

A photograph of three young adults sitting around a table, looking at their laptops. A woman on the left wears glasses and a purple shirt, a man in the center wears a grey t-shirt, and a man on the right wears glasses and a dark t-shirt with a logo. They appear to be in a workshop or classroom setting.

# How to Collaborate with GitHub

Workshop

MLH localhost

GitHub



Roshan



Harish

We are your Hosts !



***Our Mission is to Empower Hackers.***

**65,000+**  
HACKERS

**12,000+**  
PROJECTS CREATED

**3,000+**  
SCHOOLS

*We hope you learn something awesome today!*  
*Find more resources: <http://mlh.io/>*

*Using your Web Browser,  
Open this URL & Fill out the Form:*

**<http://mlhlocal.host/checkin>**

**5**

**Which MLH Localhost  
Activity are you  
participating in today?**

**HOW TO COLLABORATE  
WITH GITHUB**

**6**

**Which school is hosting  
this event?**

**NIT ROURKELA**

*Then, check your email for code, slides, and more!*



# Audience Q&A Session

Head over to [slido.com](https://slido.com)  
And use the event code #62335

- ① Start presenting to display the audience questions on this slide.

# What will you **learn today?**

1

The basics of Git and GitHub

2

The GitHub Workflow

# Table of Contents

- 
1. Introduction to Git and GitHub
  2. GitHub Collaboration WorkFlow
  3. GitHub and the Command Line
  4. Review & Quiz
  5. Next Steps

**We're all going to collaborate on a  
project together.**

**How?**

**Version control!**

# Problem Statement

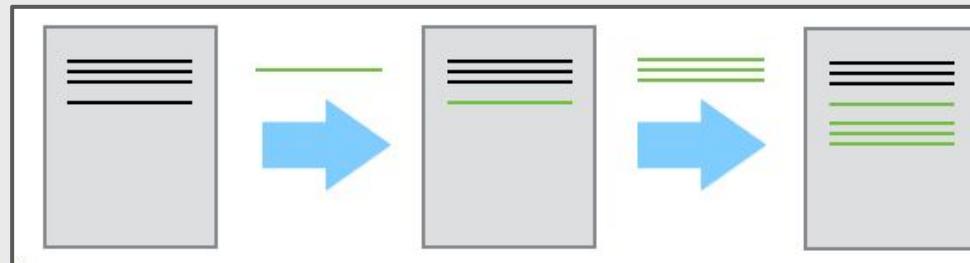
- 1. Handling lots of edits in a single file**
- 2. Saving multiple copies of a file**

# Why version control?

Because emailing a file around is painful

## Version control features:

- Log changes in a searchable way, instead of renaming a file for each version.



- Collaborators can work in parallel and merge their changes automatically, instead of manually comparing the differences between a file



# Benefits of Version Control System

- Can explore the differences easily
- Can annotate each small change.
- Can combine each collaborators' contribution easily
- Can experiment with new features

# Git vs. GitHub

What's the difference?

## Git

- Git is one of the version control system.
- It can be used with various tools or locally on your computer to help you keep track of changes in your code projects.
- Think of it like Google Docs for code, but better.



## GitHub

- GitHub is a platform for code collaboration!
- GitHub uses Git for version control and provides you all sorts of awesome collaboration tools.

# GitHub

# Let's make an account!

Visit the URL on the left if you're a student and on the right if you are not. Follow the instructions on GitHub.com to make your account!

<http://mlhlocal.host/github-edu>

<http://mlhlocal.host/github-signup>

The screenshot shows the GitHub Education landing page. At the top, there are navigation links for "Students", "Teachers", "Schools", and "Events". Below these, there is a large image of a yellow backpack with a GitHub logo on it. The text "GitHub Education" is displayed above the backpack. At the bottom of the page, there is a call-to-action button with the text "Learn to ship software like a pro." and a link to "Student Developer Pack".

The screenshot shows the GitHub sign-up page. At the top right, there is a "Sign up" button and a menu icon. The main headline reads "Built for developers". Below the headline, there is a paragraph of text: "GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers." A large input field is visible at the bottom, labeled "Username" with the placeholder "Pick a username".

# Explore GitHub

The GitHub platform provides numerous features for collaboration.

A screenshot of the GitHub homepage. At the top, there is a navigation bar with links for 'Search or jump to...', 'Pull requests' (which is highlighted with a red box), 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, there is a sidebar on the left with sections for 'Repositories' (listing 'dsc-nitr' and a 'New' button) and 'Your teams' (noting 'You don't belong to any teams yet!'). The main content area shows two follow notifications: 'Sachin2Dehury started following you 16 days ago' (with a 'Follow' button) and 'swagat-das started following you on Jul 10' (with a 'Follow' button). A pink box highlights a 'GitHub Sponsors for Organizations' announcement: 'Your organization can now receive sponsorships through GitHub Sponsors! Learn more.' with a 'Go to Sponsors' button. At the bottom right, there are download links for 'GitHub for mobile' (available on App Store and Google Play).

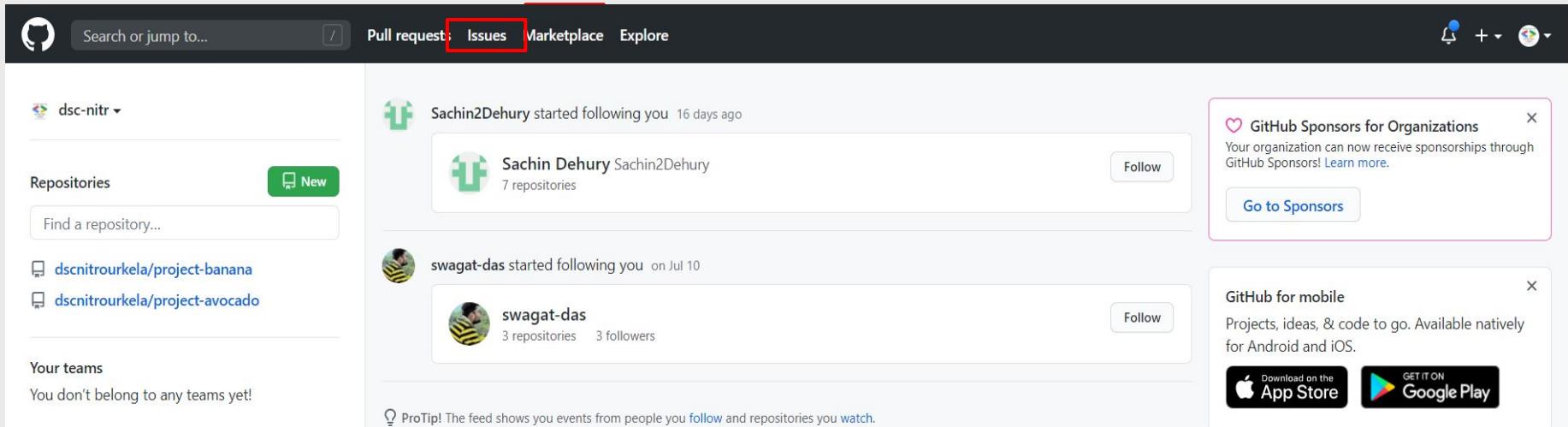
Next to the search bar, you can select Pull requests. This will show you your own pull requests against other people's repos.

## Key Term

**pull request:** a way to ask to make changes to someone else's code by submitting your own changes for their review

# Explore GitHub

The GitHub platform provides numerous features for collaboration.



A screenshot of the GitHub interface focusing on the Issues tab. The top navigation bar includes links for Pull requests, Issues (which is highlighted with a red box), Marketplace, and Explore. On the left, there's a sidebar for the user 'dsc-nitr' with sections for Repositories (one new repository available), Your teams (none), and a search bar for finding repositories. The main content area shows two notifications: one from 'Sachin2Dehury' stating they started following the user, and another from 'swagat-das' stating they started following the user. Both notifications include profile pictures, names, follower counts, and a 'Follow' button. A pink callout box highlights the 'GitHub Sponsors for Organizations' feature, which allows organizations to receive sponsorships. At the bottom right, there's an advertisement for 'GitHub for mobile' with download links for the App Store and Google Play.

Next to pull requests, you can see any issues you've opened or worked on.

## Key Term

**issues:** a way to share a problem about someone else's code, without necessarily submitting your own solution

# Explore GitHub

The GitHub platform provides numerous features for collaboration.

A screenshot of the GitHub homepage. At the top, there's a navigation bar with a search bar, 'Pull requests', 'Issues', 'Marketplace', and 'Explore' buttons. In the upper right corner, there's a user icon with a red box around it, a '+' button for creating a new project, and a gear icon for settings. The main content area shows two notifications: one from 'Sachin2Dehury' and another from 'swagat-das'. Both notifications include a 'Follow' button. A 'GitHub Sponsors for Organizations' card is also visible. On the left, there's a sidebar for the user 'dsc-nitr' with sections for 'Repositories' (including 'New'), 'Find a repository...', and 'Your teams' (noting they don't belong to any teams yet). At the bottom, there's a 'ProTip!' message and download links for 'GitHub for mobile' available on the App Store and Google Play.

- In the upper right hand corner, you can see any notifications you have received.
- The + symbol allows you to create a new project.
- Clicking your avatar opens the settings menu.

# Explore a Repository

Navigate to the URL below and let's check out the repository!

dscnitrourkela / project-banana

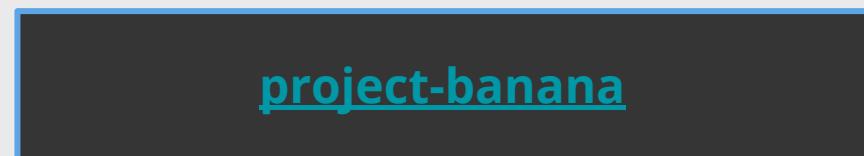
Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 15 branches 0 tags Go to file Add file Code

File	Description	Last Commit
.github/workflows	Create main.yml	4 days ago
public	all set for github workshop	4 days ago
.firebaserc	Init multiple domain for test	11 months ago
.gitignore	Init website	11 months ago
.travis.yml	Update .travis.yml	11 months ago
README.md	Update README.md	4 days ago
firebase.json	v1	9 months ago

README.md

project-banana



## Key Term

**repository**: think of this like a project folder where code is stored!

# Explore a Repository

The first tab in a repository is the **Code** tab. It shows the code in the project.

