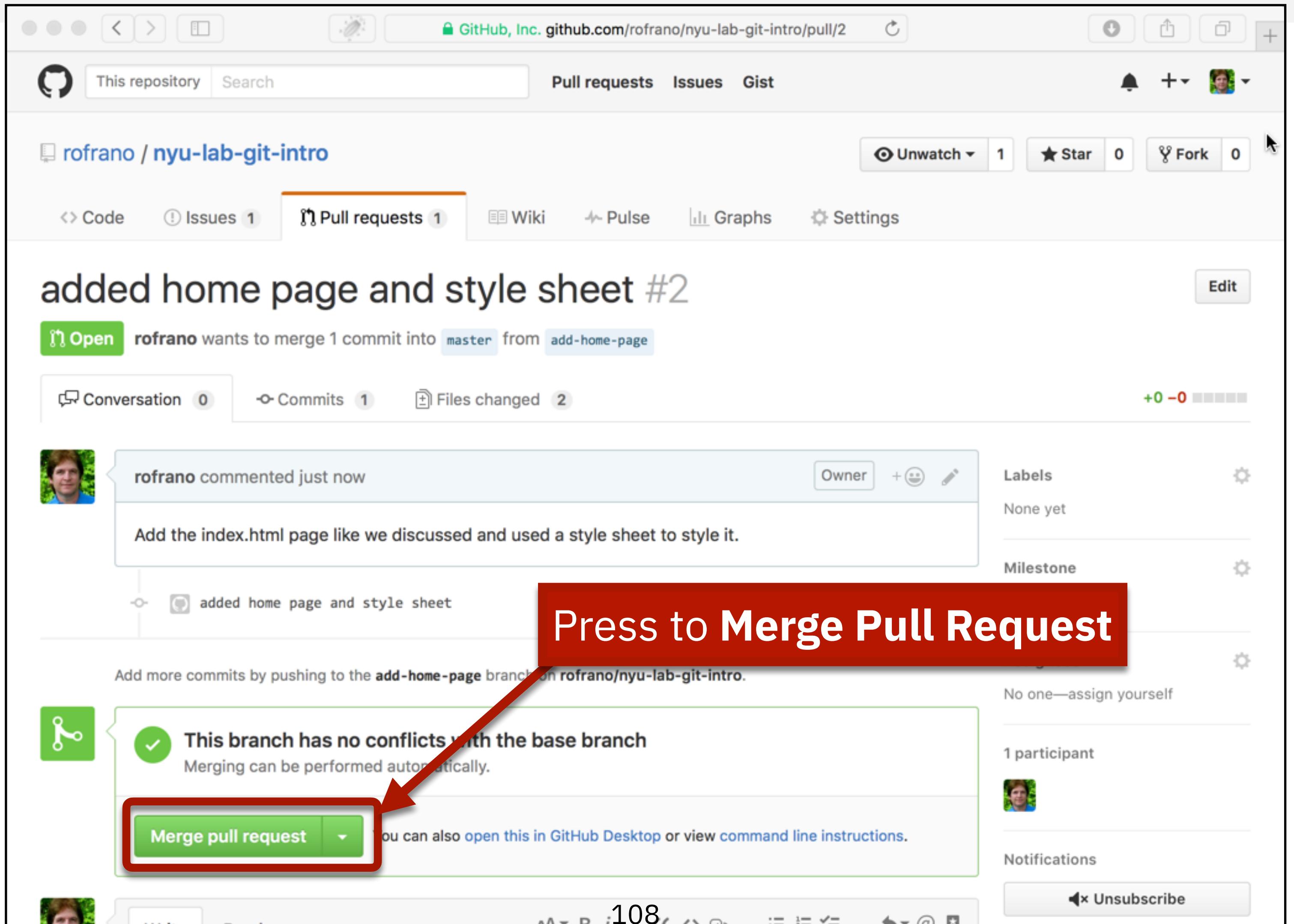
These are the files that have changed

Merge Pull Request

The screenshot shows a GitHub pull request page for a repository named 'rofrano / nyu-lab-git-intro'. The pull request is titled 'added home page and style sheet #2' and is currently 'Open'. It is merging 1 commit from the 'add-home-page' branch into the 'master' branch. The commit message is 'added home page and style sheet'. A comment from 'rofrano' says 'Add the index.html page like we discussed and used a style sheet to style it.' The pull request is marked as 'Mergeable' with a green icon and the message 'This branch has no conflicts with the base branch'. There is a 'Merge pull request' button. On the right side, there are sections for Labels (None yet), Milestone (No milestone), Assignees (No one—assign yourself), and Notifications (1 participant). The GitHub interface includes standard navigation bars like 'Pull requests', 'Issues', 'Gist', and 'Settings'.

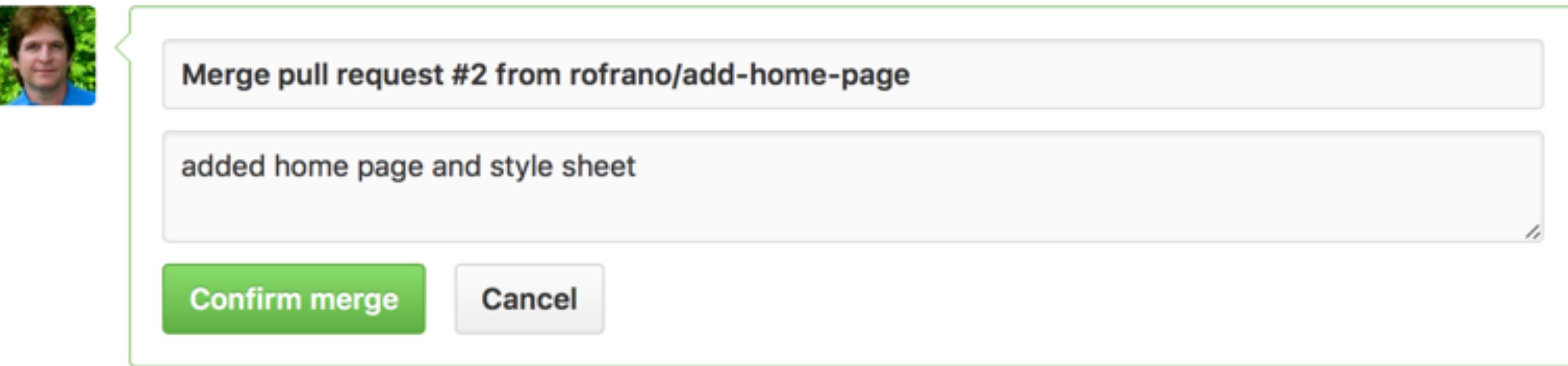
Note: Testing can be automated with tools like Travis CI, Jenkins, etc.

