

A photograph of three people 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 text on it. They appear to be in a workshop setting.

How to Collaborate with GitHub

Workshop

MLH localhost

GitHub



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/>

1

*Using your Web Browser,
Open this URL:*

<http://mlhlocal.host/lhd-resources>

2

Click on the workshop you're attending, and find:

- Setup Instructions
- The Code Samples
- A demo project
- A Workshop FAQ
- These Workshop Slides
- More Learning Resources

What will you **learn today?**

1

The basics of Git and GitHub

2

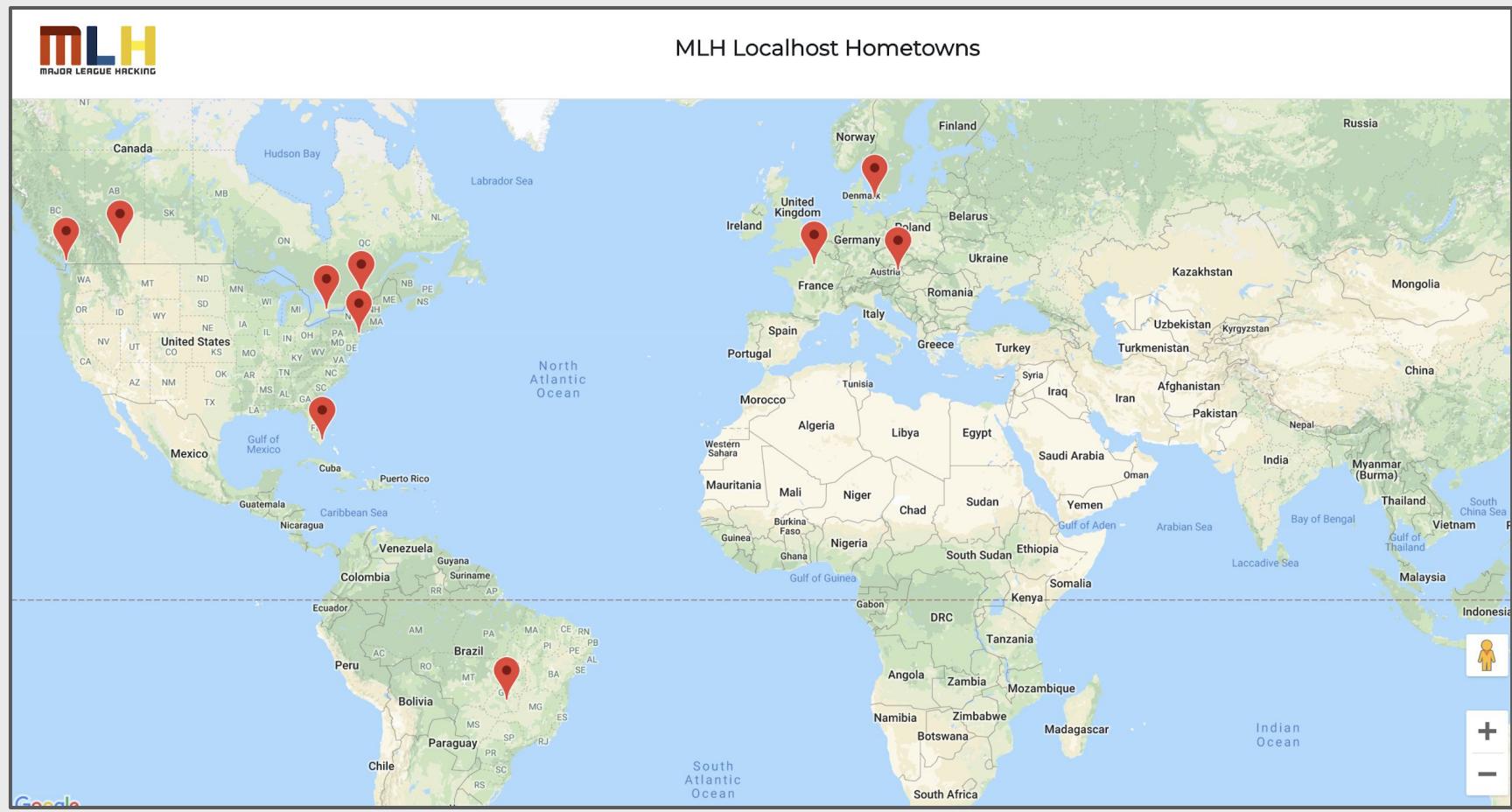
The GitHub Workflow

3

Optional: command-line Git usage

What You'll Be Building

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



How Will You Build It?

1. Make an account on GitHub.
2. Fork the repo that contains the source code for the website you just looked at.
3. Add your hometown and make a pull request.
4. Comment on someone else's pull request.
5. The workshop organizer will approve and merge pull requests.
6. Refresh the main website to see people being added!



**Now that you understand how it works,
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

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

How?

Version control!

Git vs. GitHub

What's the difference?

Git

- Git is a 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.

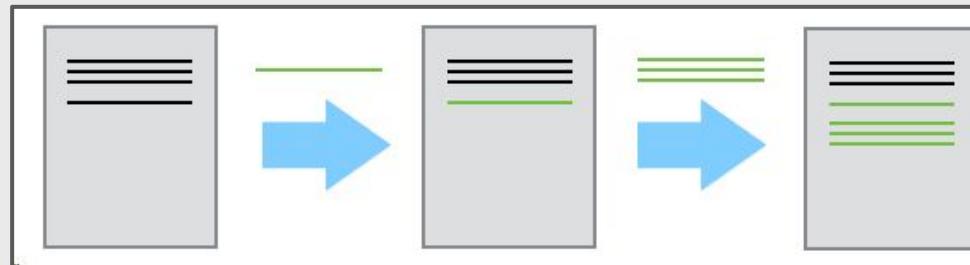
GitHub

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



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 labeled "Learn to ship software like a pro." and a link to "Student Developer Pack".

The screenshot shows the GitHub homepage. 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 for "Username" is visible at the bottom.

Explore GitHub

The GitHub platform provides numerous features for collaboration.

A screenshot of the GitHub dashboard. At the top, there is a navigation bar with a search bar, a clock icon, and tabs for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The 'Pull requests' tab is highlighted with a red box. On the left, there is a sidebar titled 'Repositories' with a message saying 'Your most active repositories will appear here.' and links to 'Create a repository' or 'explore repositories'. The main area features a large card with the heading 'Learn Git and GitHub without any code!' and a subtext: 'Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.' It has two buttons: a green 'Read the guide' button and a white 'Start a project' button. To the right, there is a 'Discover repositories' section with a welcome message: 'Welcome to the new dashboard. Get closer to the stuff you care about most.' Below it, there are two repository cards: 'ARMmbed/mbed-os-5-docs' (Python, 50 stars) and 'apache/beam' (Java, 2.6k stars).

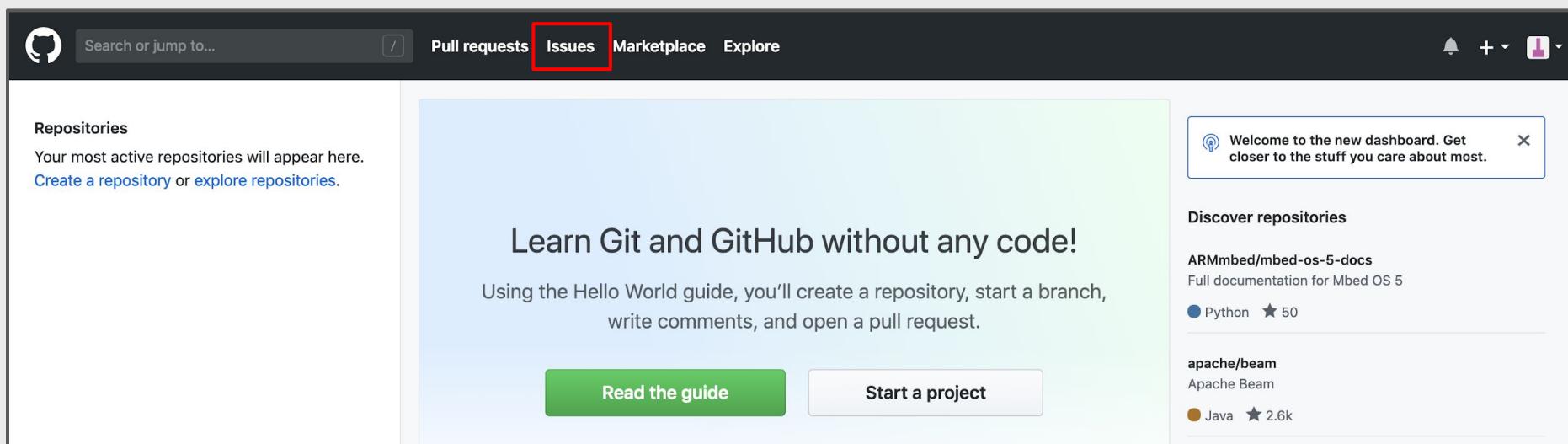
Next to the search bar, you can select Pull requests. This will show you your own pull requests against other people's repos.