A screenshot of a GitHub repository page for 'dscnitrourkela / project-banana'. The 'Code' tab is highlighted with a red box. The page displays a list of commits and files. At the top, there are buttons for 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below these are buttons for 'Go to file', 'Add file', and 'Code'. The commit list shows the following entries:

Commit	Message	Date	Commits
HarishTeens	Update README.md	20fd08b 4 days ago	63 commits
.github/workflows	Create main.yml	4 days ago	
public	all set for github workshop	4 days ago	
.firebaserc	Init multiple domain for test	11 months ago	
.gitignore	Init website	11 months ago	
.travis.yml	Update .travis.yml	11 months ago	
README.md	Update README.md	4 days ago	
firebase.json	v1	9 months ago	

Below the commit list is a preview of the 'README.md' file, which contains the text 'project-banana'.

# Explore a Repository

The second tab is the [Issues](#) tab. If there is a problem with the code in someone's project, you can open an Issue and let them know about it. Pro-tip: You should always follow the project maintainer's instructions for opening an issue.

A screenshot of the GitHub Issues page. The top navigation bar includes tabs for Issues (which is highlighted with a red box), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A modal window titled "Label issues and pull requests for new contributors" is displayed, explaining that GitHub will help potential first-time contributors discover issues labeled with "good first issue". Below the modal, there are filters for "is:issue is:open", a search bar, and buttons for "Labels 9", "Milestones 0", and "New issue". At the bottom, there are filters for "Open" (0) and "Closed" (0) issues, and dropdown menus for Author, Label, Projects, Milestones, Assignee, and Sort. The main content area displays a large exclamation mark icon and the text "There aren't any open issues." with a sub-instruction: "You could search [all of GitHub](#) or try an [advanced search](#)." A footer note says "ProTip! Ears burning? Get @dsc-nitr mentions with [mentions:dsc-nitr](#)."

# Explore a Repository

The third tab is the **Pull Requests** tab. If you want to contribute code to someone's repository, you'll open a Pull Request. You'll do that today!

The screenshot shows the GitHub repository interface with the 'Pull requests' tab selected. A red box highlights the 'Pull requests' tab in the navigation bar. The main content area displays a message: 'Label issues and pull requests for new contributors' with a 'Dismiss' button, followed by a note: 'Now, GitHub will help potential first-time contributors discover issues labeled with [good first issue](#)'. Below this is a search bar with filters ('Filters', 'is:pr is:open'), a label count ('Labels 9'), a milestone count ('Milestones 0'), and a 'New pull request' button. Underneath, there are filters for 'Open' (0) and 'Closed' (23) pull requests, and dropdown menus for 'Author', 'Label', 'Projects', 'Milestones', 'Reviews', 'Assignee', and 'Sort'. The central message states: 'There aren't any open pull requests.' with a small icon above it. A pro tip at the bottom suggests: 'ProTip! Exclude your own issues with -author:dsc-nitr.'

**There is a lot of terminology regarding  
Git and GitHub. Let's get started!**

# Table of Contents

1. Introduction to Git and GitHub
2. GitHub Collaboration WorkFlow
3. GitHub and the Command Line
4. Review & Quiz
5. Next Steps

# Developer Workflow

You're going to practice a typical best practice developer workflow.

1. First, you'll **fork** someone else's code. This means creating your own copy of it.
2. Then you'll create a **branch**. The branch is a parallel version of your copy, where you'll test your own changes.

## Key Terms

**fork:** your own copy of someone else's repository.

**branch:** a parallel version of the master copy of a repo. Making a branch allows you to edit code without accidentally breaking a working version

# Developer Workflow

3. You **stage** your changes as you go. When you're happy with them, you'll **commit** them.
4. If you want to add your code to someone else's project, you'll open a **pull request**.
5. If they approve it, they'll **merge** your branch into their master branch.

## Key Terms

**stage**: add to a cohesive group/bundle of revisions

**commit**: a group of revisions you want to officially add to your branch

**merge**: to officially add the changes from your branch into the master branch (or another branch)

**Great! That's it. That's how you work with  
Git. Let's try it using GitHub!**

**HANDS-ON SESSION**

# What You'll Be developing

<https://dscnitrourkela.tech/submissions/submissions.html>

Developer Student Clubs  
NIT Rourkela

HOME ABOUT US PROJECTS EVENTS TEAM CONTACT SUBMIT AN IDEA!

## Submissions



Shaswat Lenka  
AI | Software Design ❤️  
Creative Coding and Problem Solving



Harish  
Aspiring Programmer



Abel Mathew  
It doesn't matter how beautiful your idea is. If it doesn't see the light of execution, it's pointless



Smarak Das  
An amateur app developer, coding and tech enthusiast.  
Flutter ❤️.



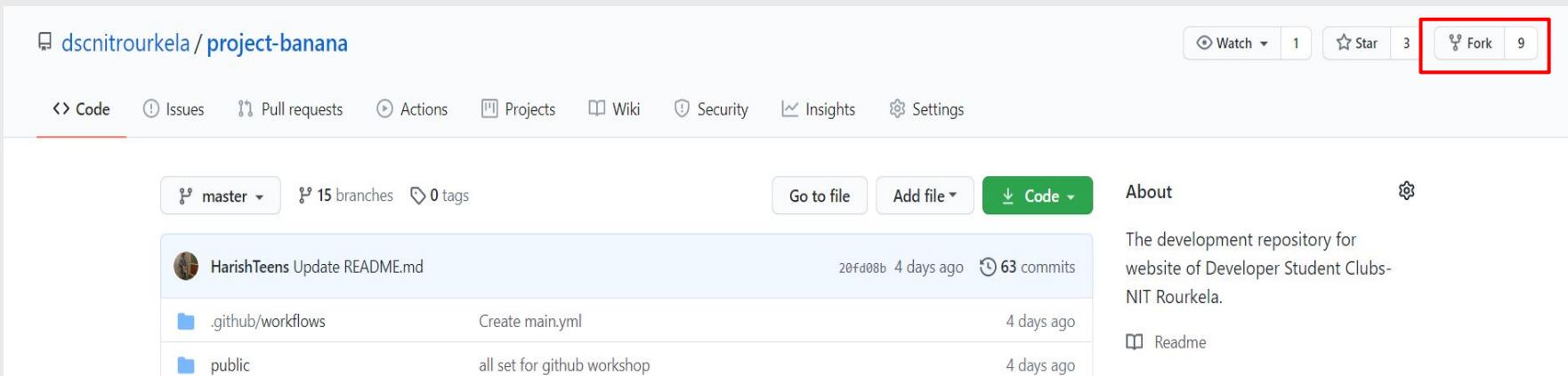
# Activity Workflow

As you can see, there are so many way to use GitHub! Today we're going to use a very common developer workflow.

Go to the project repository-

**<https://github.com/dscnitrourkela/project-banana>**

1. **Fork** the repository you want to contribute to.



The screenshot shows the GitHub repository page for `dscnitrourkela/project-banana`. At the top right, there are buttons for Watch (1), Star (3), and Fork (9). The Fork button is highlighted with a red box. Below the header, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Code tab is selected. On the left, there's a dropdown for the master branch, showing 15 branches and 0 tags. The main area displays a list of recent commits:

Commit	Author	Message	Date	Commits
HarishTeens Update README.md	HarishTeens	Update README.md	20fd08b 4 days ago	63 commits
.github/workflows		Create main.yml	4 days ago	
public		all set for github workshop	4 days ago	

To the right, there's an About section with the following text: "The development repository for website of Developer Student Clubs-NIT Rourkela." Below it is a Readme link.

# Activity Workflow

2. Go to the **Release branch** then Create a **branch**. You should always try to name branches with your name and what you're doing, or follow the conventions set by your team.

This screenshot shows the GitHub repository page for 'dscnitrourkela/project-banana'. The 'Code' tab is selected. A dropdown menu is open over the 'release' button, which is highlighted with a red border. The menu shows '17 branches' and '0 tags'. Below the menu, a message states 'This branch is 12 commits ahead of master.' There are two pull requests listed: 'HarishTeens Update README.md' and 'public'. A 'Create main.yml' button is also visible.

This screenshot shows the 'Switch branches/tags' modal for the same GitHub repository. The input field for creating a new branch is highlighted with a blue border and contains the text 'first\_branch'. Below the input field are tabs for 'Branches' and 'Tags'. At the bottom of the modal, there is a button labeled 'Create branch: first\_branch from 'release'' and a link to 'View all branches'.

# Activity Workflow

## 3. Open profiles.js

A screenshot of a GitHub repository page for 'project-banana / public / submissions /'. The top navigation bar shows 'release' and the repository path. Below the header, it says 'This branch is 2 commits ahead of master.' with 'Pull request' and 'Compare' buttons. A commit by 'HarishTeens' titled 'revert move profile.js' is shown, with three files listed: 'profiles.js', 'submissions.html', and 'submissions.js'. The 'profiles.js' file is highlighted with a red box around its icon.

File	Commit Message	Time Ago
profiles.js	revert move profile.js	3 hours ago
submissions.html	revert move profile.js	3 hours ago
submissions.js	all set for github workshop	5 days ago

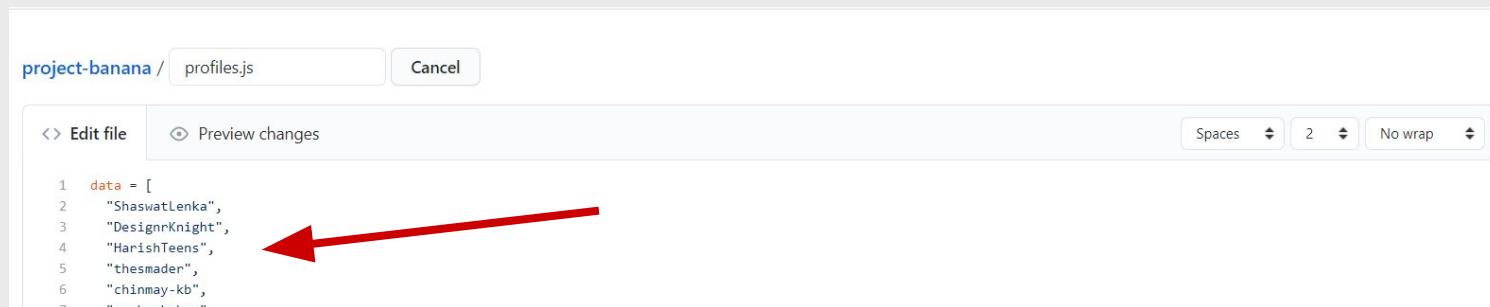
## 4. Select the edit icon. It looks like a pencil.