Merge Pull Request

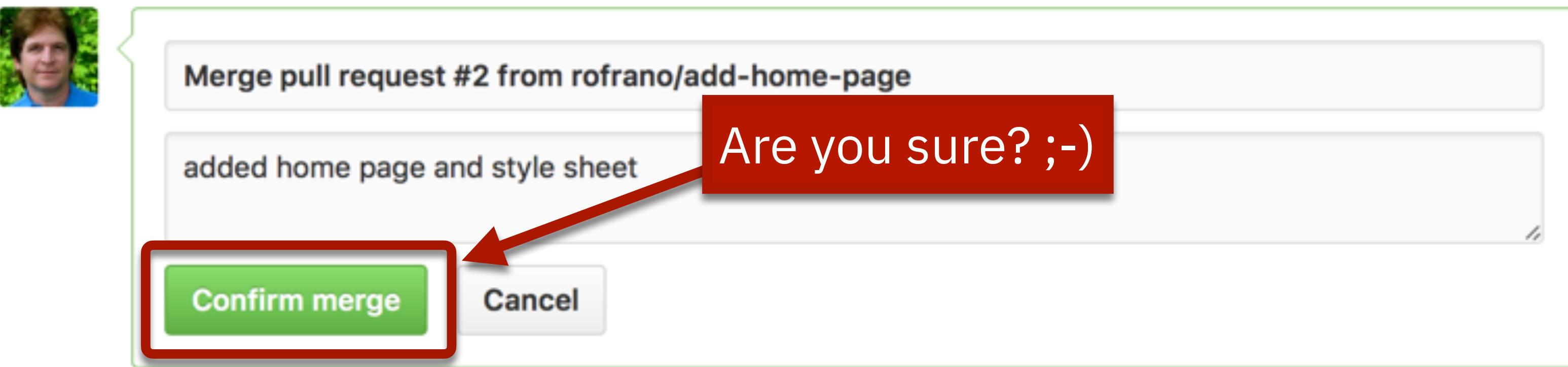


Note: Testing can be automated with tools like Travis CI, Jenkins, etc.

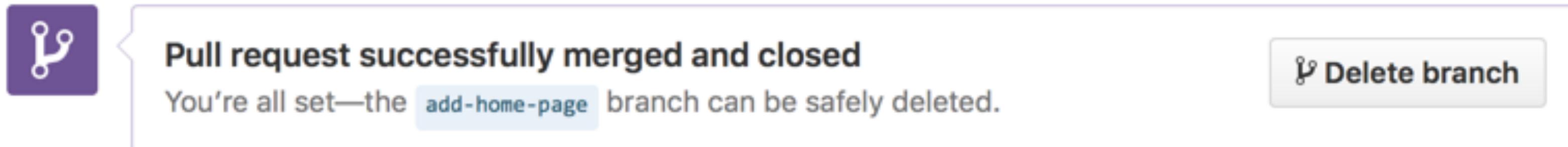
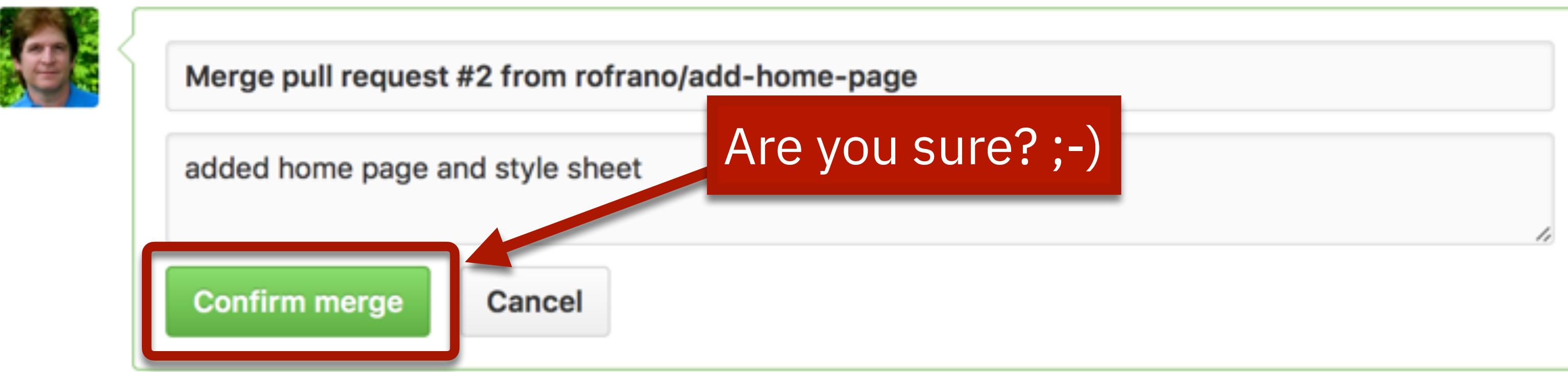
Complete the Merge