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 dashboard. At the top, there is a navigation bar with links for "Pull requests", "Issues" (which is highlighted with a red box), "Marketplace", and "Explore". On the left, there is a sidebar titled "Repositories" with a message saying "Your most active repositories will appear here." and links to "Create a repository" or "explore repositories". The main area features a large callout box with the text "Learn Git and GitHub without any code!" and "Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request." Below this are two buttons: a green "Read the guide" button and a white "Start a project" button. To the right, there is a "Discover repositories" section with cards for "ARMmbed/mbed-os-5-docs" (Python, 50 stars), "apache/beam" (Java, 2.6k stars), and a welcome message: "Welcome to the new dashboard. Get closer to the stuff you care about most." There are also notification and profile icons at the top right.

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 dashboard. At the top, there is a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. On the far right of the dashboard header, there is a red-bordered box highlighting the notification bell icon, the '+' icon for creating a new project, and the user's profile picture. The main content area features a large central box with the heading "Learn Git and GitHub without any code!" and a subtext explaining how to use the Hello World guide to create a repository, start a branch, write comments, and open a pull request. Below this are two buttons: a green "Read the guide" button and a white "Start a project" button. To the left, there is a sidebar titled "Repositories" with a message about active repositories and links to "Create a repository" or "explore repositories". On the right, there is a "Discover repositories" section listing "ARMmbed/mbed-os-5-docs" (Python, 50 stars), "apache/beam" (Java, 2.6k stars), and other repository cards.

- 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 for the map!

The screenshot shows a GitHub repository page for 'mlh-localhost-github'. The repository has 2 commits, 1 branch, 0 releases, 1 contributor, and an MIT license. The latest commit was made 4 minutes ago by JamieMLH, updating the README. The repository contains files like src, static, views, LICENSE, README.md, index.js, locations.txt, and package.json. A section titled 'MLH Localhost Hometown Map' describes the purpose of the repository. A large blue box highlights the URL 'http://mlhlocal.host/github-code'.

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

This repo contains the source code for the MLH Localhost workshop, How to Collaborate on Code Projects with GitHub. Participants learn best practice GitHub workflows, add their hometown to

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.

This screenshot shows a GitHub repository page for 'MLH / mlh-localhost-github'. The 'Code' tab is highlighted with a red box. The page displays basic repository statistics: 2 commits, 1 branch, 0 releases, 1 contributor, and an MIT license. A list of files shows recent commits from JamieMLH, all made 4 minutes ago. Below the file list is a section titled 'MLH Localhost Hometown Map' with a descriptive paragraph about the repository's purpose.

MLH / mlh-localhost-github

Watch 4 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

This repo contains the source code for the MLH Localhost workshop, How to Collaborate on Code Projects with GitHub. Edit

Manage topics

2 commits 1 branch 0 releases 1 contributor MIT

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

File	Commit Message	Time
src	Initial Commit	8 minutes ago
static	Initial Commit	8 minutes ago
views	Initial Commit	8 minutes ago
LICENSE	Initial Commit	8 minutes ago
README.md	update readme	4 minutes ago
index.js	Initial Commit	8 minutes ago
locations.txt	Initial Commit	8 minutes ago
package.json	Initial Commit	8 minutes ago

Latest commit d8409a8 4 minutes ago

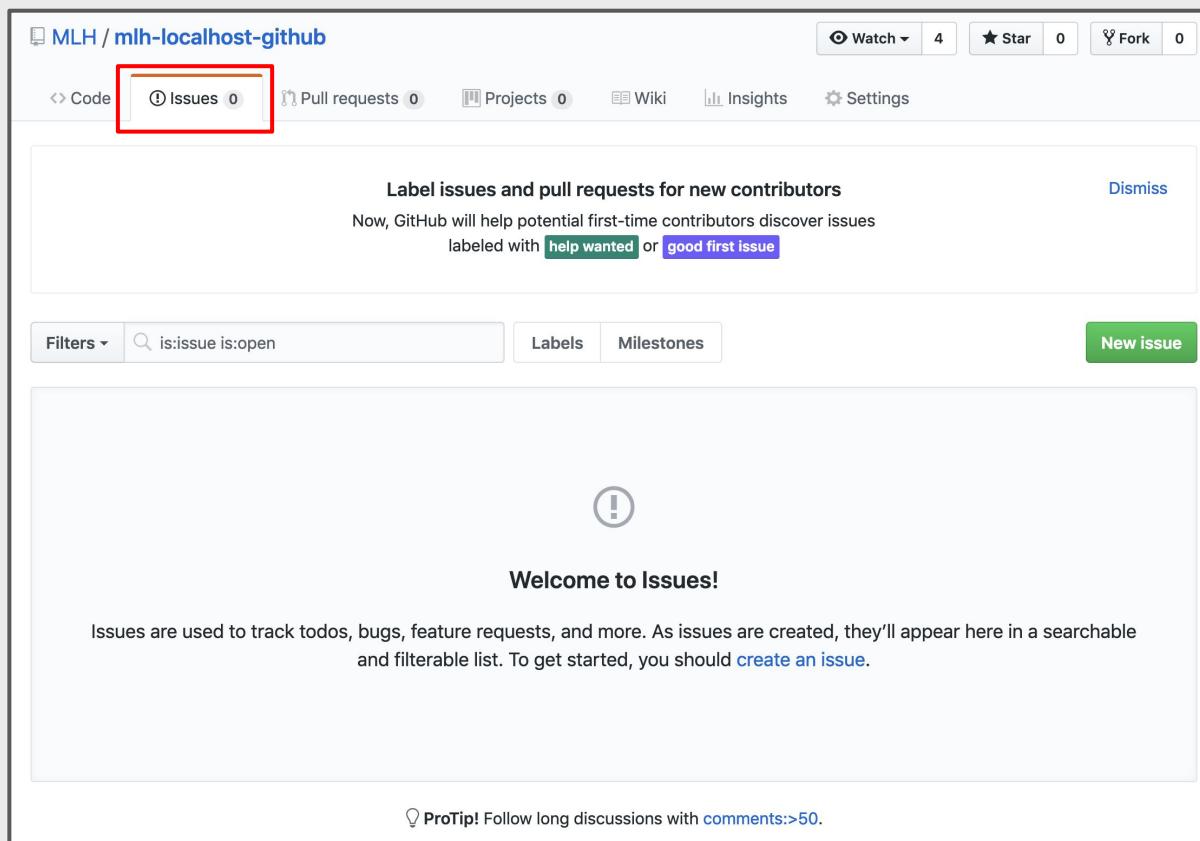
src static views LICENSE README.md index.js locations.txt package.json

MLH Localhost Hometown Map

This repository contains the source code for a map of the hometowns of anyone who attends the MLH Localhost workshop, How to Collaborate on Code Projects with GitHub. Participants learn best practice GitHub workflows, add their hometown to

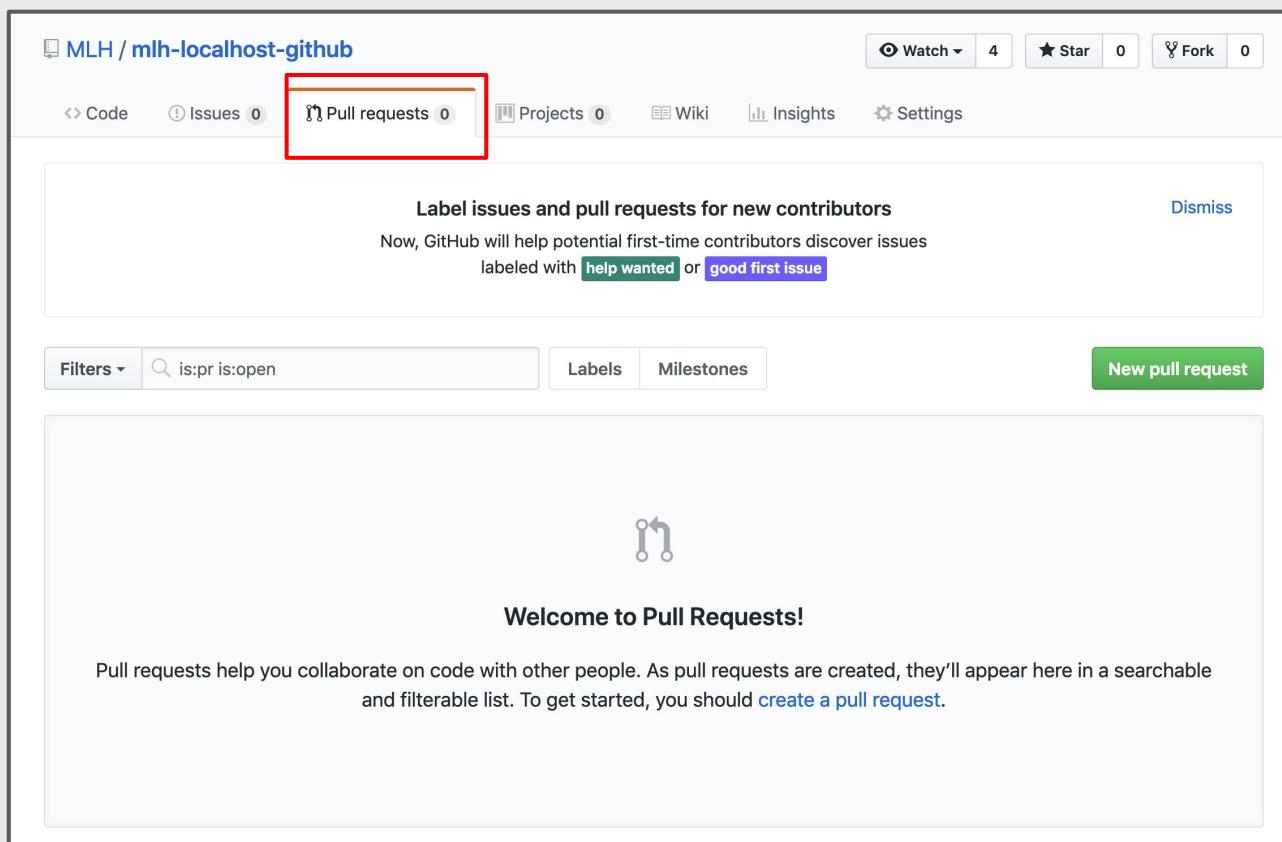
Explore a Repository

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.