A screenshot of the GitHub file editor for 'project-banana / profiles.js'. The top navigation bar shows 'release' and the file path. The commit history for this file is shown, with one commit by 'HarishTeens' titled 'move profile to root folder' with a green checkmark. Below the commit, it says '1 contributor'. At the bottom, it shows '10 lines (10 sloc) | 142 Bytes' and the first few lines of the code. On the right side, there are buttons for 'Raw', 'Blame', a copy icon, an edit icon (a pencil), and a delete icon. The edit icon is highlighted with a red box.

```
1 data = [
```

# Activity Workflow

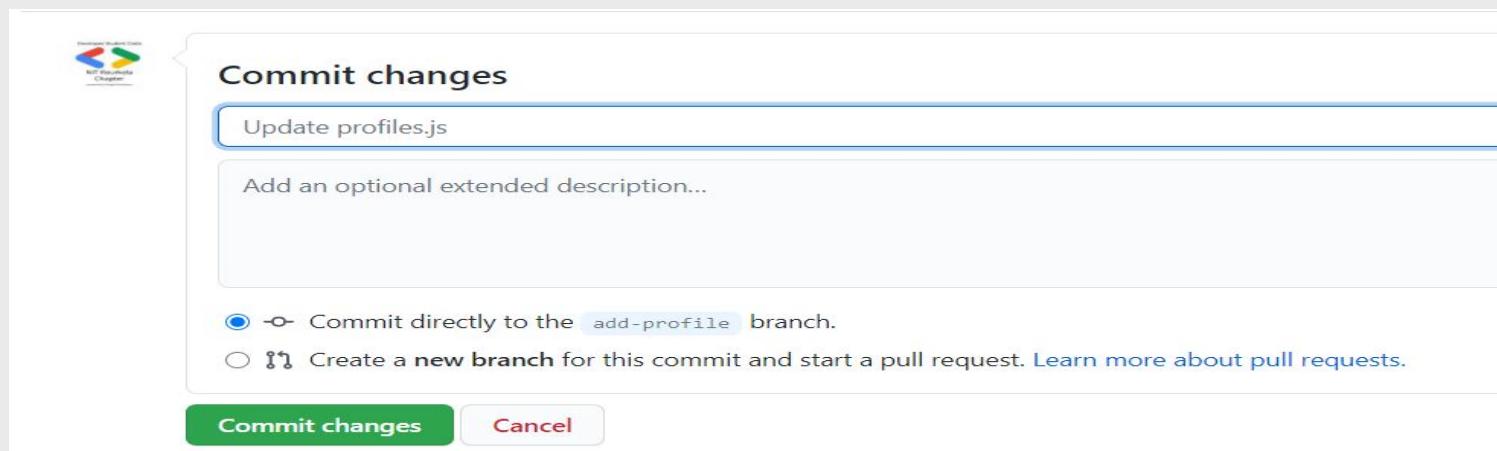
5. Add your profile name. Choose a random blank line!



A screenshot of a GitHub code editor window for a file named 'profiles.js'. The code contains an array of strings representing profile names. A red arrow points to the line containing the string 'thesmader'.

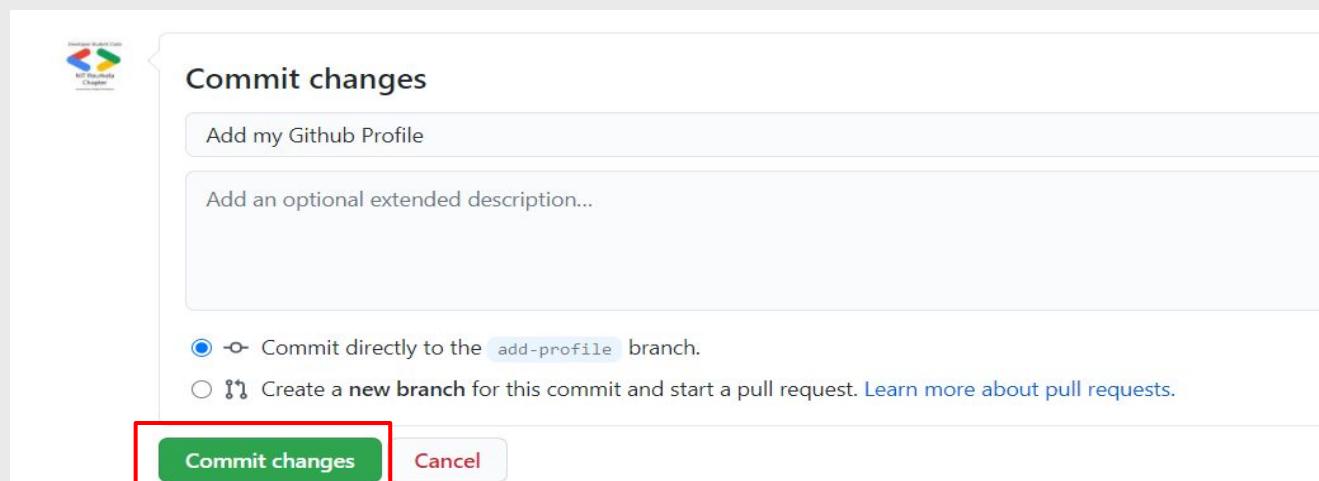
```
project-banana / profiles.js Cancel  
Edit file Preview changes Spaces 2 No wrap  
1 data = [  
2 "ShaswatLenka",  
3 "DesignerKnight",  
4 "HarishTeens",  
5 "thesmader",  
6 "chinmay-kb",  
7 "narenkaran"
```

6. Scroll to the bottom to the Commit changes section.



# Activity Workflow

7. Add a descriptive message. Select **Commit changes**.



# Activity Workflow

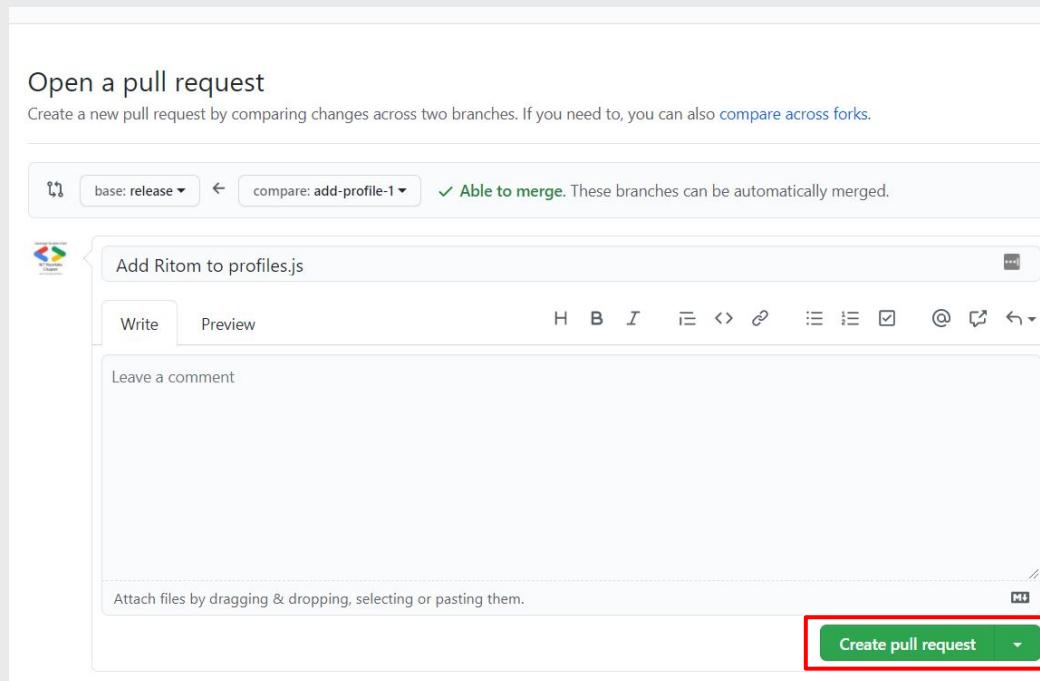
8. Return to the main page of your repo.
9. Select **Compare and pull request**.

The screenshot shows a GitHub repository interface. At the top, there's a yellow banner with the message: "add-profile-1 had recent pushes less than a minute ago". To the right of this message is a green button with the text "Compare & pull request", which is highlighted with a red border. Below the banner, there are navigation links: "master" (with a dropdown arrow), "17 branches" (with a dropdown arrow), "0 tags" (with a dropdown arrow), "Go to file", "Add file", and "Code". The main area displays two recent commits:

- A commit by "dsc-nitr" titled "Update README.md" with a timestamp of "34 minutes ago" and "77 commits".
- A commit by "Create main.yml" with a timestamp of "4 days ago".

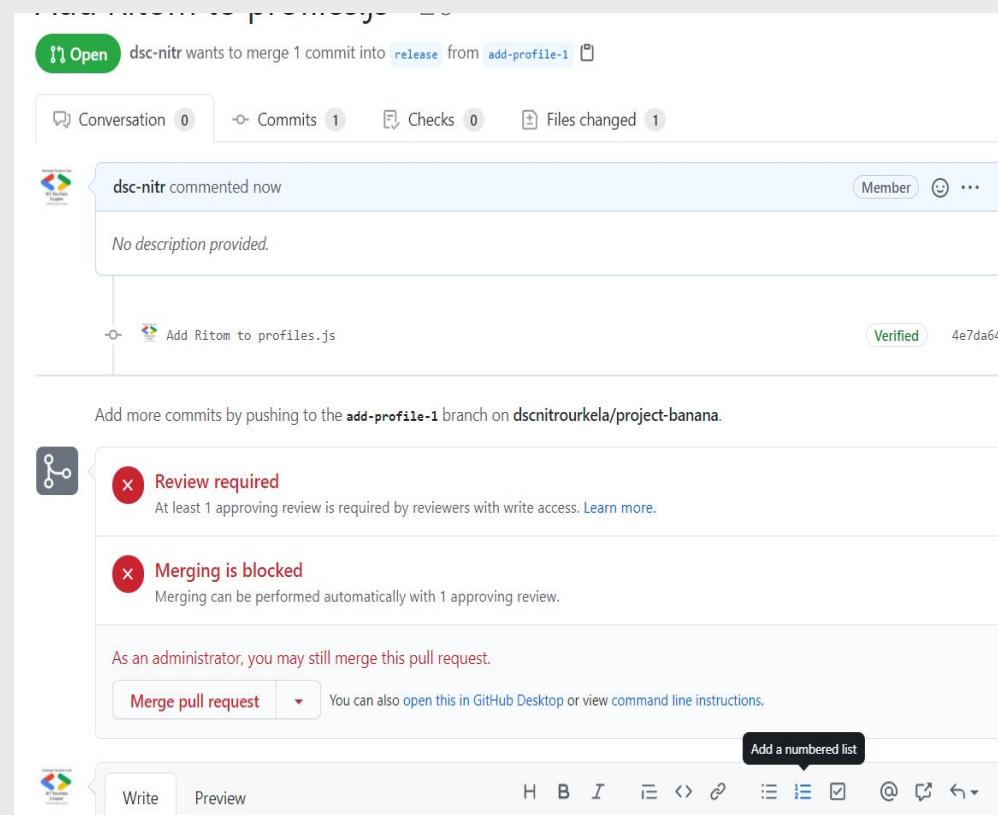
# Activity Workflow

- Because this is a simple pull request, you can simply select [Create Pull Request](#). If you were contributing to an open source project, you would want to be very descriptive about the changes you made.