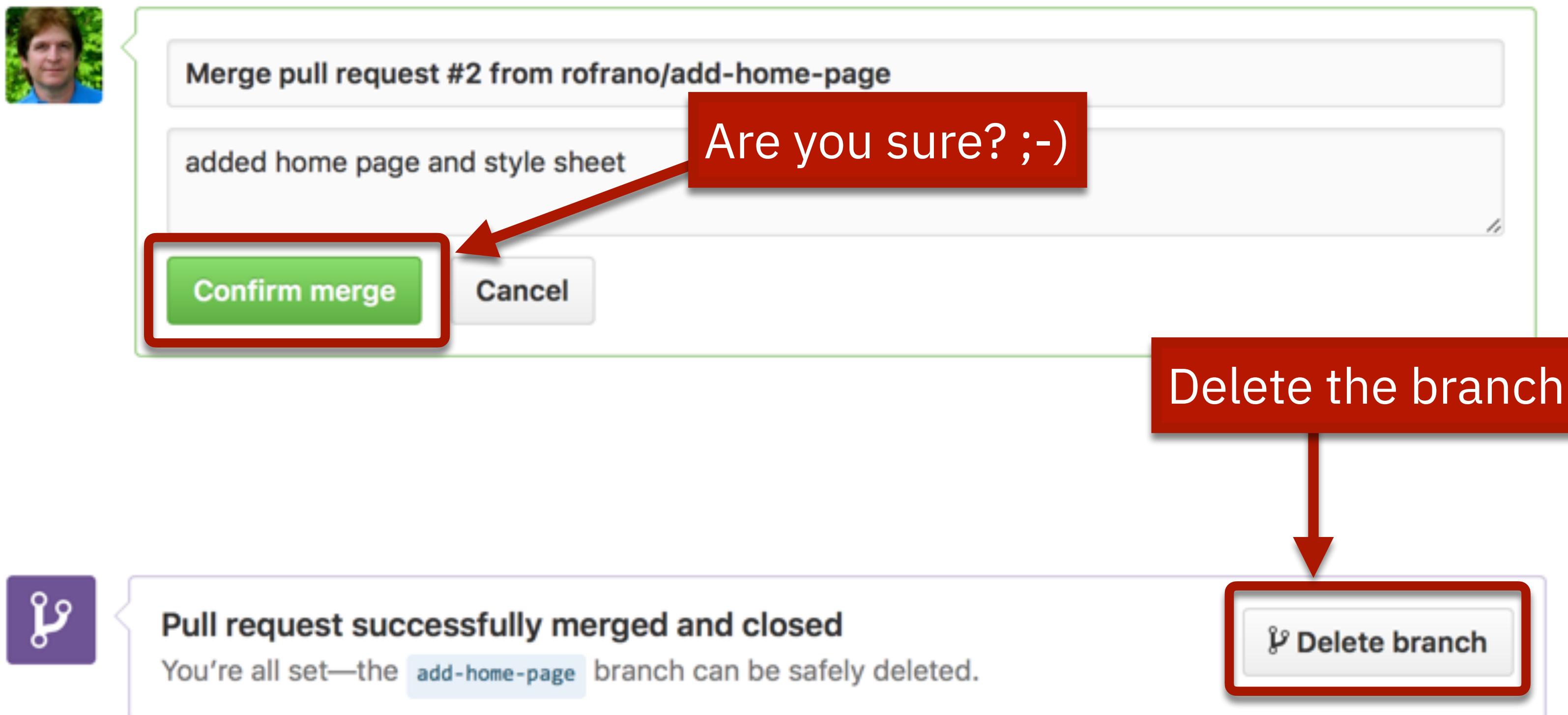
Complete the Merge



Complete the Merge



Complete the Merge



Delete Your Local Branch

- You will need to delete your local branch manually
- Make sure you are not on the branch that you want to delete

```
$ git branch
* add-home-page
  master
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
$ git branch -d add-home-page
Deleted branch add-home-page (was ab9d8be).
$ git branch
* master
$
```

Delete Your Local Branch

- You will need to delete your local branch manually
- Make sure you are not on the branch that you want to delete

```
$ git branch
* add-home-page
  master
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
$ git branch -d add-home-page
Deleted branch add-home-page (was ab9d8be).
$ git branch
* master
$
```

Why does it think we are up-to-date?
(we just changed master via pull request)



Prune Missing Branches

- You can list all branches that can be removed because the remote upstream branch does not exist anymore.
- Removing `--dry-run` will delete the remote branches in your local repo that don't exist anymore on the remote repo

```
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/add-home-page
  remotes/origin/master

$ git remote prune origin --dry-run
Pruning origin
URL: git@github.com:rofrano/nyu-lab-git-intro.git
* [would prune] origin/add-home-page
$
```

Refresh from the new Master

- Don't forget to fetch the latest changes and pull them into master

```
$ git fetch
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From github.com:rofrano/nyu-lab-git-intro
  907cda6..5f7a2de  master      -> origin/master
$
```

Refresh from the new Master

- Switch to the **master** branch so that we can **pull** the changes

```
$ git fetch
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From github.com:rofrano/nyu-lab-git-intro
  907cda6..5f7a2de  master      -> origin/master
$ git checkout master
Switched to branch 'master'
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
$
```

Refresh from the new Master

- Issue a `git pull` to bring down the new code

```
$ git fetch -p
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From github.com:rofrano/nyu-lab-git-intro
  907cda6..5f7a2de  master      -> origin/master
$ git checkout master
Switched to branch 'master'
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
$ git pull
Updating 907cda6..5f7a2de
Fast-forward
  index.css | 0
  index.html| 0
  2 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 index.css
  create mode 100644 index.html
$
```

What About Mistakes?