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!



**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

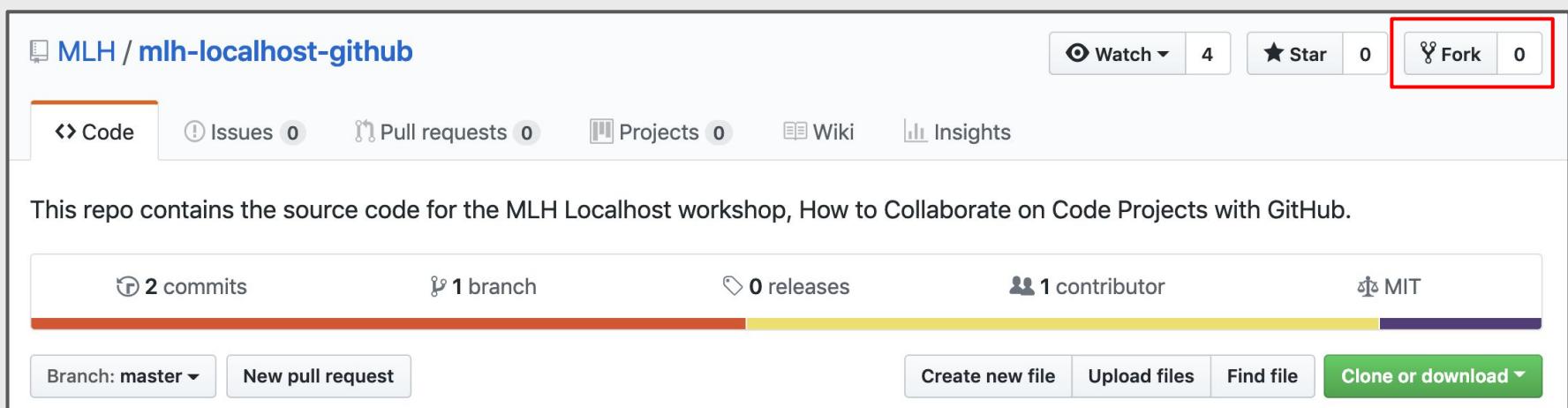
merge: to officially add the changes from your branch into the master branch (or another branch)

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

Network 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.

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



The screenshot shows a GitHub repository page for 'MLH / mlh-localhost-github'. The top navigation bar includes 'Watch' (4), 'Star' (0), and a 'Fork' button (0), which is highlighted with a red box. Below the header, there are links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', and 'Insights'. A descriptive text block states: 'This repo contains the source code for the MLH Localhost workshop, How to Collaborate on Code Projects with GitHub.' Key statistics are displayed: '2 commits', '1 branch', '0 releases', '1 contributor', and 'MIT'. At the bottom, there are buttons for 'Branch: master ▾', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a large green 'Clone or download ▾' button.

Note: This is only necessary if you are not a Collaborator on the project.

Network Activity Workflow

2. 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.

The image shows two side-by-side screenshots of a GitHub interface. On the left, a repository named 'mlh-localhost / mlh-localhost-github' is displayed. The 'Code' tab is selected. At the bottom of the page, there is a red box around the 'Branch: master' dropdown menu. On the right, a modal dialog box is open, titled 'Switch branches/tags'. It contains a search input field with 'mlh-add-city' typed into it. Below the input field are tabs for 'Branches' and 'Tags', with 'Branches' being the active tab. A large blue button at the bottom of the dialog box says 'Create branch: mlh-add-city from 'master''.

mlh-localhost / mlh-localhost-github
forked from MLH/mlh-localhost-github

Code Pull requests 0 Projects 0 Wiki Insights

This repo contains the source code for the MLH Localhost workshop, How to

Manage topics

2 commits 1 branch 0 releases

Branch: master ▾ New pull request

Branch: master ▾ New pull request

Switch branches/tags

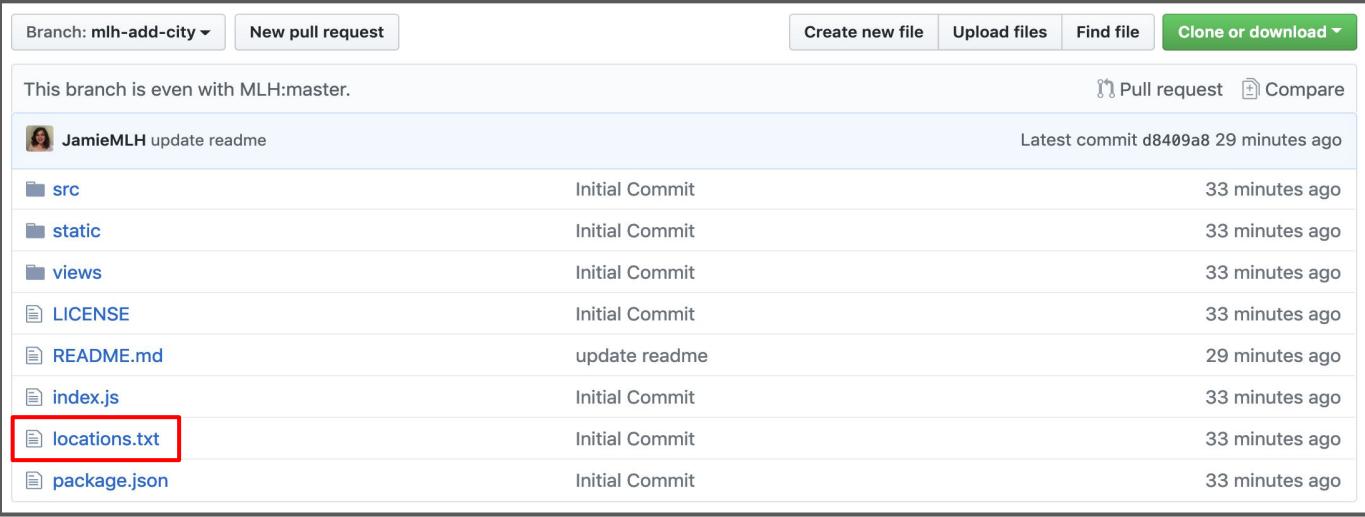
mlh-add-city

Branches Tags

Create branch: mlh-add-city from 'master'

Developer Workflow

3. Open locations.txt.



This branch is even with MLH:master.

Branch: mlh-add-city ▾ New pull request Create new file Upload files Find file Clone or download ▾

Pull request Compare Latest commit d8409a8 29 minutes ago

File	Commit Message	Time
src	Initial Commit	33 minutes ago
static	Initial Commit	33 minutes ago
views	Initial Commit	33 minutes ago
LICENSE	Initial Commit	33 minutes ago
README.md	update readme	29 minutes ago
index.js	Initial Commit	33 minutes ago
locations.txt	Initial Commit	33 minutes ago
package.json	Initial Commit	33 minutes ago