# Activity Workflow

11. Wait for your pull request to be reviewed by a reviewer. The reviewer is usually the owner or a maintainer of the repository.



# Activity Workflow

12. Ask someone else to comment on your PR. Comment on someone else's PR, too!

You can find other pull requests by selecting the Pull requests tab on the repo's home page.

The screenshot shows a GitHub pull request titled "Add my hometown to locations.txt #1". The pull request has 1 commit and 1 file changed. A comment from "mlh-localhost" is visible, stating "mlh-localhost commented 3 minutes ago" and "No description provided.". Below the comment is the commit message "mlh Add my hometown to locations.txt". A green checkmark indicates "This branch has no conflicts with the base branch". A reply to this comment is shown, with the message "Looks good to me!" and a "Comment" button at the bottom right.

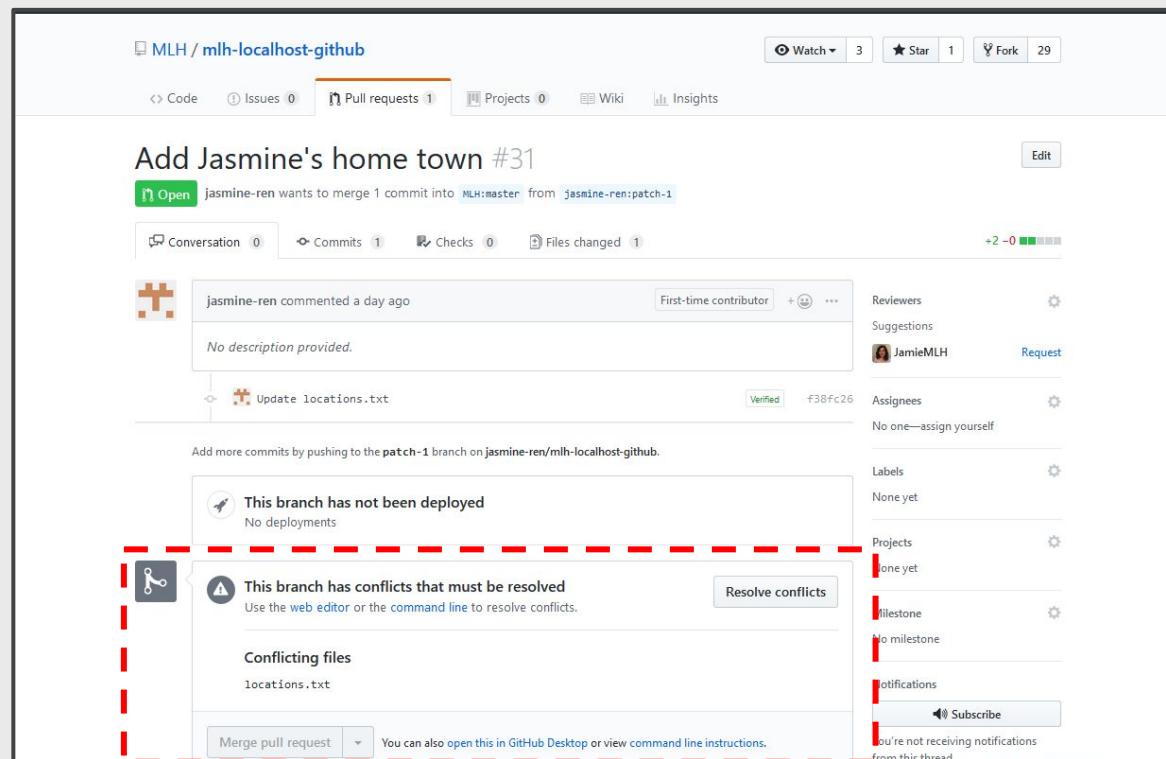
# Activity Workflow

13. This is what a pull request looks like for someone who has Write access to a repo. The organizer of this workshop will click Merge pull request, and your Profile will be added!

The screenshot shows a GitHub pull request interface. At the top, it says "dsc-nitr requested a review from HarishTeens 2 minutes ago". Below that, "HarishTeens approved these changes now" with a "View changes" button. A comment from "HarishTeens" is shown: "left a comment" and "all good here". Below the comments, a message says "Add more commits by pushing to the `add-profile-1` branch on `dscnitrourkela/project-banana`". The main review summary is highlighted with a green border and a red box around the "Merge pull request" button. It includes sections: "Changes approved" (1 approving review by reviewers with write access), "1 approval", and "This branch has no conflicts with the base branch" (Merging can be performed automatically). The "Merge pull request" button is at the bottom.

# Conflict Resolution

When a file has multiple edits, it can be unclear which change should be committed. This is a conflict and must be resolved before merging.



# Conflict Resolution

A conflict is marked by

<<<<< new branch

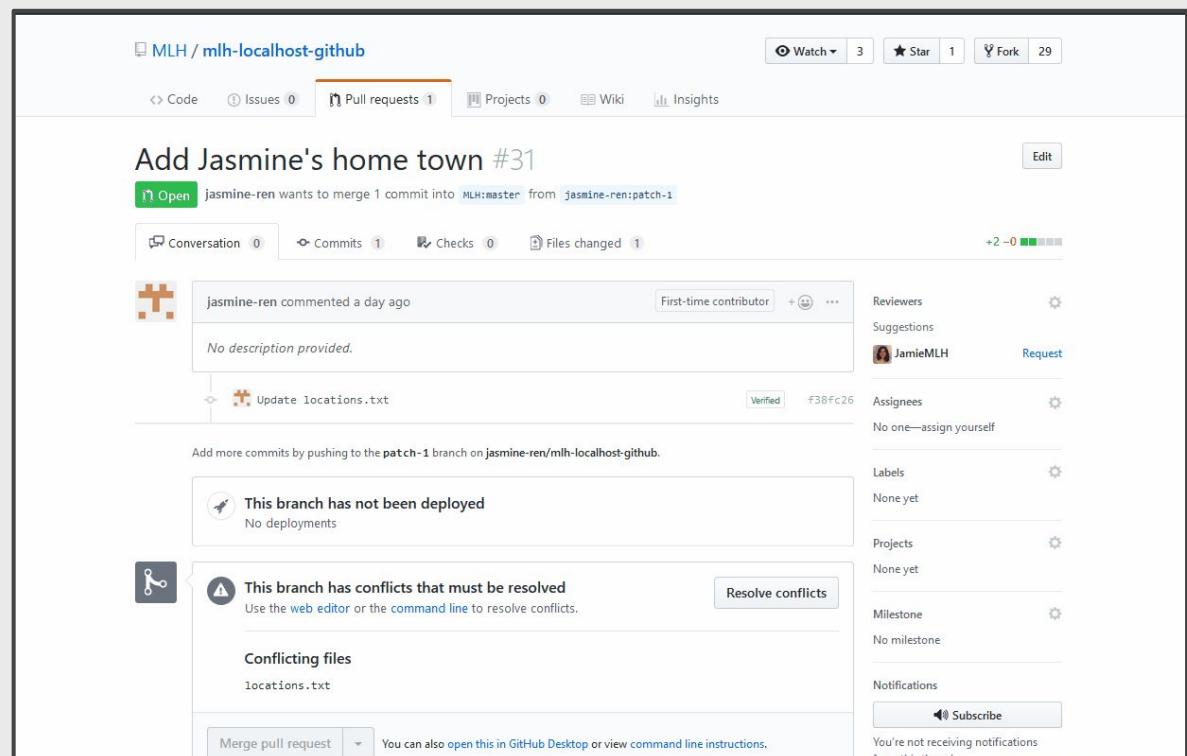
changes

=====

original

>>>>> original-branch

Edit the file to the desired state, mark it resolved and then merge it.



# Activity Workflow

Refresh the original webpage and wait for more people's hometowns to be added! **Note:** this web site is set up to auto-deploy when new code is merged to the repository, but it can take a few minutes.

Developer Student Clubs  
NIT Rourkela

HOME ABOUT US PROJECTS EVENTS TEAM CONTACT SUBMIT AN IDEA

## Submissions



Shaswat Lenka  
AI | Software Design ❤️  
Creative Coding and Problem Solving



Harish  
Aspiring Programmer



Abel Mathew  
It doesn't matter how beautiful your idea is. If it doesn't see the light of execution, it's pointless



Smarak Das  
An amateur app developer, coding and tech enthusiast.  
Flutter ❤️.



**Amazing! Now you know how to work with GitHub in the browser. The next section will teach you how to use GitHub on the command line on your computer!**

# **What did you learn today?**

We created a fun quiz to test your knowledge and see what you learned from this workshop.