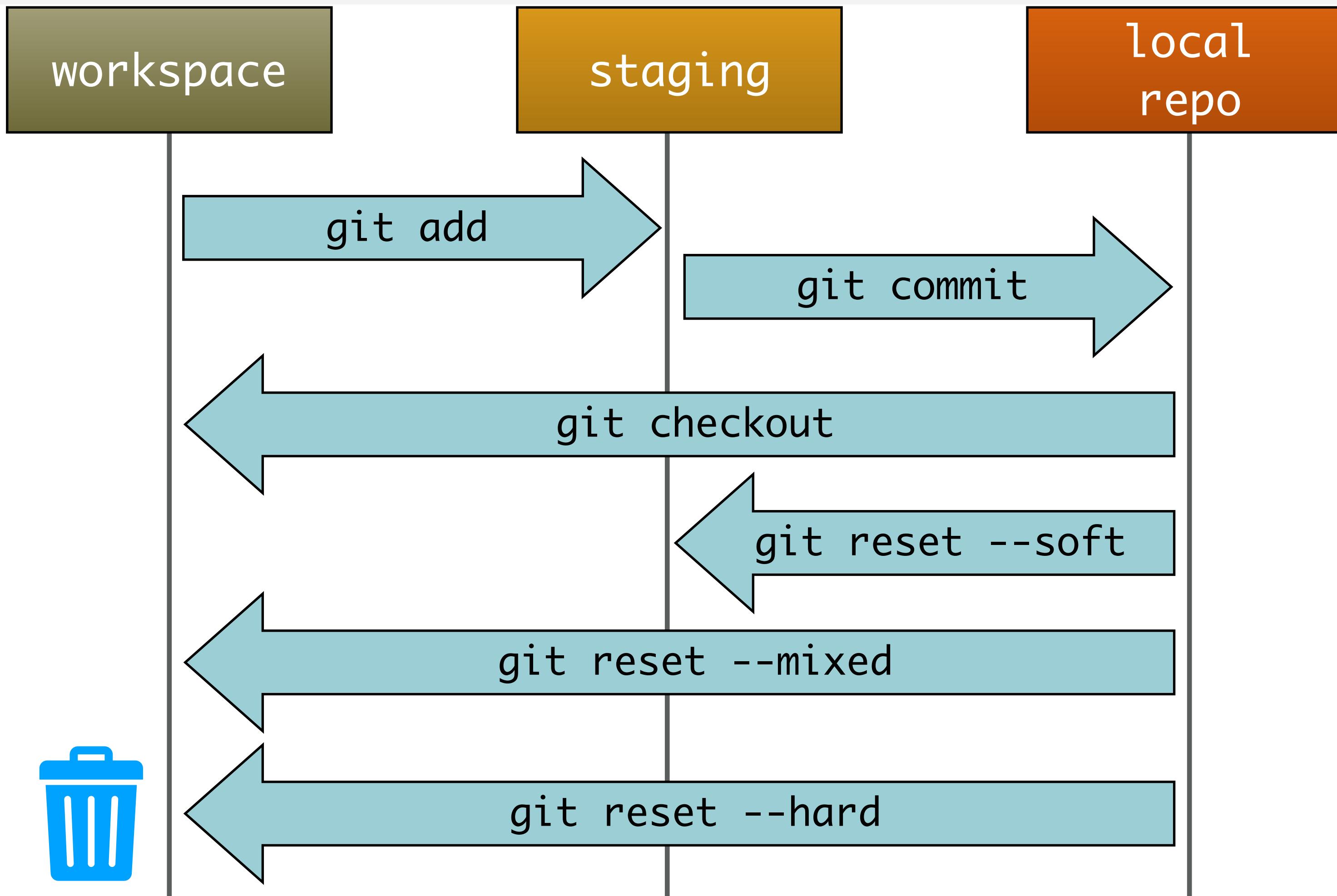
- If you have NOT pushed to a remote branch
- You can use RESET to go back to a previous commit
- Reset rewinds the commits as if they never happened



RESET

- To get rid of the last two commits
 - `git reset --soft HEAD~2`
 - `git reset --hard master@{1}` (undo fast forward merge)
- **--soft** will undo the commit history and leave the files staged
- **--mixed** will undo the history and the staging so the files are untracked (default)
- **--hard** will undo and delete your files!!!

Summary of reset Movements



REVERT

- If you made a commit and pushed it to the remote, how to do undo it because other people may have access to the remote?
 - `git revert <sha1>`
- Revert is the only *safe* undo for a commit that has a remote push
- Note: revert will not remove files that are pushed remotely



Reflog and Revert

- You can use reflog to see your activities and revert them
- Reflog shows the history of all the commands issued in git

```
$ git reflog
ab9d8be HEAD@{0}: commit: added home page and style sheet
907cda6 HEAD@{1}: checkout: moving from master to add-home-page
907cda6 HEAD@{2}: clone: from git@github.com:rofrano/nyu-lab-git-intro.git
$
```

How Reset and Revert are Different

- **Reset** will alter history as if it never existed
 - This will confuse remote repos so once you push a commit you should not reset it
- **Revert** will undo history by adding a commit that reverses the changes
 - This is remote repo friendly



Fun Facts about this Class...

- You will not be working alone in GitHub
- Someone on your team will edit the same file that you do
- There will be merge conflicts!
- These are not the conflicts you're looking for
- There is no fear...
- There is only the FORCE



Merge and Conflicts

- Git can usually figure out how to merge code as long as different lines are changed
- Git cannot merge code that has the same line of code changed in two different branches
- This requires manual intervention and probably collaboration with the developer who made the other changes



Conflict Scenario

- All it takes to create a merge conflict is for two branches to change the same line in a common file
- For this scenario we will change the file `index.html`
- One developer is working on a story to make the index say 'hello'
- Another developer is working on a story to make the index say 'welcome'

Let's Create Some Conflicts!



✓ Create a local git repo called merge-conflict

```
$ mkdir merge-conflict
$ cd merge-conflict
$ git init
Initialized empty Git repository in /home/vagrant/merge-conflict/.git/
$ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
$
```

Add A File

✓ Add an index.html file and commit the change as 'add index'

```
$ touch index.html
$ git add index.html
$ git commit -m 'add index'
[master (root-commit) 1e796aa] add index
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
$ git reflog
1e796aa HEAD@{0}: commit (initial): add index
$ git log --oneline --decorate --all --graph
* 1e796aa (HEAD -> master) add index
$ git status
On branch master
nothing to commit, working directory clean
$
```

Create the Hello Branch

✓ Create a branch called 'hello' and edit the 1st line of index.html and commit

```
$ git checkout -b hello
Switched to a new branch 'hello'
$ echo 'Hello' > index.html
$ cat index.html
Hello
$ git status
On branch hello
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
$ git commit -am 'added hello to index'
[Hello 92b526a] added hello to index
  1 file changed, 1 insertion(+)
$
```

Create the Welcome Branch

- ✓ Switch back to master
- ✓ Create a branch called 'welcome' and edit the 1st line of index.html and commit

```
$ git checkout master
Switched to branch 'master'
$ cat index.html
$ git checkout -b welcome
Switched to a new branch 'welcome'
$ echo 'Welcome' > index.html
$ cat index.html
Welcome
$ git commit -am 'added welcome'
[welcome 5e8197b] added welcome
 1 file changed, 1 insertion(+)
$
```

Where Are We At?