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



Branch: mlh-add-city ▾ mlh-localhost-github / locations.txt Find file Copy path

JamieMLH Initial Commit 0948092 34 minutes ago

1 contributor

17 lines (16 sloc) | 260 Bytes

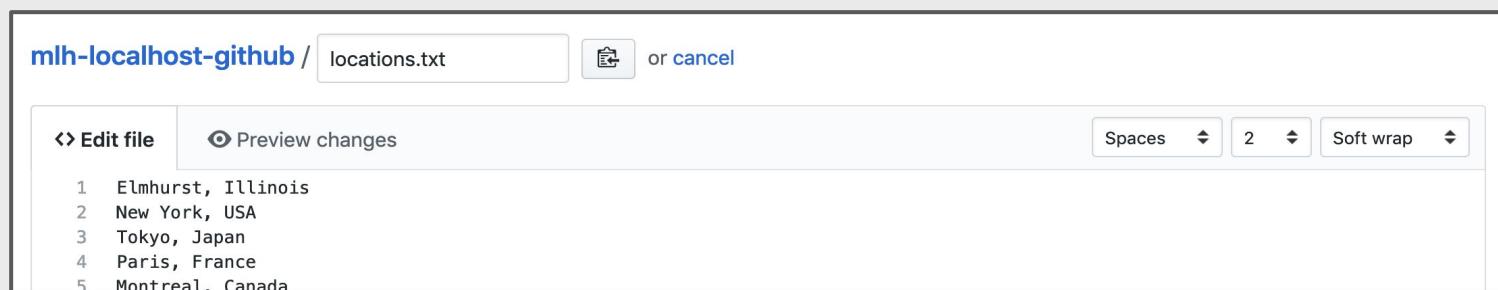
Raw Blame History

1 New York, USA		
-----------------	---	---

2 Tokyo, Japan

Network Activity Workflow

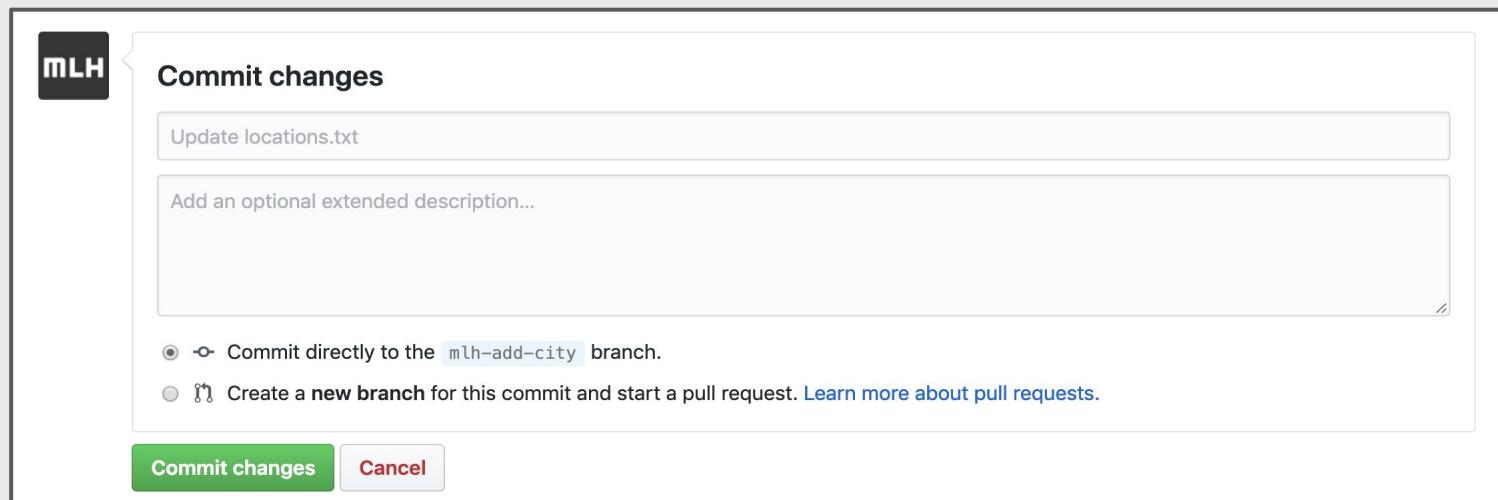
5. Add the name of your hometown. Choose a random blank line!



The screenshot shows a GitHub code editor interface. The repository path is 'mlh-localhost-github / locations.txt'. There are two tabs: 'Edit file' (selected) and 'Preview changes'. On the right, there are buttons for 'Spaces', '2', and 'Soft wrap'. The code editor contains the following text:

```
1 Elmhurst, Illinois
2 New York, USA
3 Tokyo, Japan
4 Paris, France
5 Montreal, Canada
```

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



Network Activity Workflow

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

The screenshot shows a 'Commit changes' dialog box. At the top left is the MLH logo. The main title is 'Commit changes'. Below it is a text input field containing 'Add my hometown to locations.txt'. Underneath is a larger, empty text area labeled 'Add an optional extended description...'. At the bottom, there are two radio button options: one selected ('Commit directly to the mlh-add-city branch.') and one unselected ('Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)'). At the very bottom are two buttons: a green 'Commit changes' button with a red border and a white 'Cancel' button.

Commit changes

Add my hometown to locations.txt

Add an optional extended description...

⚡ Commit directly to the `mlh-add-city` branch.

🏙 Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

Network Activity Workflow

8. Return to the main page of your repo.
9. Select **Pull request**.

This repo contains the source code for the MLH Localhost workshop, How to Collaborate on Code Projects with GitHub. [Edit](#)

[Manage topics](#)

3 commits 2 branches 0 releases 1 contributor MIT

Your recently pushed branches:

mlh-add-city (less than a minute ago)[Compare & pull request](#)

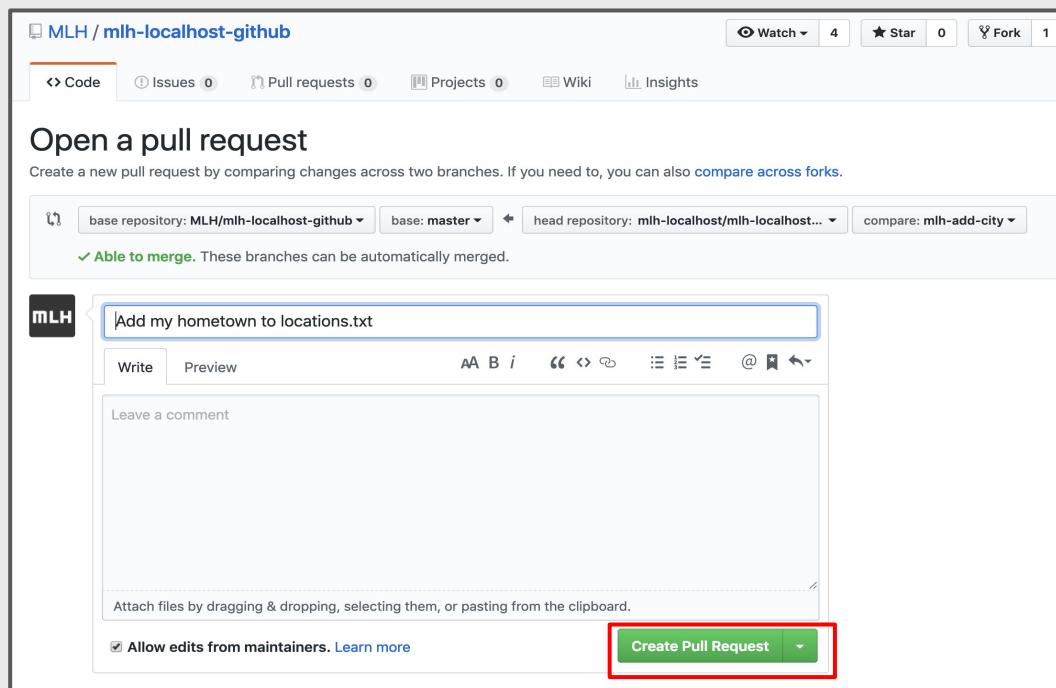
Branch: mlh-add-city ▾New pull requestCreate new fileUpload filesFind fileClone or download ▾

This branch is 1 commit ahead of MLH:master.[Pull request](#)[Compare](#)

mlh-localhost Add my hometown to locations.txtLatest commit 488c579 23 seconds ago

Network Activity Workflow

10. 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.



Network Activity Workflow

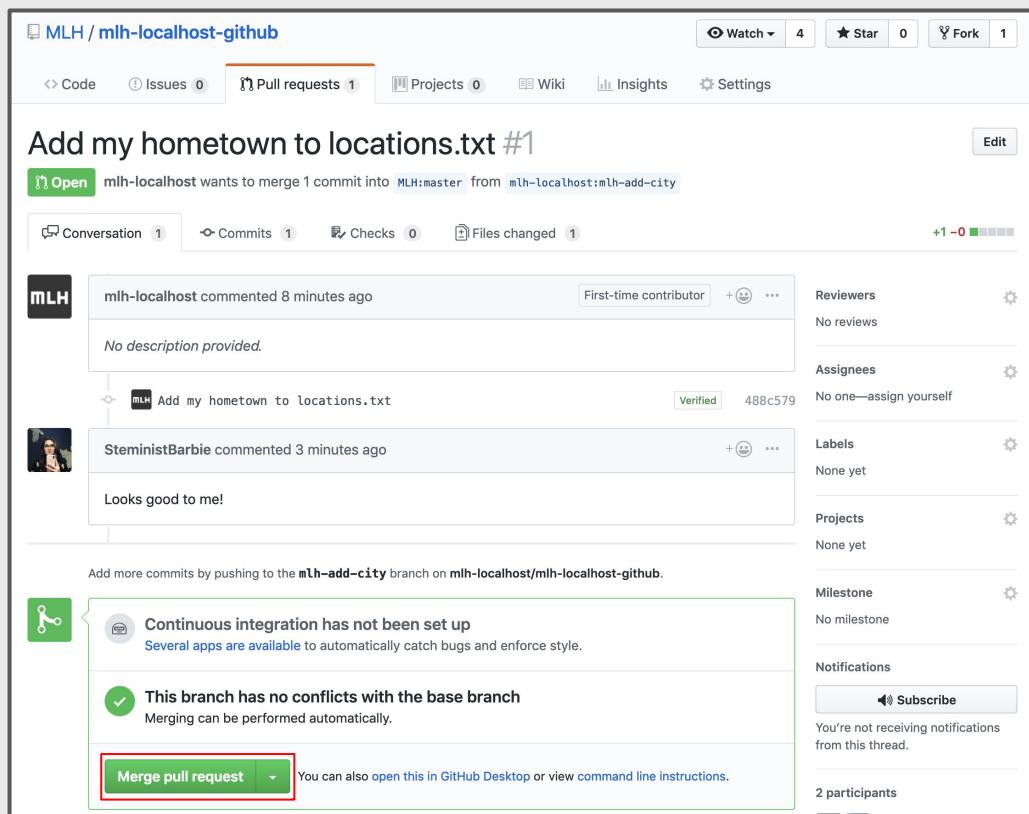
11. Ask someone else to comment on your PR. Comment on someone else's PR, too!