**<http://mlhlocal.host/quiz>**

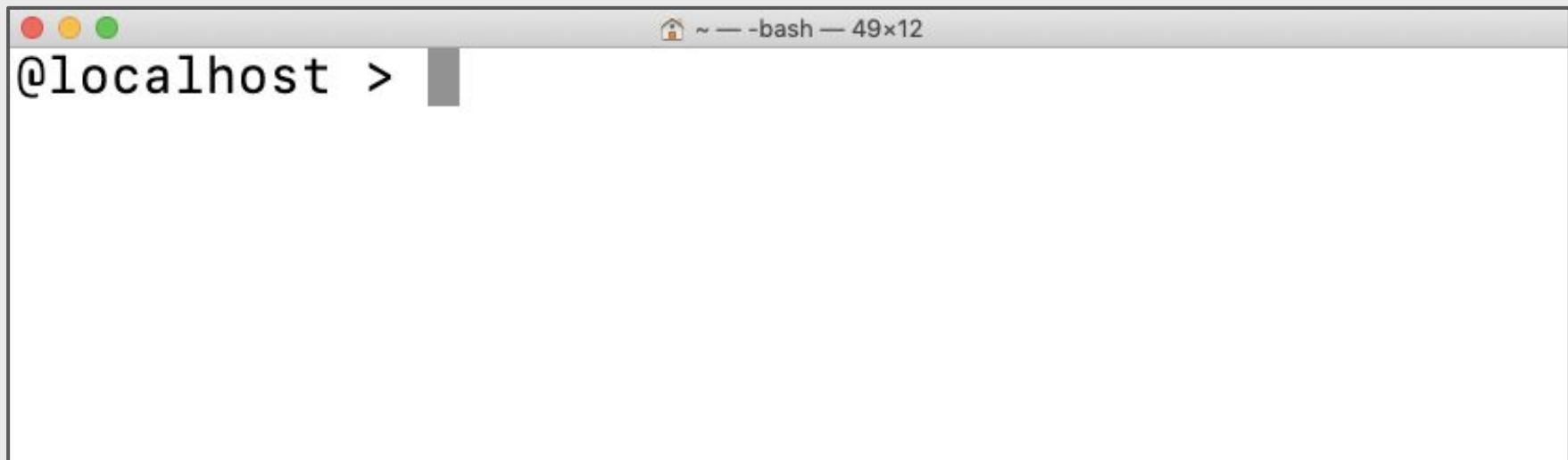
# Table of Contents

- 1. Introduction to Git and GitHub
- 2. GitHub Collaboration WorkFlow
- 3. GitHub and the Command Line
- 4. Review & Quiz
- 5. Next Steps



# GitHub From the Command Line

Once you start working on projects that are more complex, you might find that you prefer using your own local coding environment with GitHub. Let's learn how to set that up! We'll also learn how our local environment works with our repos on GitHub.



```
@localhost >
```

# Install Git

1. First, let's see if you have Git already installed on your computer. Type the command below in Terminal, PowerShell, or any terminal application.

If you already have git installed, you can skip the next few slides!



```
~ - bash - 49x12
@localhost > git --version
git version 2.17.2 (Apple Git-113)
@localhost >
```

A screenshot of a macOS Terminal window. The window title bar shows three colored dots (red, yellow, green) and the text "~- bash ~ 49x12". The main pane contains the command "git --version" followed by its output: "git version 2.17.2 (Apple Git-113)". Below the output, the prompt "@localhost >" is visible. The background of the slide features a faint watermark of the GitHub logo.

# Install Git

<http://mlhlocal.host/install-git>

2. If you do not have git installed, navigate to the URL above.
3. Select your operating system and the installer will download.

## Downloads



Mac OS X



Windows



Linux/Unix

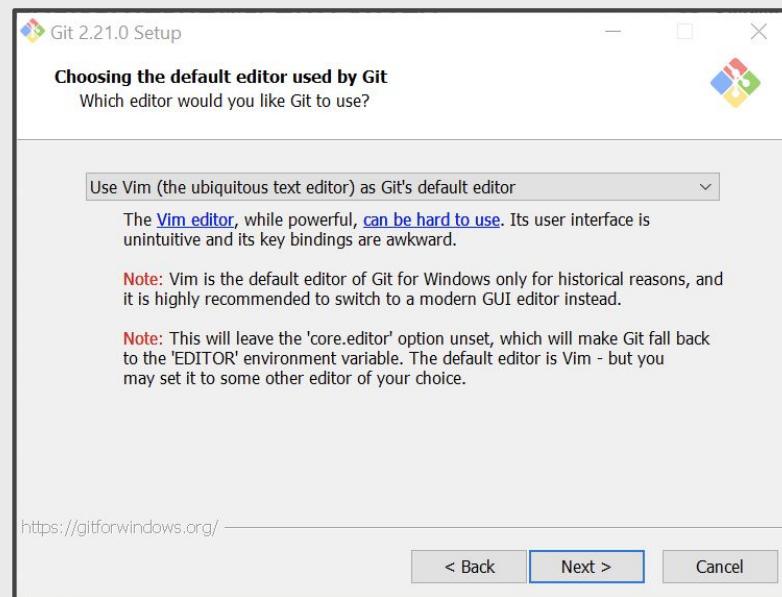
# Install Git: MacOS

4. Open the installer and follow the instructions. It should be fine to select all of the default options.



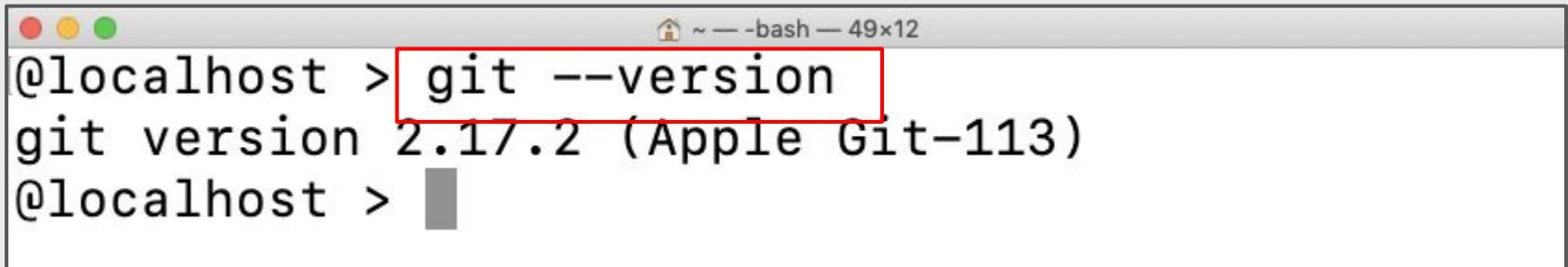
# Install Git: Windows

4. Open the installer and follow the instructions. It should be fine to select all of the default options, except for the one below. We recommend choosing your preferred text editor in this step.



# Install Git

5. Restart your terminal, PowerShell, or Git Bash.
6. Run the command below again. Raise your hand if this is not what you see.



```
~ -bash - 49x12
@localhost > git --version
git version 2.17.2 (Apple Git-113)
@localhost >
```

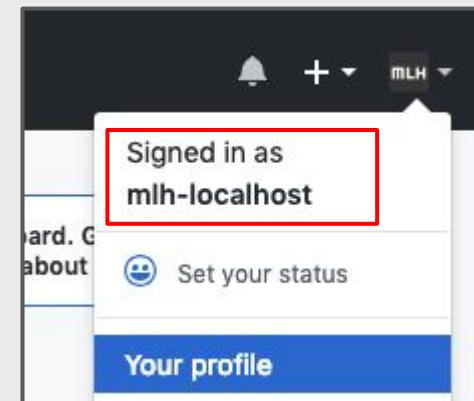
A screenshot of a macOS terminal window. The window title bar shows the standard red, yellow, and green close buttons, followed by the path '~ - bash - 49x12'. The main pane contains a command-line session. The command 'git --version' is typed at the prompt and is highlighted with a red rectangle. The output of the command, 'git version 2.17.2 (Apple Git-113)', is displayed below it. The terminal window has a light gray background and a dark gray sidebar on the right.

**Great! Now you have Git installed. Let's connect your local environment with your GitHub account.**

# Configure Git

We want to configure our local environments so that the correct GitHub account is associated with our commits.

1. On GitHub, find your user name. You can find it by clicking your avatar in the upper right hand corner. It will say "Signed in as"



# Configure Git

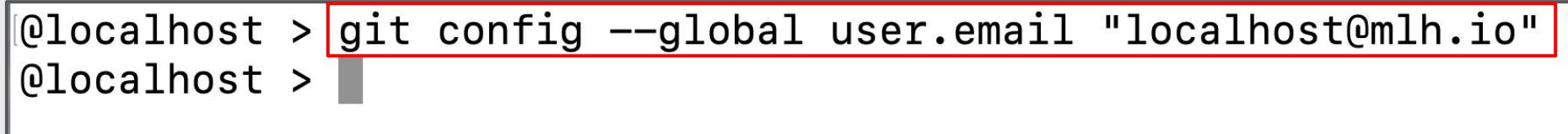
2. Return to your command line or terminal. Replace `mlh-localhost` with your username.



```
localhost ~ -- bash -- 63x13
@localhost > git config --global user.name "mlh-localhost"
@localhost >
```

A screenshot of a macOS terminal window titled "localhost ~ -- bash -- 63x13". The window shows two lines of text input: "@localhost > git config --global user.name "mlh-localhost"" and "@localhost >". The first line is highlighted with a red rectangle.

3. Use the command below to configure your email address as well.



```
localhost > git config --global user.email "localhost@mlh.io"
localhost >
```

A screenshot of a macOS terminal window showing the command to set the global user email. The command "git config --global user.email "localhost@mlh.io"" is highlighted with a red rectangle.

# Configure Git

4. You can use the two commands below to double check that you've set this up.

```
@localhost > git config user.name  
mlh-localhost  
@localhost > git config user.email  
localhost@mlh.io  
@localhost >
```

**Now, let's cover some useful commands.**

# Individual Workflow

The workflow from the command line has a few added steps compared to using GitHub.

1. You can **clone** any repository you have access to using `git clone`.
2. Then, you create a branch with `git branch` "name-of-branch".
3. After you make changes, you'll use `git add` to **stage** your changes.

## Key Terms

**clone**: to copy a repository to your computer

**stage**: saving your changes so they are ready to be added to your branch

# Individual Workflow

The workflow from the command line has a few added steps compared to using GitHub.

4. You'll use `git commit -m "explain your changes"` to add changes to your branch.
5. When you're ready to **push** your local changes to your repository on GitHub, you'll use `git push`.

## Command Explanation

`-m` is a flag for message. That means that whatever comes after `-m` is a message explaining your commit. Your commit message doesn't have any effect on your code; it's like a comment.

## Key Terms

**push**: add the changes on your computer to your GitHub repository.

# Individual Workflow

The workflow from the command line has a few added steps compared to using GitHub.

6. You can make a pull request and have your PR merged on GitHub like before.
7. You can pull changes from GitHub to your local repository.

## Key Terms

**pull**: Bring changes from GitHub down to your local copy.

You might feel a little confused by committing, staging, and pushing your changes. Let's explore a visual representation!

# Three phases of Git

There are three main phases of Git. You can think of these like a work desk!

## Working directory

Where the writing happens. This is when you are coding and saving your work.



## Repository

When you are happy with your commits, you commit them to the repository. These are like final drafts stored more permanently in a drawer

## Staging area

When you are happy with a group of changes and use `git add`, you are staging your changes. These are like called commits, and they are like rough drafts stored in a bundle.