✓ Let's see where we're at with log and reflog

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> welcome) added welcome
| * 92b526a (hello) added hello to index
|/
* 1e796aa (master) add index

$ git reflog
5e8197b HEAD@{0}: commit: added welcome
1e796aa HEAD@{1}: checkout: moving from master to welcome
1e796aa HEAD@{2}: checkout: moving from hello to master
92b526a HEAD@{3}: commit: added hello to index
1e796aa HEAD@{4}: checkout: moving from master to hello
1e796aa HEAD@{5}: commit (initial): add index
```

Customize Log Command

- You can create a nice looking log command my using git aliases

```
git config --global alias.lg "log --oneline --decorate --all --graph"
```

- Or a really fancy one with formatting

```
git config --global alias.lg "log --decorate --all --graph --  
pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold  
blue)<%an>%Creset'"
```

- Then just use git lg to use it

Let's Simulate a Pull Request

✓ To simulate a pull request lets merge welcome into master

```
$ git checkout master
Switched to branch 'master'
$ cat index.html
$ git merge welcome
Updating 1e796aa..5e8197b
Fast-forward
 index.html | 1 +
 1 file changed, 1 insertion(+)

$ cat index.html
Welcome
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> master, welcome) added welcome
| * 92b526a (hello) added hello to index
|/
* 1e796aa add index
$
```

How Did The Log Change?

Before

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> welcome) added welcome
| * 92b526a (hello) added hello to index
|
* 1e796aa (master) add index
```

After

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> master, welcome) added welcome
| * 92b526a (hello) added hello to index
|
* 1e796aa add index
$
```

- Notice that master simply moved forward to join welcome
- The commit sha1 is the same as welcome (5e8197b)
- This is called a fast-forward



How Did The Log Change?

Before

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> welcome) added welcome
| * 92b526a (hello) added hello to index
|
* 1e796aa (master) add index
```

After

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> master, welcome) added welcome
+ * 92b526a (hello) added hello to index
|
* 1e796aa add index
$
```

- Notice that master simply moved forward to join welcome
- The commit sha1 is the same as welcome (5e8197b)
- This is called a fast-forward



Simulate Pull Request With Hello

✓ Try to simulate a pull request by merging hello into master

```
$ git status
On branch master
nothing to commit, working directory clean

$ git merge hello
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

$ cat index.html
<<<<<< HEAD
Welcome
=====
Hello
>>>>> hello
$ git merge --abort
```

Simulate Pull Request With Hello

✓ Try to simulate a pull request by merging hello into master

```
$ git status  
On branch master  
nothing to commit, working directory clean  
  
$ git merge hello  
Auto-merging index.html  
CONFLICT (content): Merge conflict in index.html  
Automatic merge failed; fix conflicts and then commit the result.
```

```
$ cat index.html  
<<<<< HEAD  
Welcome  
=====  
Hello  
>>>>> hello  
$ git merge -abort
```

The dreaded merge conflict 🔥

What Happened?

- The `hello` branch and `welcome` branch were both direct descendants of `master` (*this is key*)
- The `hello` branch modified the same line as the `welcome` branch
- Git needs you to tell it which version of the line to use
- *Hint:* Your pull request would have had a conflict



What's A Possible Fix?

- ✓ The owner of hello should merge their work on the current version of master before submitting the pull request

```
$ git checkout hello  
Switched to branch 'hello'
```

```
$ git merge master  
Auto-merging index.html  
CONFLICT (content): Merge conflict in index.html  
Automatic merge failed; fix conflicts and then commit the  
result.
```

```
$ cat index.html  
<<<<< HEAD  
Hello  
=====  
Welcome  
>>>>> master
```

Different Perspectives

- Note how the merge is reflected depending on your current branch

Perspective from master branch

```
$ git checkout master
$ git merge hello
$ cat index.html
<<<<<< HEAD
Welcome
=====
Hello
>>>>> hello
```

Perspective from hello branch

```
$ git checkout hello
$ git merge master
$ cat index.html
<<<<<< HEAD
Hello
=====
Welcome
>>>>> master
```

Different Perspectives

- Note how the merge is reflected depending on your current branch

Perspective from master branch

```
$ git checkout master
$ git merge hello
$ cat index.html
<<<<< HEAD
Welcome
=====
Hello
>>>>> hello
```

Perspective from hello branch

```
$ git checkout hello
$ git merge master
$ cat index.html
<<<<< HEAD
Hello
=====
Welcome
>>>>> master
```

Different Perspectives

- Note how the merge is reflected depending on your current branch

Perspective from master branch

```
$ git checkout master
$ git merge hello
$ cat index.html
<<<<< HEAD
Welcome
=====
Hello
>>>>> hello
```

Perspective from hello branch

```
$ git checkout HEAD always points to
$ git merge master
$ cat index.html
<<<<< HEAD
Hello
=====
Welcome
>>>>> master
```

Resolve The Conflict

- ✓ You must edit the file in conflict and then tell rebase to continue

```
$ echo "Hello and Welcome" > index.html
$ git status
On branch hello
You have unmerged paths.
  (fix conflicts and run "git commit")

    Unmerged paths:
```

(use "git add <file>..." to mark resolution)

both modified: index.html

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

```
$ git commit -am 'fix merge conflict'
[hello 218eb38] fix merge conflict
$
```

Results of the merge

Before

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> master, welcome) added welcome
| * 92b526a (hello) added hello to index
|/
* 1e796aa add index
```

After

```
$ git log --oneline --decorate --all --graph
* 218eb38 (HEAD -> hello) fix merge conflict
|\ \
| * 5e8197b (welcome, master) added welcome
* | 92b526a added hello to index
|/
* 1e796aa add index
```

- Notice that hello is now the HEAD
- We are now ready to merge this back into master

Results of the merge

Before

```
$ git log --oneline --decorate --all --graph
* 5e8197b (HEAD -> master, welcome) added welcome
| * 92b526a (hello) added hello to index
|/
* 1e796aa add index
```

After

```
$ git log --oneline --decorate --all --graph
* 218cb39 (HEAD -> hello) fix merge conflict
|\ 
| * 5e8197b (welcome, master) added welcome
* | 92b526a added hello to index
|/
* 1e796aa add index
```

- Notice that hello is now the HEAD
- We are now ready to merge this back into master

Continue Pull Request Simulation

✓ To simulate a pull request lets merge hello into master

```
$ git checkout master
Switched to branch 'master'

$ cat index.html
Welcome

$ git merge hello
Updating 5e8197b..218eb38
Fast-forward
 index.html | 2 +-+
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git log --oneline --decorate --all --graph
* 218eb38 (HEAD -> master, hello) fix merge conflict
|\ \
| * 5e8197b (welcome) added welcome
* | 92b526a added hello to index
|/
* 1e796aa add index

$ cat index.html
Hello and Welcome
```