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 interface for a repository named "mlh-localhost". The pull request is titled "Add my hometown to locations.txt #1". It 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 a link to the pull request "mlh Add my hometown to locations.txt". A green callout box highlights a message: "This branch has no conflicts with the base branch. Only those with write access to this repository can merge pull requests." In the bottom right corner of the comment area, there is a text input field containing "Looks good to me!" and a "Comment" button.

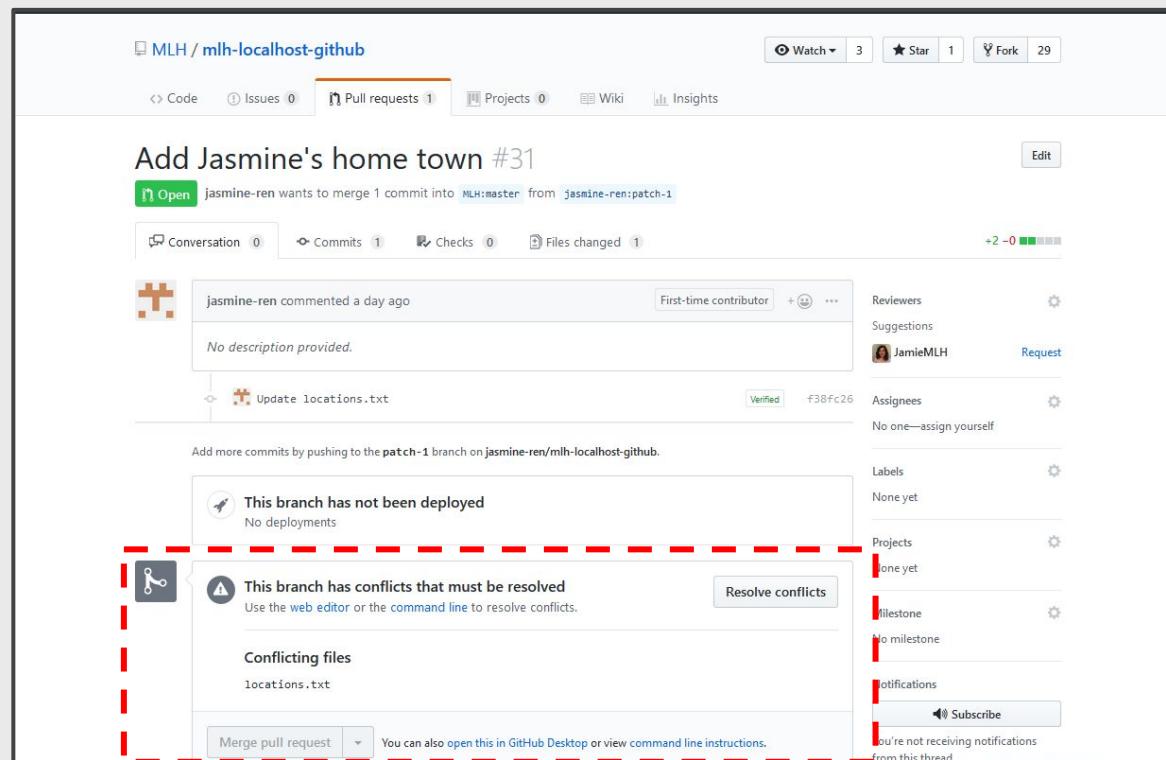
Network Activity Workflow

12. 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 hometown will be added!