**Here's another visual representation of  
what you just saw.**

# Three phases of Git

Use the staging area to build a commit

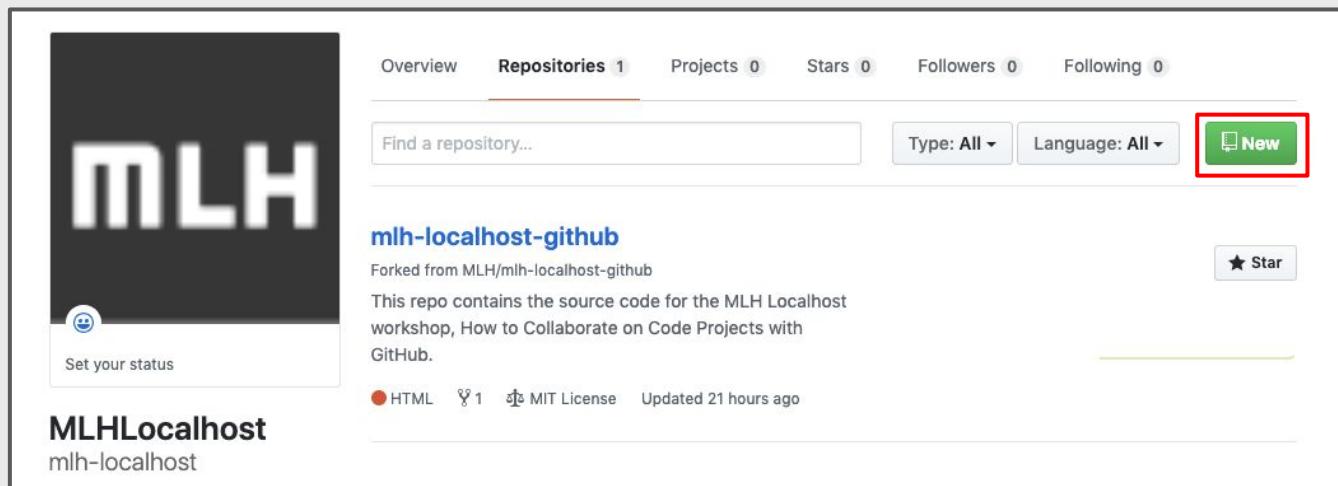


**Great! Let's try it.**

# Create a Repo

Let's create a new repository on GitHub and work with it locally.

1. From your profile or homepage on GitHub, click New.



# Create a Repo

2. Name your repository  
(repo) whatever you want!  
We chose `hello-world`.
3. Write a sentence  
describing your repo.
4. Select **Initialize this  
repository with a  
README**.

Create a new repository

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

Owner mlh localhost / Repository name \* hello-world ✓

Great repository names are short and memorable. Need inspiration? How about [solid-succotash](#)?

Description (optional)  
This repository contains my first GitHub project!

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

# Create a Repo

5. If this were a code project, you would usually add a .gitignore and/or a license but you don't need to.
6. Click **Create repository**.

Create a new repository  
A repository contains all project files, including the revision history.

Owner  / Repository name \*  

Great repository names are short and memorable. Need inspiration? How about [solid-succotash](#)?

Description (optional)

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

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

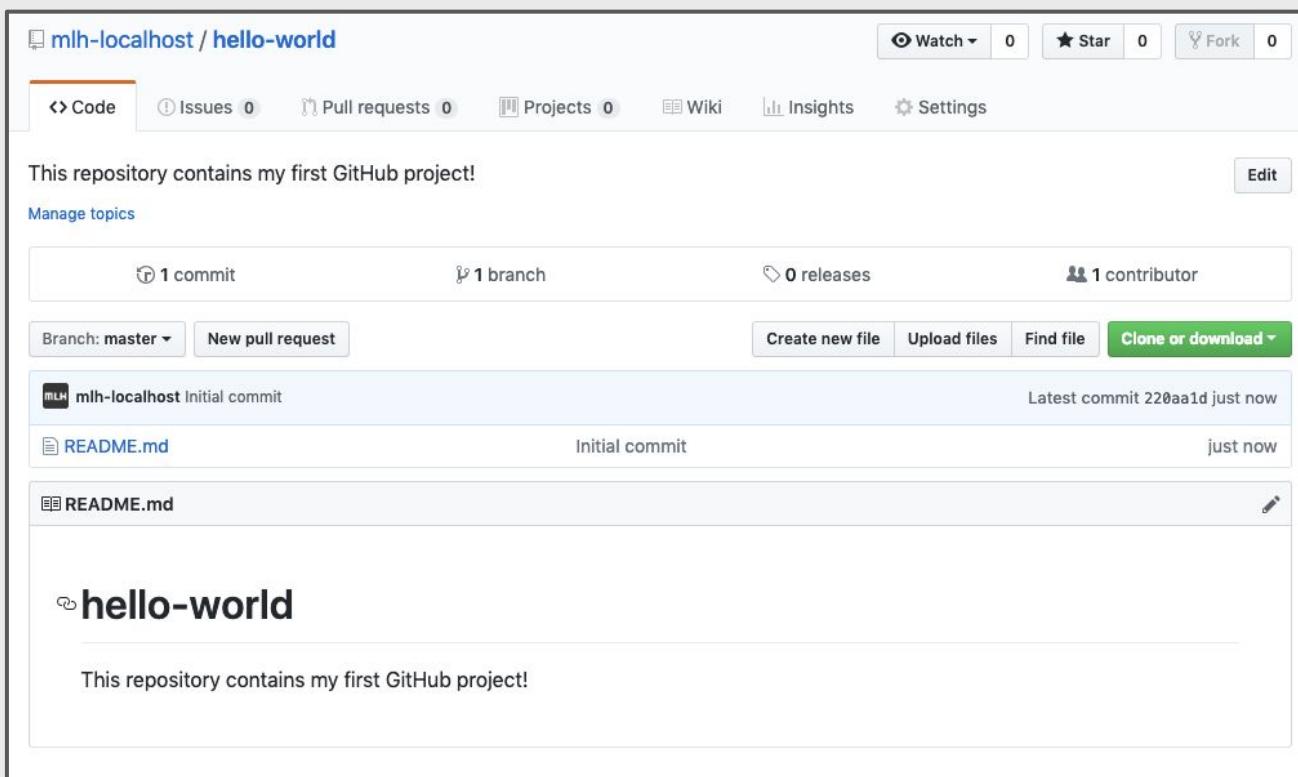
Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore:  Add a license:  

**Create repository**

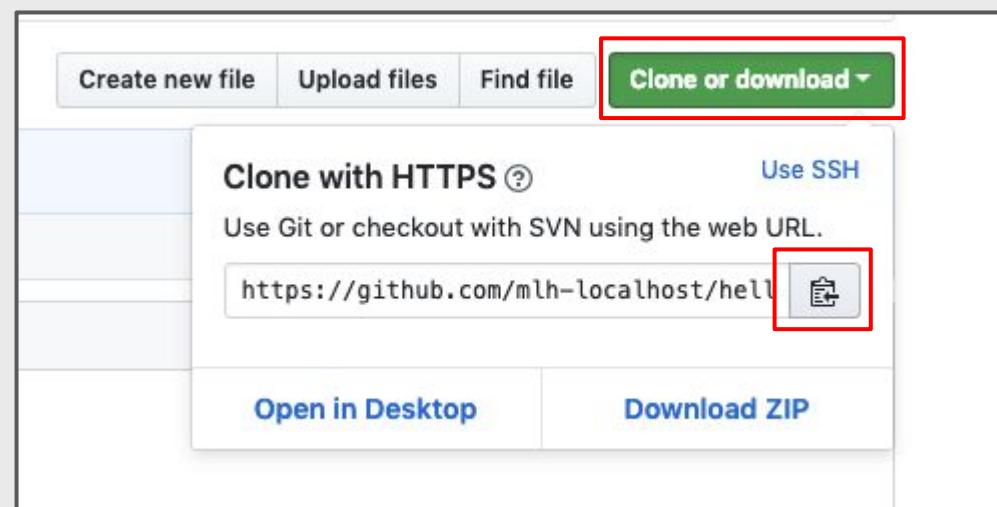
# Create a Repo

7. You will be redirected to the homepage for your new repository!



# Clone Your Repo

8. Click **Clone or download**.
9. Then click the clipboard symbol to copy the address of your repo.



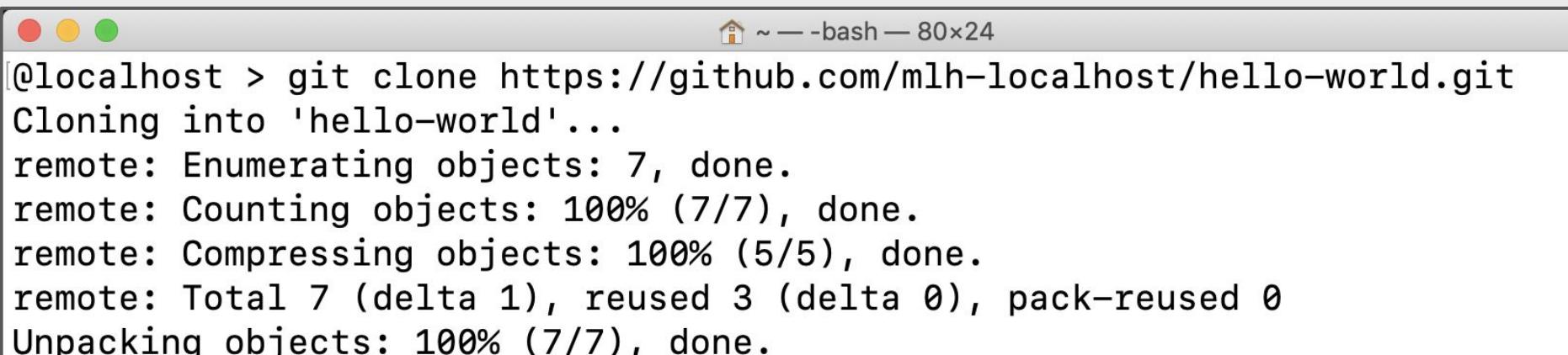
# Clone Your Repo

10. Back in the terminal, type `git clone` then paste the address of your repo.



```
localhost ~ -bash — 80x24
@localhost > git clone https://github.com/mlh-localhost/hello-world.git
```

You will see output like this:



```
localhost ~ -bash — 80x24
@localhost > git clone https://github.com/mlh-localhost/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 1), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
```

# Explore Your Repo

It's important to note that you did not simply download the code. You cloned a Git repo. A git repo comes with a special folder called `.git`. You don't need to know a lot about this folder. Just keep in mind that it connects your local project to the project on GitHub.