View the Log Ref

✓ You can always use git reflog to see all of the work done

```
$ git reflog
218eb38 HEAD@{0}: merge hello: Fast-forward
5e8197b HEAD@{1}: checkout: moving from hello to master
218eb38 HEAD@{2}: commit (merge): fix merge conflict
92b526a HEAD@{3}: checkout: moving from master to hello
5e8197b HEAD@{4}: merge welcome: Fast-forward
1e796aa HEAD@{5}: checkout: moving from welcome to master
5e8197b HEAD@{6}: commit: added welcome
1e796aa HEAD@{7}: checkout: moving from master to welcome
1e796aa HEAD@{8}: checkout: moving from hello to master
92b526a HEAD@{9}: commit: added hello to index
1e796aa HEAD@{10}: checkout: moving from master to hello
1e796aa HEAD@{11}: commit (initial): add index

$
```

How To Abort

- Never leave a merge or rebase in an unfinished state!
- You can abort a merge with:
 - `git merge --abort`
- You can abort a rebase with:
 - `git rebase --abort`



Now that you know Git



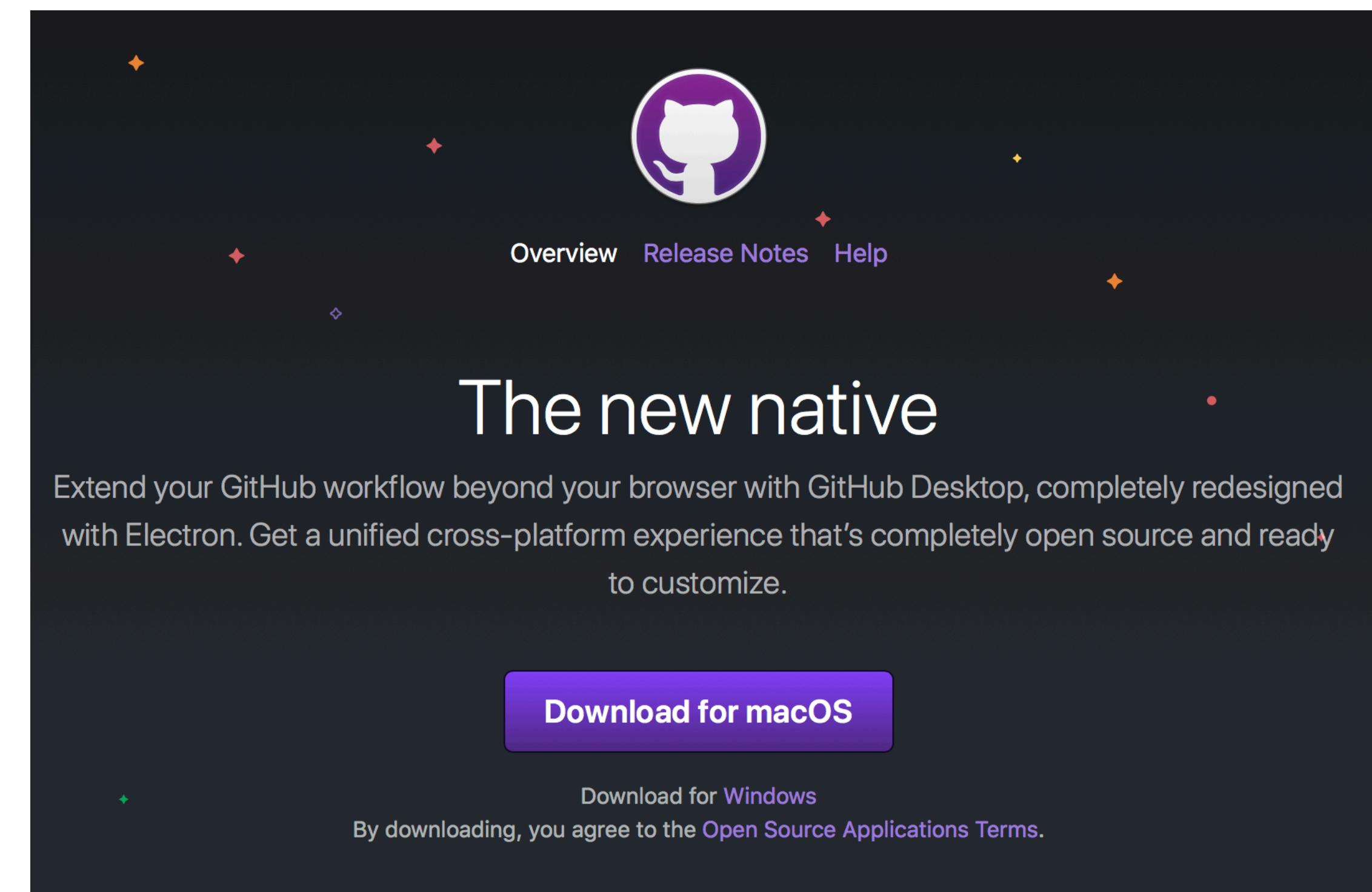
Let's look at GitHub Desktop



GitHub Desktop

<http://desktop.github.com>

- GitHub Desktop is a Graphical User Interface for GitHub
- Makes it easy to contribute to projects on GitHub and GitHub Enterprise
- Integrates with the Atom Editor and Terminal





GitHub Desktop UI

The screenshot shows the GitHub Desktop application interface. At the top, the repository is set to "Current Repository: lab-flask-sqlalchemy" and the "Current Branch" is "master". The status bar indicates "Push origin" was last fetched 12 minutes ago. The main area displays a diff for the file "app/__init__.py". The left sidebar lists "Changes 7" with checkboxes next to each file: "app/__init__.py", "docker-compose.yml", "Dockerfile.db2", "Jenkinsfile", "manage.py", "README.md", and "Vagrantfile". The right pane shows the diff content:

```
@@ -1,4 +1,4 @@
-# Copyright 2016, 2017 John J. Rofrano. All Rights Reserved.
+# Copyright 2016, 2019 John J. Rofrano. All Rights Reserved.

#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
@@ -23,10 +23,11 @@ from flask import Flask
from flask_sqlalchemy import SQLAlchemy
import pymysql

"># These next lines are positional:
+# These next 4 lines are positional:
# 1) We need to create the Flask app
-# 2) Then configure it
+# 2) Then configure it for the database
# 3) Then initialize SQLAlchemy after it has been configured
+# 4) Finally we can import our database models

app = Flask(__name__)
# Load the configuration
```

At the bottom, there is a "Summary (required)" field with a user profile picture, a "Description" text area, a "Commit to master" button, and a note indicating the commit was "Committed Jan 14, 2019 converted from DB2 to MySQL". There is also an "Undo" button.