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

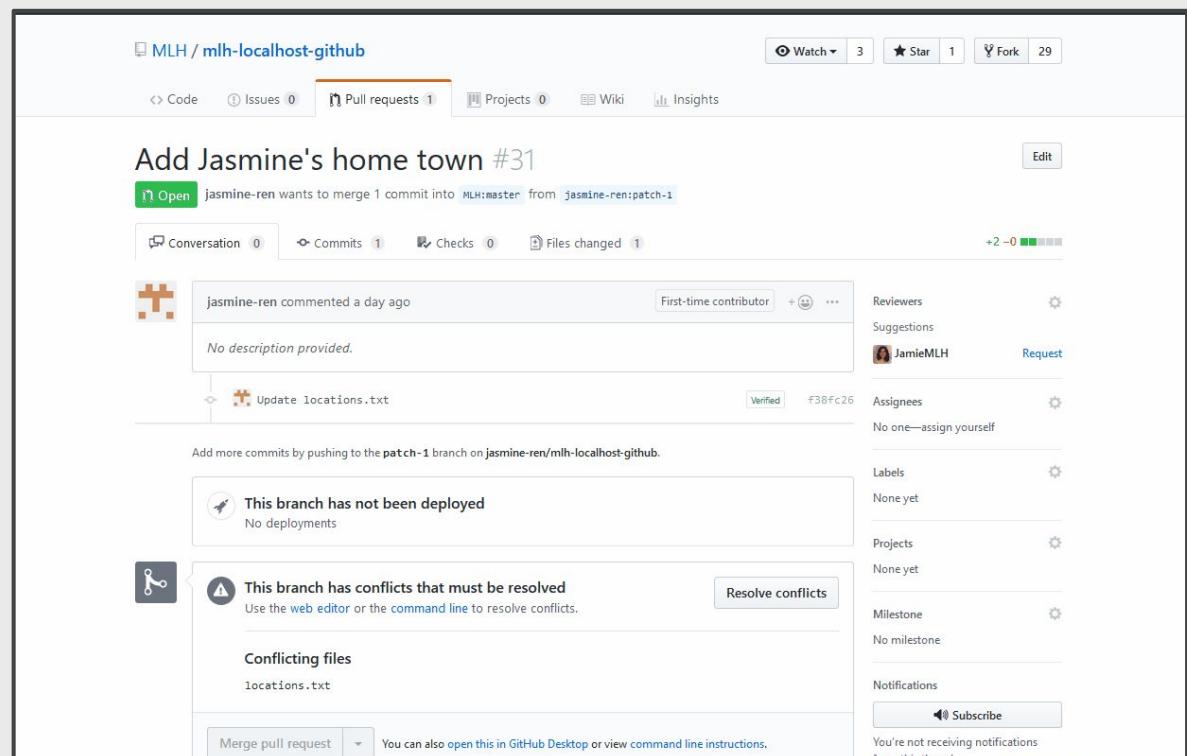
changes

=====

original

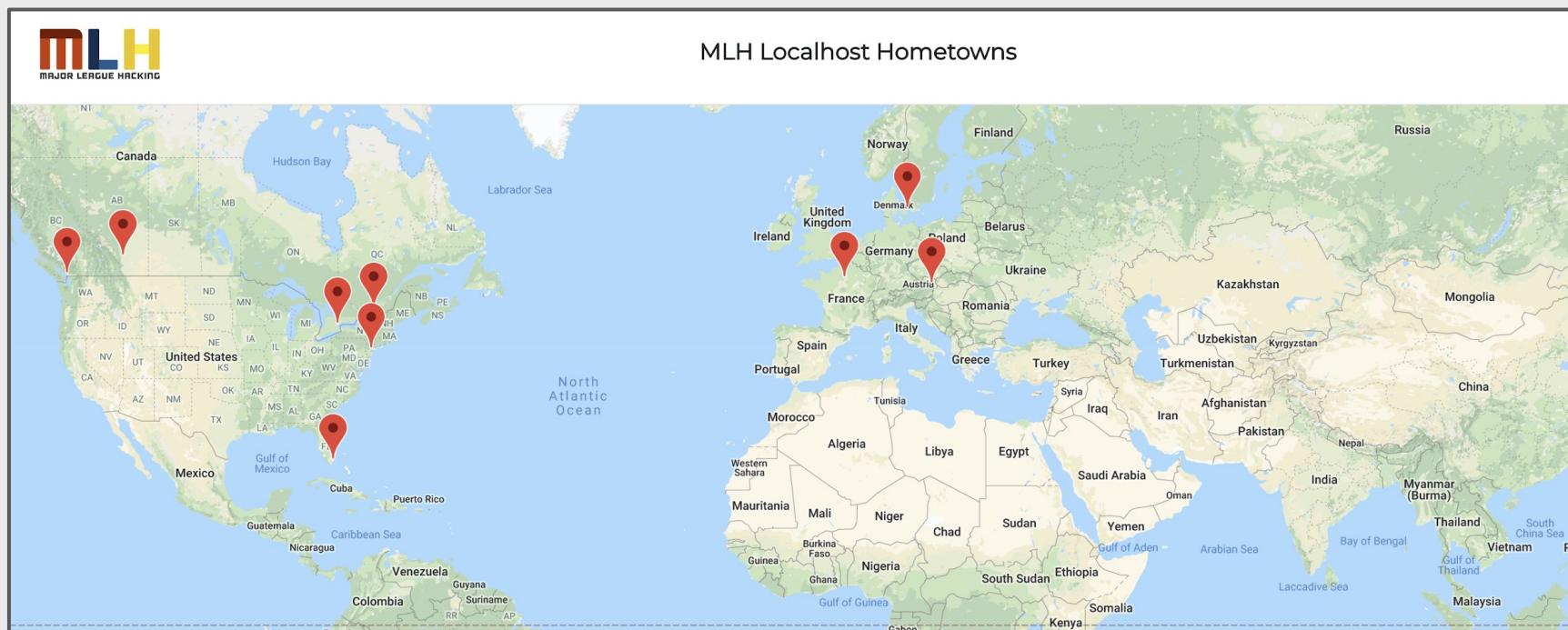
>>>>> original-branch

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



Network 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.



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!

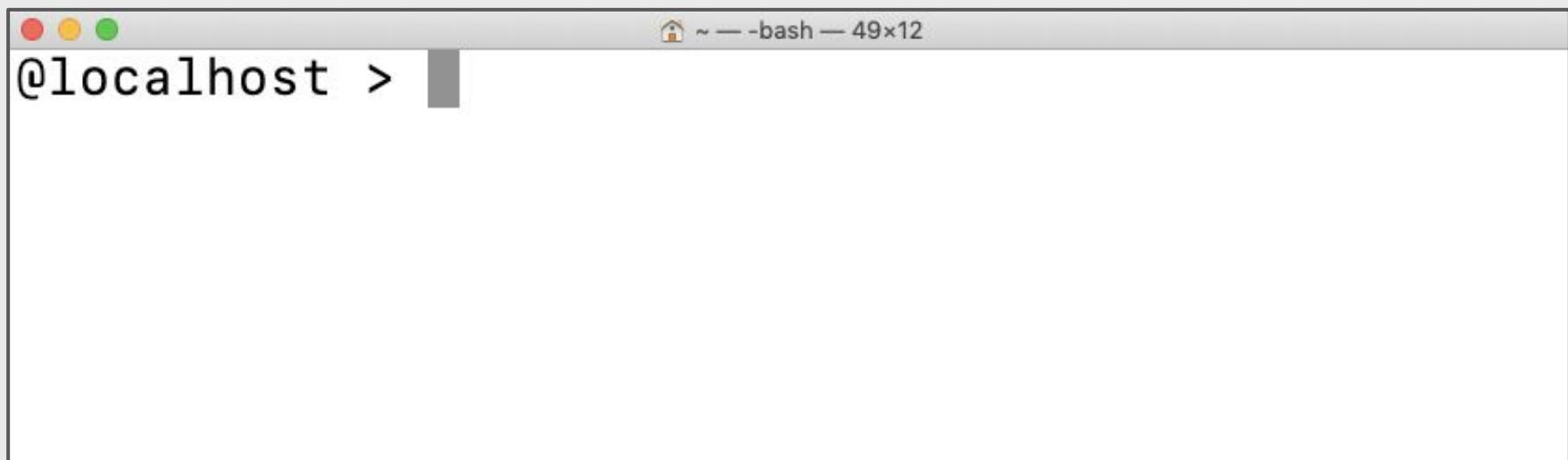
Table of Contents

- 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.



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 the standard red, yellow, and green buttons. The title bar itself says '~ - bash - 49x12'. The main pane contains the command '@localhost > git --version' followed by the output 'git version 2.17.2 (Apple Git-113)'. Below the output is another prompt '@localhost >'. The background of the slide is light gray, and the terminal window has a dark gray border.

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

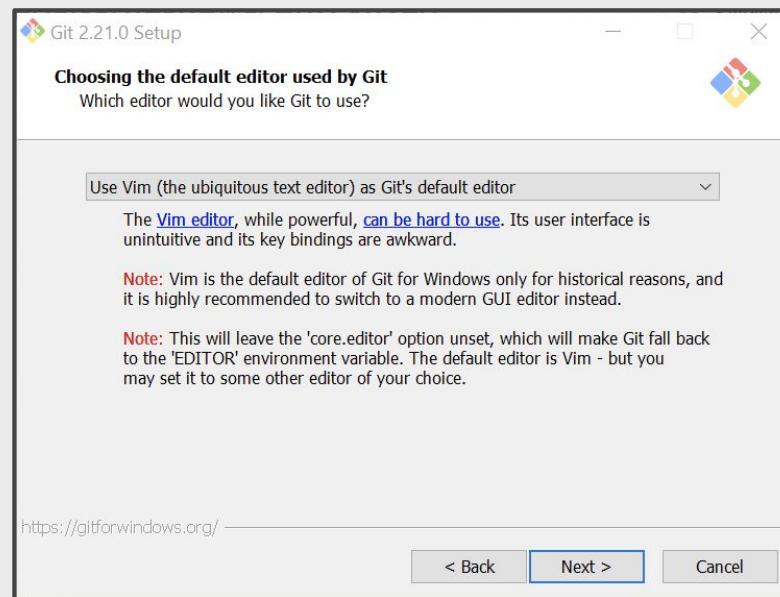
Install Git: MacOS

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 >
```

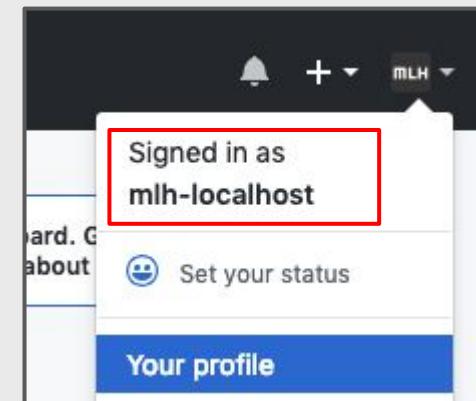
A screenshot of a macOS terminal window. The window title bar shows the standard red, yellow, and green close buttons, followed by the path '~ - bash - 49x12'. The main pane of the terminal contains the command 'git --version' followed by its output 'git version 2.17.2 (Apple Git-113)'. The command 'git --version' is highlighted with a red rectangular box. The terminal 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

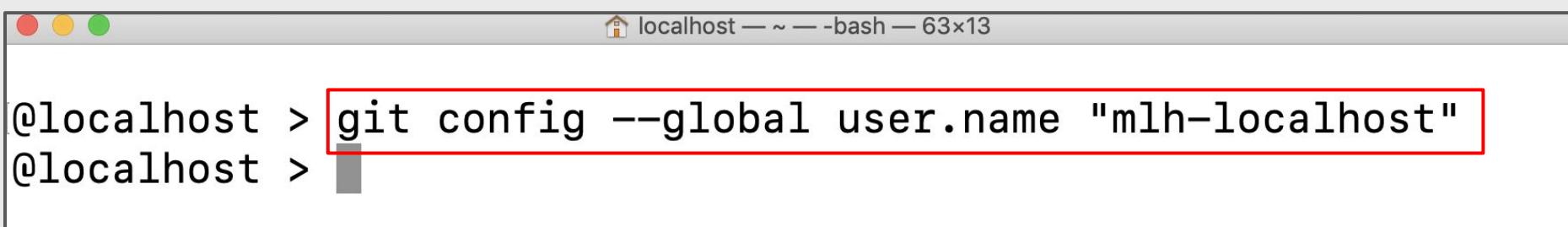
We want to configure our local environments so that the correct GitHub account is associated with our commits.

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"



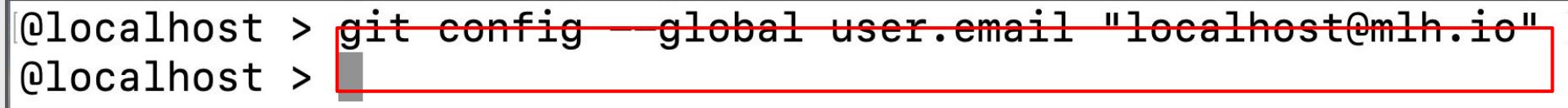
Configure Git

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 >
```