The screenshot shows a terminal window with the following session:

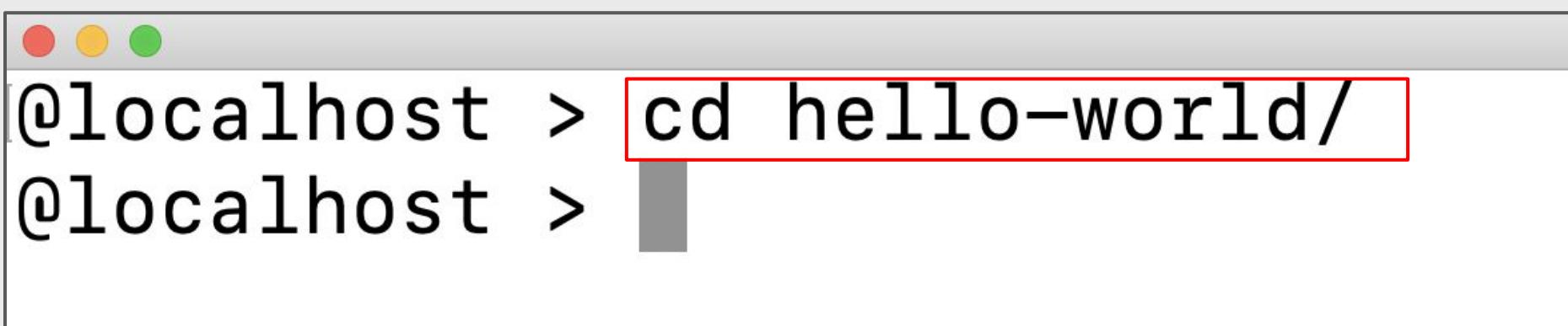
```
[@localhost ~] ~/hello-world — bash — 80x24
[@localhost > cd hello-world/
[@localhost > ls -a
.                         ..
[@localhost > .git          README.md
```

A red box highlights the `.git` folder in the directory listing, indicating its significance as the connection point to the GitHub repository.

# Let's make a change!

1. Change directory into the repo you cloned by typing the command below.

**Note:** If you did not name your repo hello-world, you should substitute hello-world with the name of your repo.

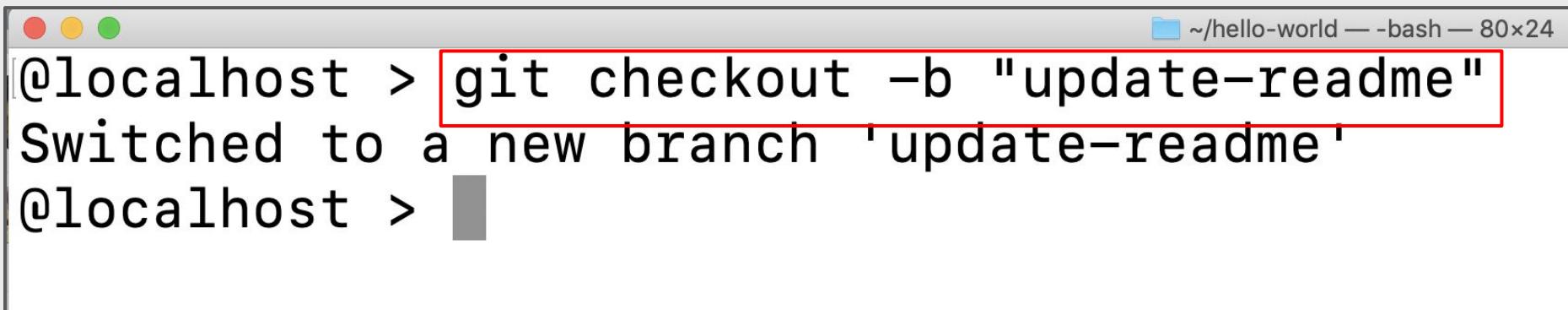


A screenshot of a terminal window on a Mac OS X system, indicated by the red, yellow, and green window control buttons at the top. The terminal is displaying a command-line interface. The first line shows the prompt '@localhost >' followed by the command 'cd hello-world/'. A red rectangular box highlights the entire command 'cd hello-world/'. The second line shows another prompt '@localhost >' followed by a gray vertical bar representing the cursor or the end of the line.

```
@localhost > cd hello-world/
@localhost >
```

# Let's make a change!

2. We're going to make some changes. You never want to commit directly to master, so let's checkout a new branch. You did this in the browser before. Use the command below to create a new branch.

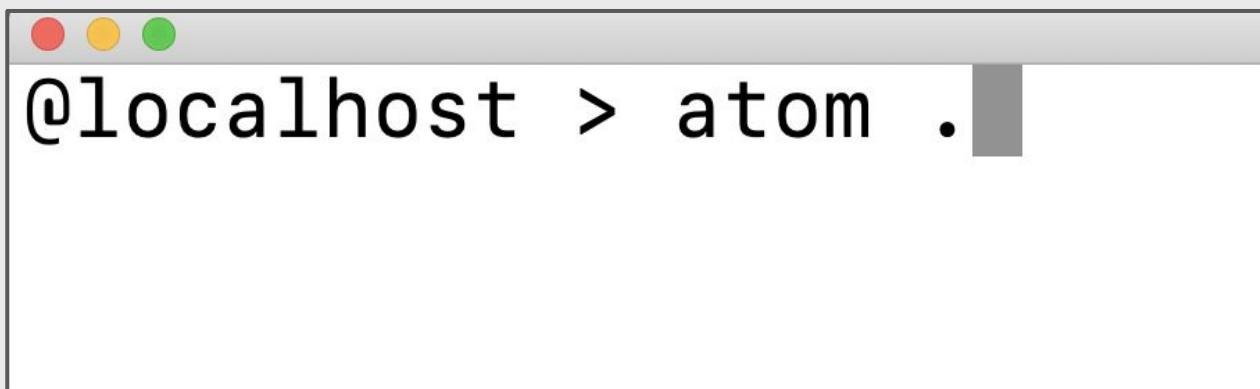


A screenshot of a macOS terminal window titled '~/.hello-world — -bash — 80x24'. The window shows the command 'git checkout -b "update-readme"' highlighted with a red rectangle. The output of the command 'Switched to a new branch 'update-readme'' is also visible. The terminal has its characteristic red, yellow, and green window control buttons at the top left.

```
@localhost > git checkout -b "update-readme"
Switched to a new branch 'update-readme'
@localhost >
```

# Let's make a change!

3. Open your project folder in the code editor of your choice. For example, if you have Atom installed, you can use the command below to open the project in Atom.



# Let's make a change!

4. Make any changes you want to your README and save them.

*README.md*

```
1 # hello-world
2 This repository contains my first GitHub project!
3
4 ## MLH Localhost
5
6 MLH Localhost empowers you to develop your local tech community!
7
```

# Let's make a change!

5. Return to your command line. Type the first command below. You should see output like this.

```
@localhost > git status
On branch update-readme
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)

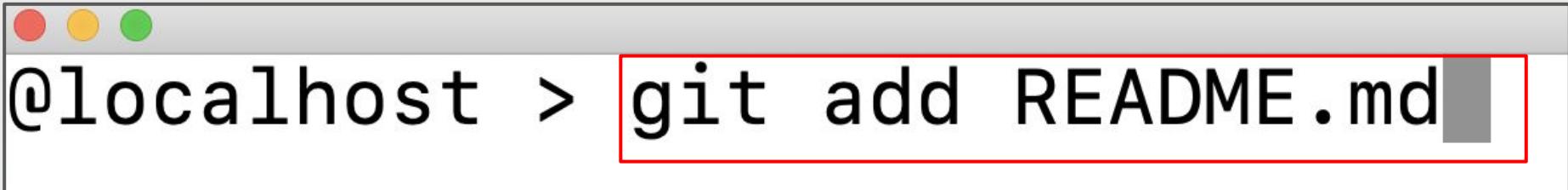
    modified:   README.md

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

The output tells us that we have modified the README.md file. At the bottom, it tells us that we have not added any changes to commit. Let's do that now!

# Let's make a change!

6. Type the command below. This command gets your changes ready to be committed.



A screenshot of a terminal window on a Mac OS X system, indicated by the red, yellow, and green window control buttons at the top left. The terminal is displaying a command line with the text '@localhost > git add README.md' in black font. A red rectangular box highlights the entire command line, specifically the part 'git add README.md'. The background of the terminal window is white.

# Let's make a change!

7. If you enter `git status` again, you will see that now the changes you made to `README.md` are ready to be committed!

```
@localhost > git status  
On branch update-readme  
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)
```

```
    modified: README.md
```

```
@localhost > █
```

# Let's make a change!

- When you commit changes, you will include a commit message. This message should be descriptive of your changes. It's also a standard practice for your first commit to be "first commit."

```
@localhost > git commit -m "first commit"
[update-readme eda5e57] first commit
 1 file changed, 5 insertions(+), 1 deletion(-)
@localhost >
```

# Let's make a change!

- Now for a cool GitHub trick. We want to push the changes that we have committed up to GitHub. Type `git push`.

Git will tell you the correct command! Copy and paste the command to add your local branch to GitHub and commit your changes.

```
@localhost > git push
fatal: The current branch update-readme has no upstream branch.
To push the current branch and set the remote as upstream, use

git push --set-upstream origin update-readme
```

```
@localhost > █
```

# Let's make a change!

10. You'll need to enter your GitHub username and password.

**Note:** When typing your password, you won't see anything.

```
@localhost > git push --set-upstream origin update-readme
Username for 'https://github.com': mlh-localhost
Password for 'https://mlh-localhost@github.com': 
```

# Let's make a change!

11. You will see this output. Now you can create a pull request in the browser like you did before by going to the URL provided.