GitHub Desktop UI

Current branch / Change Branch

The screenshot shows the GitHub Desktop application interface. At the top, there's a navigation bar with a dropdown labeled "Current Branch master". Below the navigation bar is a list of "Changes" showing 7 changed files: app/__init__.py, docker-compose.yml, Dockerfile.db2, Jenkinsfile, manage.py, README.md, and Vagrantfile. The main area displays the contents of app/__init__.py with a diff view. The commit message editor at the bottom includes fields for "Summary (required)" and "Description", and a "Commit to master" button. A status bar at the bottom indicates the commit was made on Jan 14, 2019, and provides an "Undo" option.

Changes

History

app/__init__.py

7 changed files

- ✓ app/__init__.py
- ✓ docker-compose.yml
- ✓ Dockerfile.db2
- ✓ Jenkinsfile
- ✓ manage.py
- ✓ README.md
- ✓ Vagrantfile

Current Branch
master

Push origin
Last fetched 12 minutes ago

Summary (required)

Description

Commit to master

Committed Jan 14, 2019
converted from DB2 to MySQL

Undo



GitHub Desktop UI

The screenshot shows the GitHub Desktop application interface. The top bar displays the current repository as "Current Repository lab-flask-sqlalchemy", the current branch as "Current Branch master", and the status of the last push to "Push origin Last fetched 12 minutes ago".

The main area is divided into two sections: "Changes" (highlighted with a red box) and "History". The "Changes" section shows a list of 7 changed files: app/_init_.py, docker-compose.yml, Dockerfile.db2, Jenkinsfile, manage.py, README.md, and Vagrantfile. Each file entry has a checkbox and a small icon next to it. The "History" section shows the content of the app/_init_.py file, displaying a diff between lines 1 and 30. The diff highlights changes in the code, such as the addition of copyright notices and imports.

At the bottom of the screen, a commit dialog is open. It includes fields for "Summary (required)" (with a placeholder "Summary (required)"), "Description" (with a placeholder "Description"), and a "Commit to master" button. A note at the bottom states "Committed Jan 14, 2019 converted from DB2 to MySQL".

A red callout box with the text "Files to be committed Not all need to be checked" points to the "Changes" tab in the main interface.

```
@@ -1,4 +1,4 @@
-# Copyright 2016, 2017 John J. Rofrano. All Rights Reserved.
+# Copyright 2016, 2019 John J. Rofrano. All Rights Reserved.

1   1
2   2
3   3
4   4
@@ -23,10 +23,11 @@ from flask import Flask
23  23
24  24
25  25
26  26
@@ -27,2 +27,2 @@ # These next lines are positional:
27  27
28  28
@@ -29,2 +29,2 @@ # 1) We need to create the Flask app
29  29
30  30
@@ -31,2 +31,2 @@ # 2) Then configure it
31  31
32  32
@@ -32,2 +32,2 @@ # 3) Then initialize SQLAlchemy after it has been configured
32  33
# 4) Finally we can import our database models
app = Flask(__name__)
# Load the configuration
```



GitHub Desktop UI

The screenshot shows the GitHub Desktop application interface. The top bar displays the current repository as "Current Repository lab-flask-sqlalchemy", the current branch as "Current Branch master", and the status of the push to origin as "Push origin Last fetched 12 minutes ago".

The main area is divided into two sections: "Changes" (7) and "History". The "Changes" section lists seven changed files: app/_init_.py, docker-compose.yml, Dockerfile.db2, Jenkinsfile, manage.py, README.md, and Vagrantfile. The "History" section is currently empty.

A red box highlights the code diff for the file app/_init_.py. The diff shows the following changes:

```
@@ -1,4 +1,4 @@
 1   1  #@ Copyright 2016, 2017 John J. Rofrano. All Rights Reserved.
 2   2 +# Copyright 2016, 2019 John J. Rofrano. All Rights Reserved.
 3   3  #
 4   4  # Licensed under the Apache License, Version 2.0 (the "License");
@@ -23,10 +23,11 @@
 23  23  from flask_sqlalchemy import SQLAlchemy
 24  24  import pymysql
 25  25
 26  26  #-# These next lines are positional:
 27  27 +## These next 4 lines are positional:
 28  28  #-# 1) We need to create the Flask app
 29  29  #-# 2) Then configure it
 30  30 +## 2) Then configure it for the database
 31  31  #-# 3) Then initialize SQLAlchemy after it has been configured
 32  32 +## 4) Finally we can import our database models
 33  33  app = Flask(__name__)
 34  34  # Load the configuration
```

A red callout box with the text "Diff of what changed since last commit" points to the diff area.

The bottom section of the interface includes fields for "Summary (required)" and "Description", a "Commit to master" button, and a note indicating the commit was "Committed Jan 14, 2019 converted from DB2 to MySQL". There is also an "Undo" button.



GitHub Desktop UI

The screenshot shows the GitHub Desktop application interface. At the top, there's a header bar with the GitHub logo, the title "Current Repository lab-flask-sqlalchemy", the current branch "master", and a "Push origin" status. A red box highlights the repository name "lab-flask-sqlalchemy". Below the header is a "Changes" tab showing 7 changed files, with "app/__init__.py" selected. A red arrow points from a callout box containing the text "Select a different repository here" to the repository name in the header. The main pane displays a diff of the file "app/__init__.py", comparing two versions. The bottom part of the window contains a "Summary (required)" section with fields for "Summary" and "Description", and a "Commit to master" button.