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



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

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

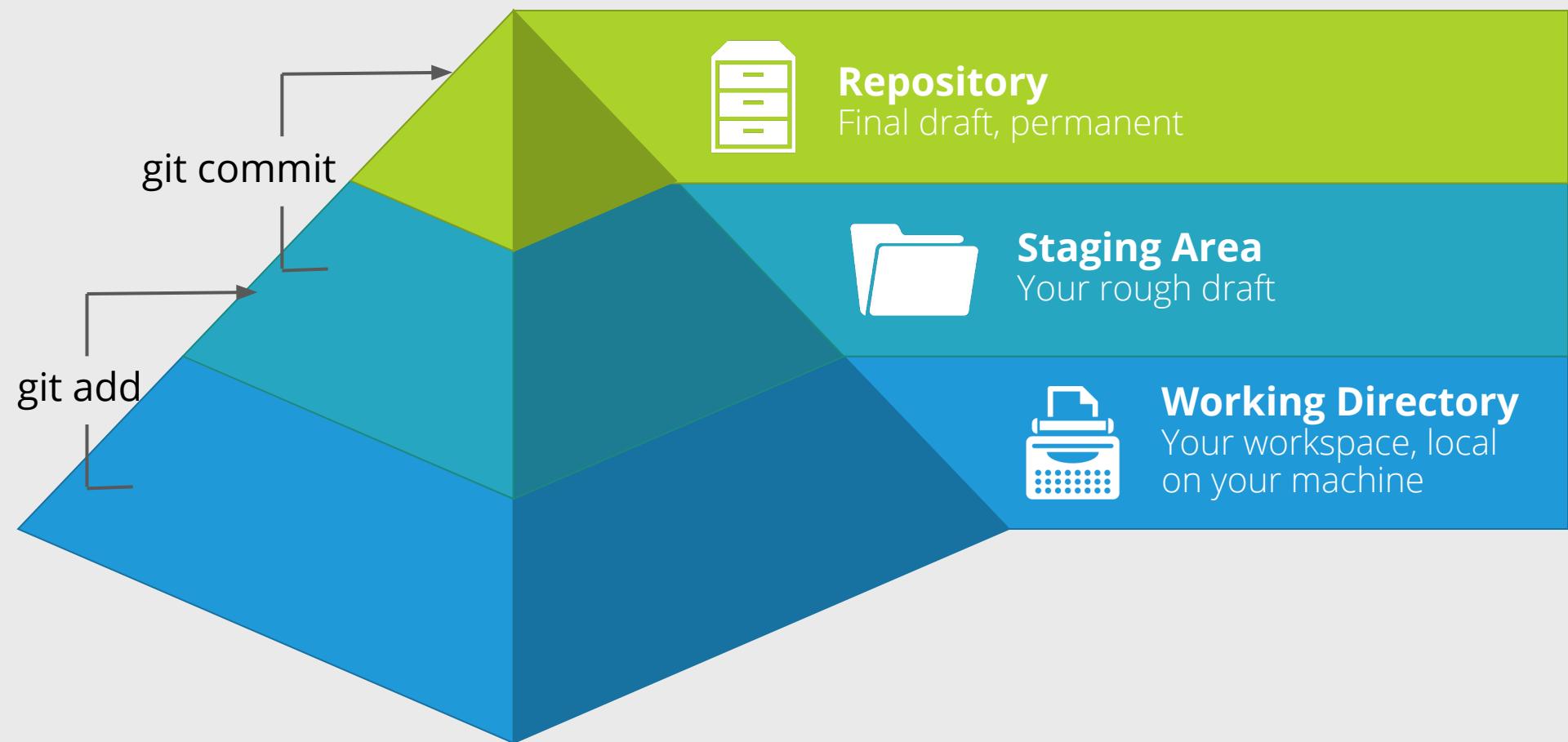
Staging area

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

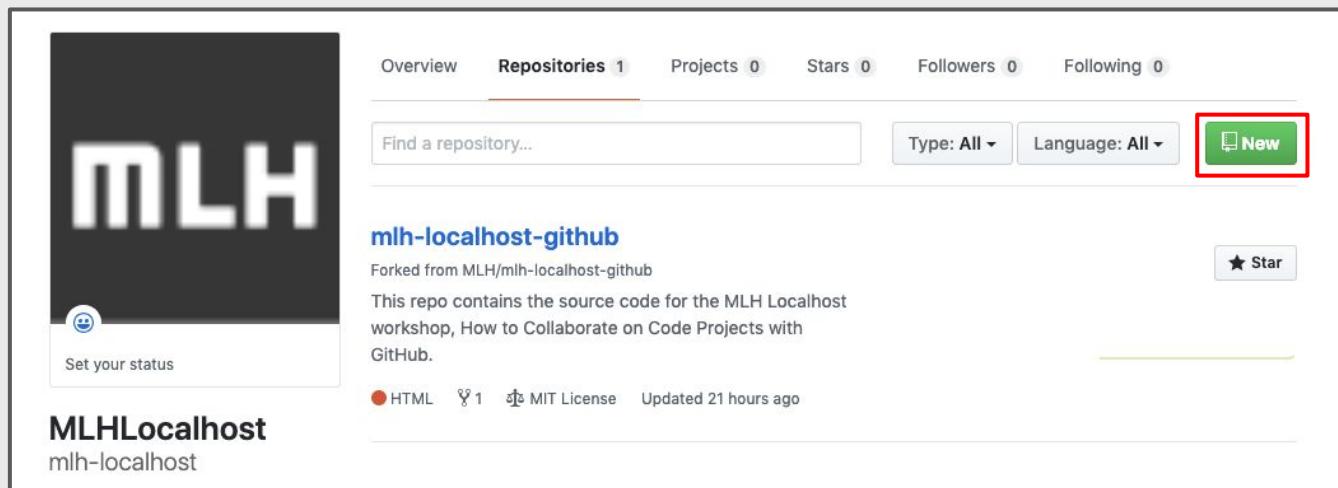


Great! Let's try it.

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.

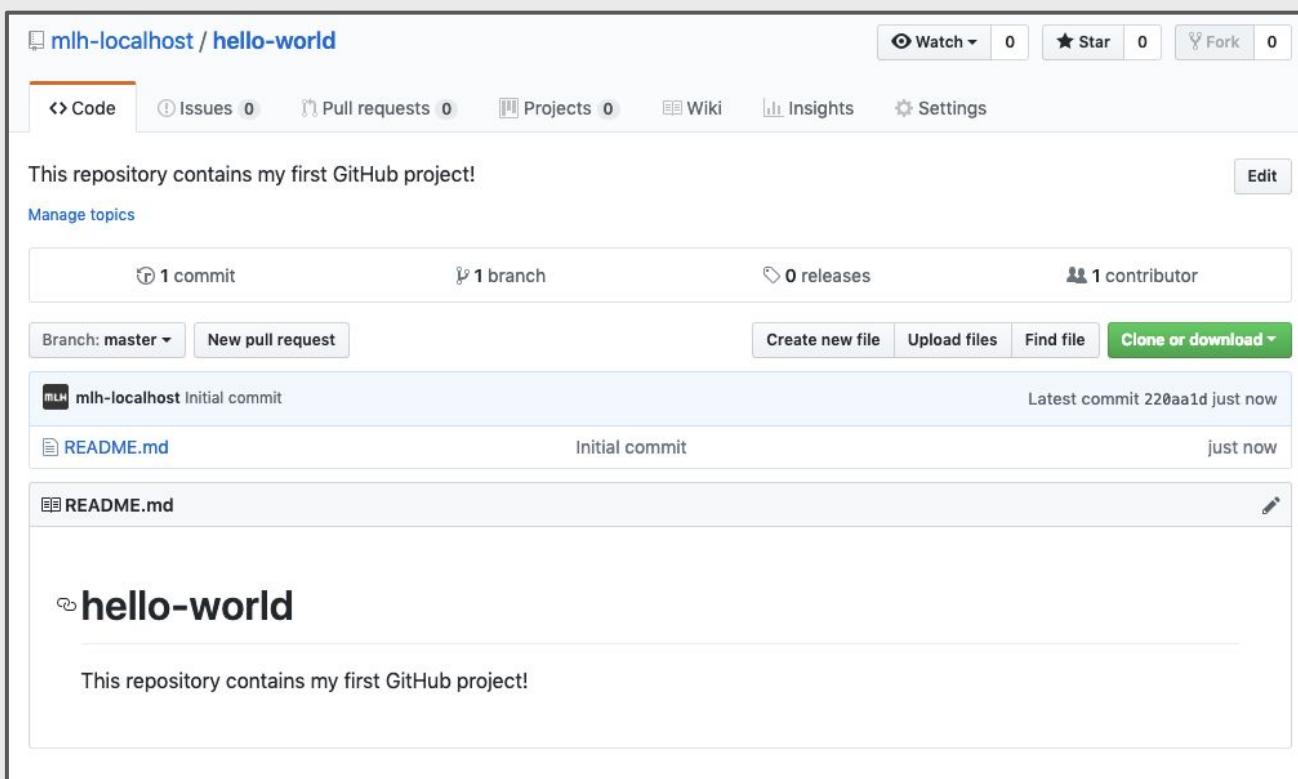
Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Add a license: 