```
Writing objects: 100% (3/3), 351 bytes | 351.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'update-readme' on GitHub by visiting:
remote:     https://github.com/mlh-localhost/hello-world/pull/new/update-readme
remote:
To https://github.com/mlh-localhost/hello-world.git
 * [new branch]      update-readme -> update-readme
Branch 'update-readme' set up to track remote branch 'update-readme' from 'origin'
.
@localhost > █
```

# Let's make a change!

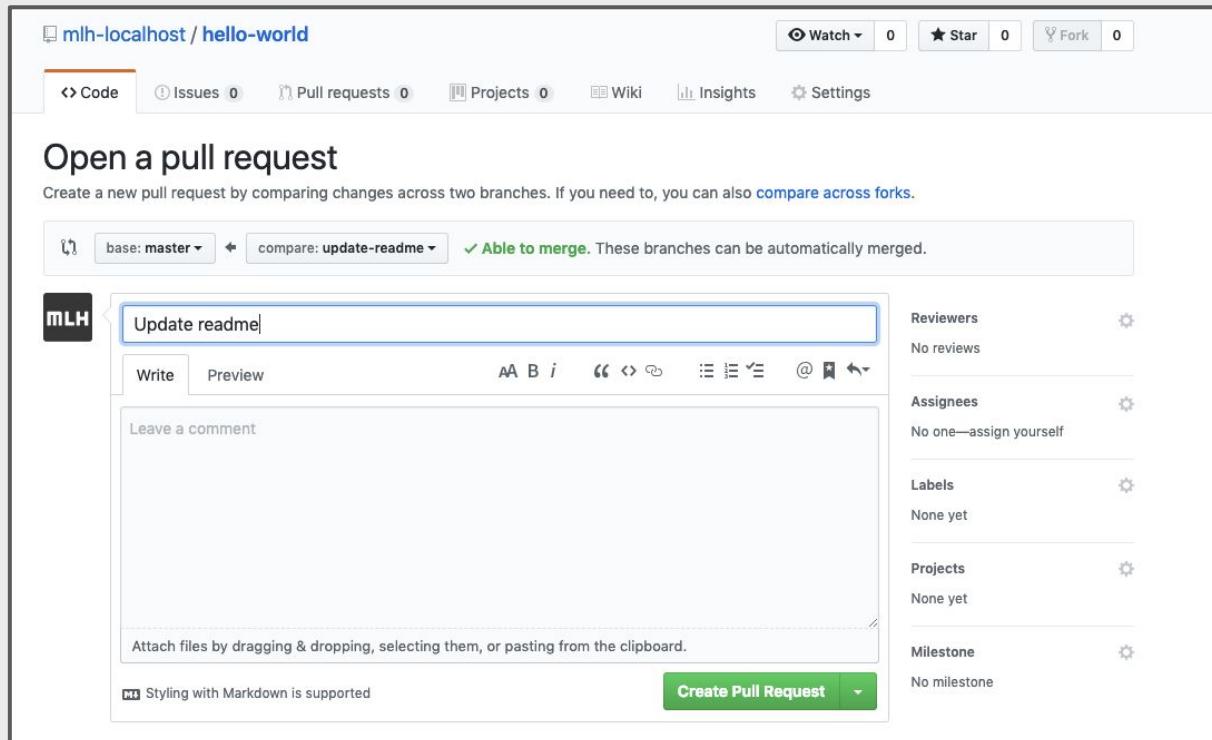
## 12. Select **Compare & pull request**.

This screenshot shows a GitHub repository page for 'mlh-localhost / hello-world'. The page includes standard navigation like Watch (0), Star (0), and Fork (0). Below the navigation, there are tabs for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. A message states 'This repository contains my first GitHub project!' with an 'Edit' button. Below this, there are summary statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. A section for recently pushed branches lists 'update-readme' (less than a minute ago) with a 'Compare & pull request' button. At the bottom, there are buttons for Branch: master, New pull request, Create new file, Upload files, Find file, and Clone or download. A commit history table shows a single commit from 'mlh-localhost' titled 'Initial commit' in 'README.md' at 4 minutes ago.

File	Commit Message	Time Ago
README.md	Initial commit	4 minutes ago

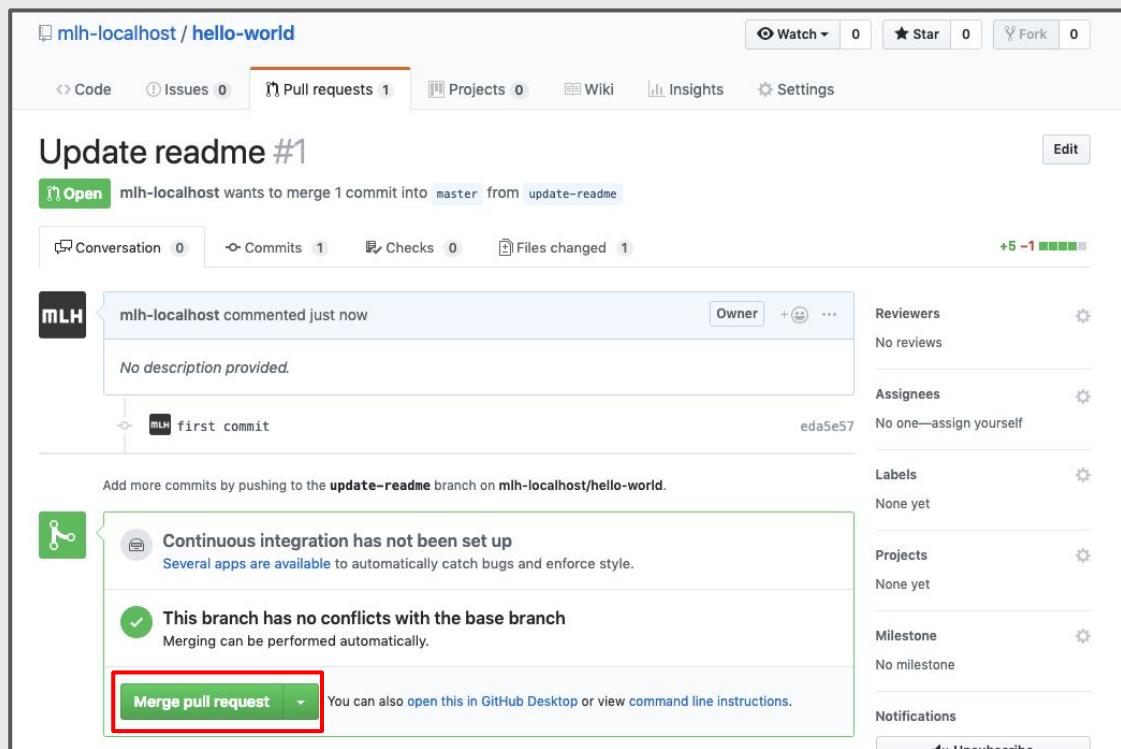
# Let's make a change!

13. Name your pull request appropriately, then select **Create Pull Request**.



# Let's make a change!

- Finally, because you own the repo, you'll be able to merge your own pull request.



# Let's make a change!

15. Now you want to make sure that your local master branch matches the master branch on GitHub. Return to your terminal and type the command below.



```
localhost ~ mlh-hackathon-flask-starter/hello-world -bash 89x16
@localhost > git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

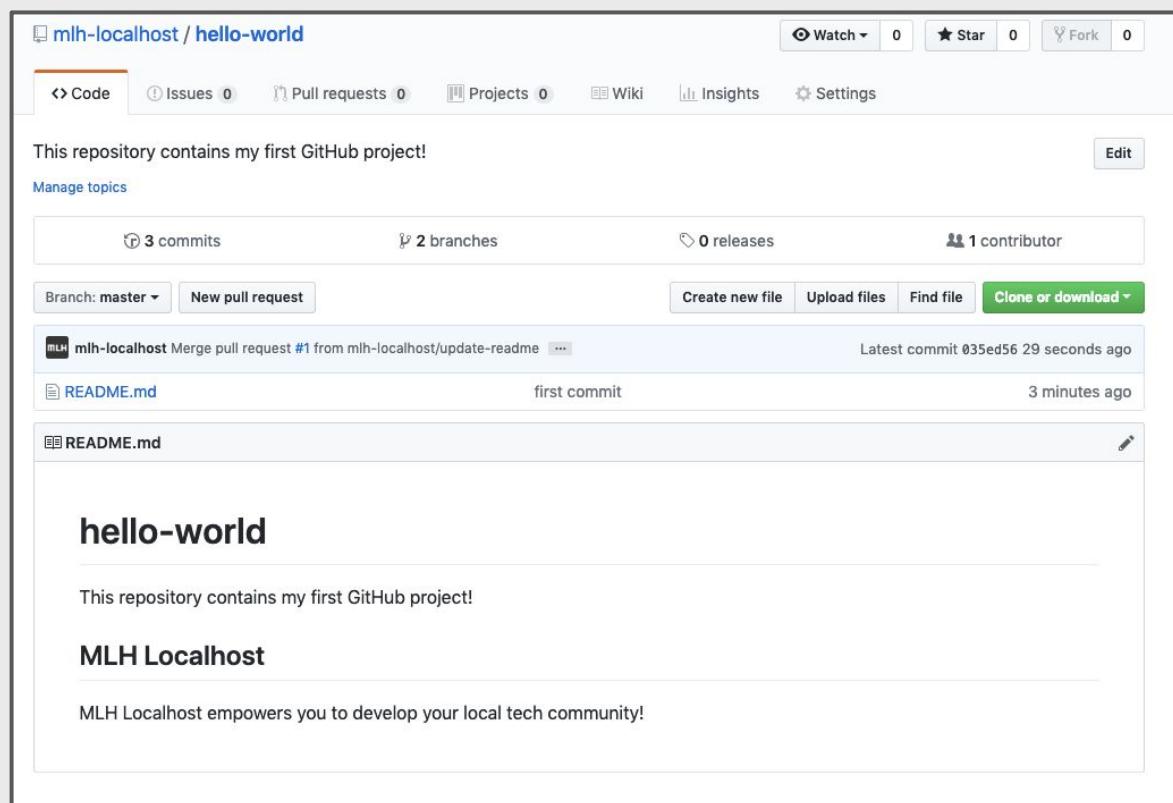
# Let's make a change!

- Finally, pull the changes on master down to your local repo with the command below.

```
@localhost > git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From https://github.com/mlh-localhost/hello-world
  220aa1d..035ed56  master      -> origin/master
Updating 220aa1d..035ed56
Fast-forward
 README.md | 6 +++++-
 1 file changed, 5 insertions(+), 1 deletion(-)
@localhost >
```

# Let's make a change!

Now you should be able to see your changes in the browser!



# Table of Contents

1. Introduction to Git and GitHub
2. GitHub Collaboration WorkFlow
3. GitHub and the Command Line
4. Review & Quiz
5. Next Steps



# GitHub

*Let's recap quickly...*

- 1 Git is a version control system allowing multiple people to collaborate
- 2 GitHub adds features to Git and provides a web interface
- 3 Command-line Git lets you propose changes without the web interface

# Table of Contents

1. Introduction to Git and GitHub
2. GitHub Collaboration WorkFlow
3. GitHub and the Command Line
4. Review & Quiz
5. Next Steps

# Keep Learning: Practice Problems for Later

## Extra Practice Problem 1:

Configure Git with credentials so you don't have to enter your password when pushing code.

## Extra Practice Problem 2:

Use `git stash` to move changes between branches.

# Learning shouldn't stop when the workshop ends...

**Check your email for access to:**



- These workshop slides
- Practice problems to keep learning
- Deeper dives into key topics
- Instructions to join the community
- More opportunities from MLH!



# How to Collaborate with GitHub

---

Workshop

MLH localhost

GitHub