Current Repository
lab-flask-sqlalchemy

Current Branch
master

Push origin
Last fetched 12 minutes ago

Changes 7 History app/__init__.py

7 changed files

- ✓ app/__init__.py
- ✓ docker-compose.yml
- ✓ Dockerfile.db2
- ✓ Jenkinsfile
- ✓ manage.py
- ✓ README.md
- ✓ Vagrantfile

from flask_sqlalchemy import SQLAlchemy
import pymysql
These next lines are positional:
1) We need to create the Flask app
2) Then configure it
3) Then initialize SQLAlchemy after it has been configured
4) Finally we can import our database models
app = Flask(__name__)
Load the configuration

Select a different repository here

Summary (required)

Description

Commit to master

Committed Jan 14, 2019
converted from DB2 to MySQL

Undo



Change Repository

The screenshot shows the GitHub desktop application interface. At the top, it displays the current repository as "Current Repository lab-flask-sqlalchemy", the current branch as "Current Branch master", and the push origin status as "Push origin Last fetched 17 minutes ago". The main area shows a diff view of the file "app/_init__.py". A red box highlights the sidebar on the left, which lists repositories under three categories: GitHub.com, Enterprise, and Other. The repository "lab-flask-sqlalchemy" is selected in the GitHub.com section. An orange callout bubble points to this sidebar with the text "List of Repositories".

```
diff --git app/__init__.py app/__init__.py
--- a/app/__init__.py
+++ b/app/__init__.py
@@ -1,4 +1,4 @@
-# Copyright 2016, 2017 John J. Rofrano. All Rights Reserved.
+## Copyright 2016, 2019 John J. Rofrano. All Rights Reserved.
@@ -23,10 +23,11 @@ from flask import Flask
@@ -26,10 +27,11 @@ # These next lines are positional:
+## These next 4 lines are positional:
@@ -27,28 +27,29 @@ # 1) We need to create the Flask app
-# 2) Then configure it
+## 2) Then configure it for the database
@@ -29,29 +29,30 @@ # 3) Then initialize SQLAlchemy after it has been configured
+## 4) Finally we can import our database models
@@ -31,32 +31,32 @@ app = Flask(__name__)
@@ -32,32 +32,32 @@ # Load the configuration
```



GitHub Desktop UI

The screenshot shows the GitHub Desktop application interface. At the top, there are three dropdown menus: "Current Repository" set to "lab-flask-sqlalchemy", "Current Branch" set to "master", and "Push origin" with a status message "Last fetched 12 minutes ago". Below these is a toolbar with a "Changes" tab (selected), a "History" tab, and a search bar. The main area displays a diff view for the file "app/__init__.py". The left pane lists 7 changed files: app/__init__.py, docker-compose.yml, Dockerfile.db2, Jenkinsfile, manage.py, README.md, and Vagrantfile. The right pane shows the code changes, with lines 1 through 33 highlighted in blue and pink/green. A red box highlights the bottom-left corner of the application, which contains a modal dialog titled "Summary (required)". The dialog has a "Description" field and a "Commit to master" button. A large orange arrow points from the text "Commit message and submit" to this dialog. At the bottom of the main window, a status bar shows "Committed Jan 14, 2019 converted from DB2 to MySQL" and "Undo".

Commit message and submit

Summary (required)

Description

Commit to master

Committed Jan 14, 2019
converted from DB2 to MySQL

Undo



GitHub Desktop UI

The screenshot shows the GitHub Desktop application interface. At the top, it displays the current repository as "Current Repository lab-flask-sqlalchemy" and the current branch as "Current Branch master". A red box highlights the status bar message "Push origin Last fetched 12 minutes ago". Below this, the main window shows a list of "Changes" (7 changed files) and a diff view for the file "app/__init__.py". A large red arrow points from the "Push to GitHub" button to the status bar message. In the bottom-left corner, a modal dialog box is open, also highlighted with a red box. It contains fields for "Summary (required)" (with a profile picture placeholder), "Description" (with a text area and a plus icon), and a "Commit to master" button. A second large red arrow points from the "Commit message and submit" text to the "Commit to master" button.

Push origin
Last fetched 12 minutes ago

Push to GitHub

Commit message and submit

Commit to master

Summary (required)

Description

Commit to master

Committed Jan 14, 2019
converted from DB2 to MySQL

Undo



History of Commits

The screenshot shows a GitHub commit history interface. At the top, there are tabs for 'Changes' (7) and 'History'. The 'History' tab is highlighted with a red box and an arrow pointing to it from the left. The main area displays a list of commits:

- converted from DB2 to MySQL (John Rofrano committed Jan 14, 2019)
- removed need for json library (John Rofrano committed Jan 14, 2019)
- made name and category_id not nullable (John Rofrano committed Jan 14, 2019)
- updated backend introducing separate Cate... (John Rofrano committed Dec 9, 2018)
- fixed logging problem (John Rofrano committed Dec 7, 2018)
- fixed booleans being passed as strings (John Rofrano committed Dec 4, 2018)
- updated Flask version (John Rofrano committed Nov 25, 2018)
- Modified for DB2 and PostgreSQL (John Rofrano committed Nov 25, 2018)
- added manage.py to create database and ta... (John Rofrano committed Apr 12, 2018)
- added database create to Vagrantfile (John Rofrano committed Nov 18, 2017)
- Added Procfile back to use with honcho (John Rofrano committed Nov 18, 2017)
- added xml reports to .gitignore (John Rofrano committed Nov 18, 2017)
- updated README.md with vcap_services.py (John Rofrano committed Nov 18, 2017)

To the right of the commit list, a detailed view of the last commit is shown, with the title 'History of commits' overlaid in a large orange box. The commit details show a diff of changes:

```
@@ -0,0 +1,102 @@
+## Local environment
++.DS_Store
+Thumbs.db
+
+## Text reports
+unittests.xml
+
+## Vagrant
+.vagrant
+
+## Ignore database files
+db/*.db
+
+## Byte-compiled / optimized / DLL files
+__pycache__/
+*.py[cod]
+$py.class
+
+## C extensions
+*.so
+
+## Distribution / packaging
+.Python
+env/
+build/
+develop-eggs/
+dist/
+downloads/
+eggs/
+.eggs/
+lib/
```

What Have We Covered?

The basic workflow for cloning, branching, pushing, and pull requests

How to merge, handle merge conflicts, and rebase

How to use status, logs, and reflog to see where you are at



Additional Reading

- GitHub Learning Resources
 - <https://github.github.io/on-demand/>
- Git documentation
 - <https://git-scm.com/docs/>
- Git Feature Branch Workflow:
 - <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>