Create repository

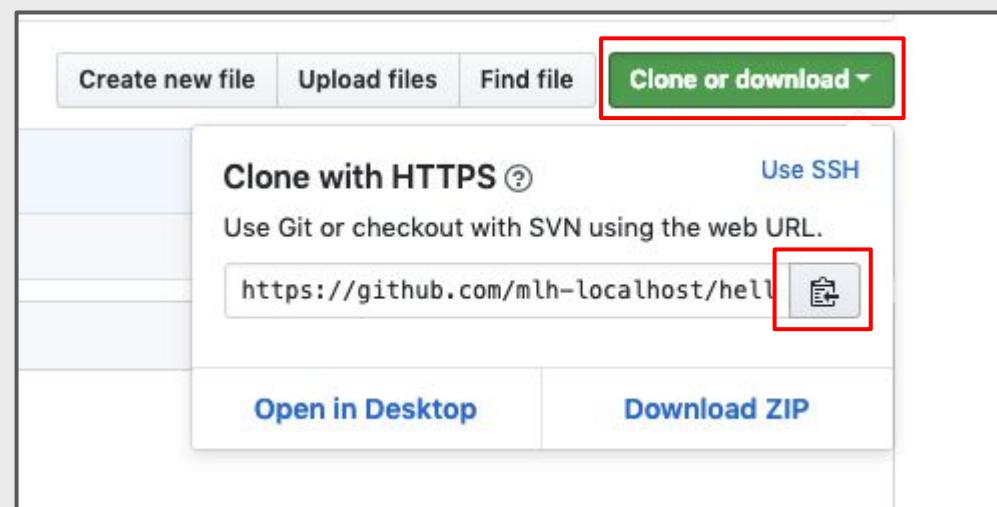
Create a Repo

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.



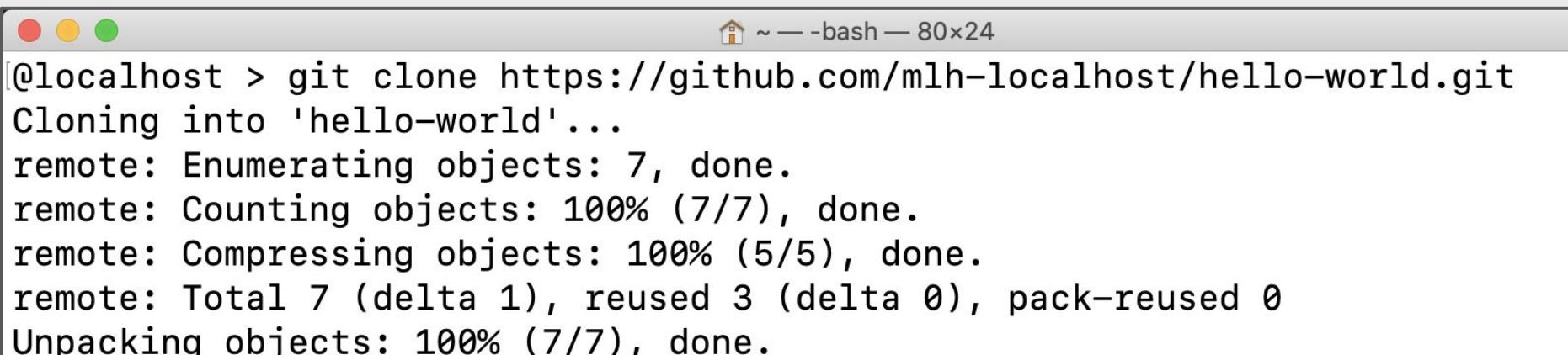
Clone Your Repo

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:

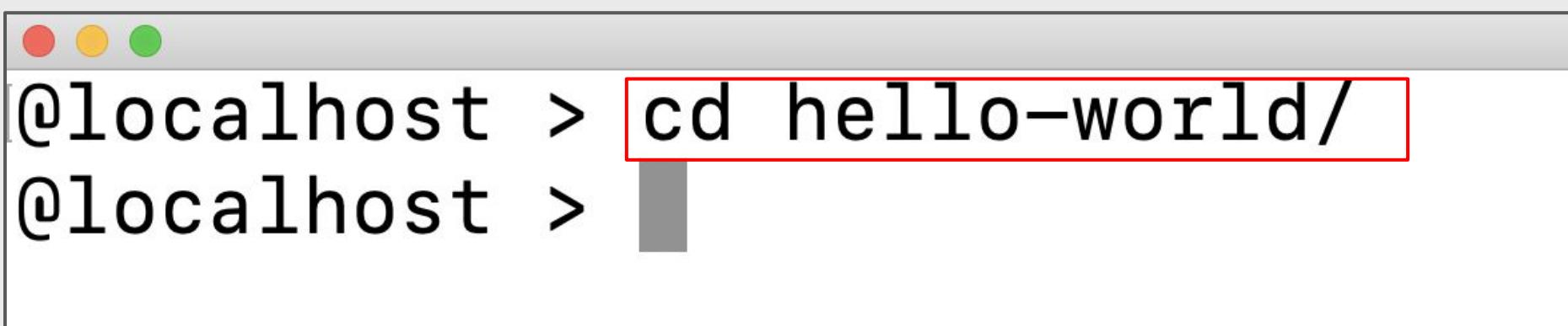
```
[@localhost ~] ~/hello-world — bash — 80x24
[@localhost > cd hello-world/
[@localhost > ls -a
.                         ..
[@localhost > .git          README.md
```

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.



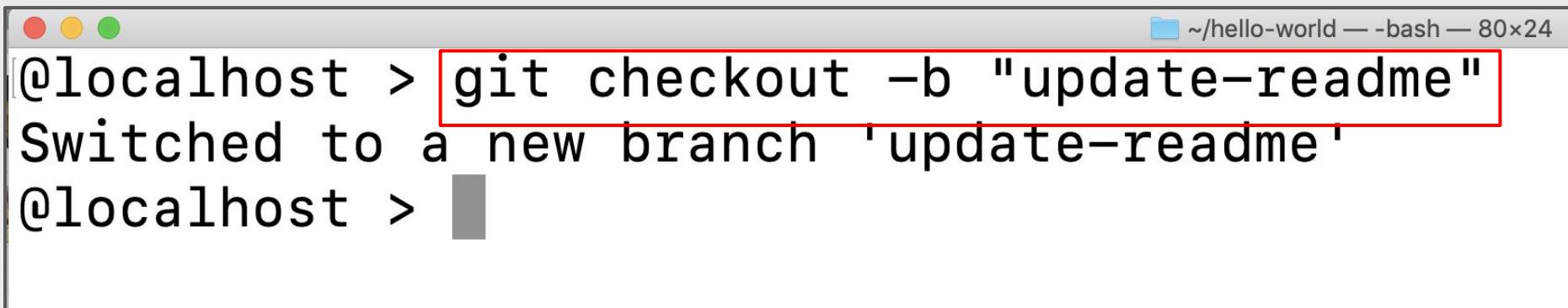
A screenshot of a terminal window on a Mac OS X system, indicated by the red, yellow, and green window control buttons at the top. The terminal text area shows two lines of input:

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

The command `cd hello-world/` is highlighted with a red rectangular box.

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.

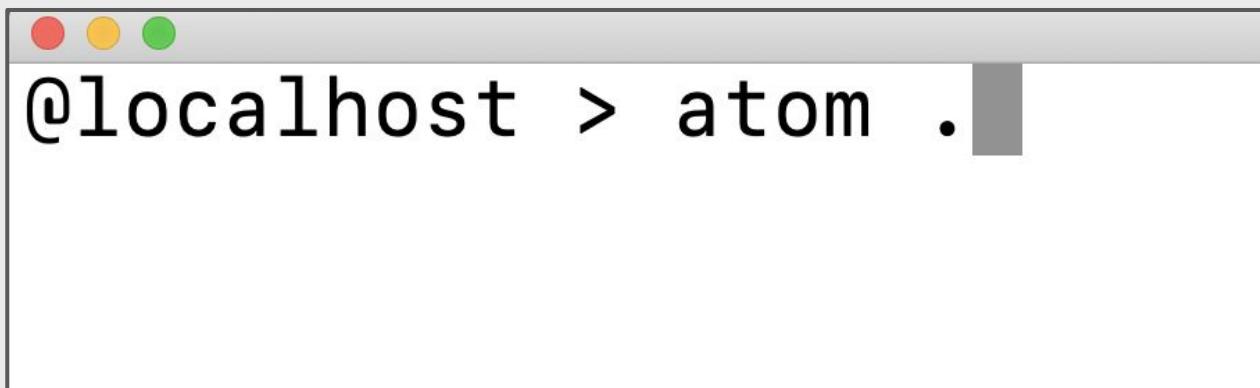


A screenshot of a macOS terminal window titled '~/.hello-world — -bash — 80x24'. The window shows the command 'git checkout -b "update-readme"' highlighted with a red rectangle. The output of the command 'Switched to a new branch 'update-readme'' is also visible. The terminal has its characteristic red,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

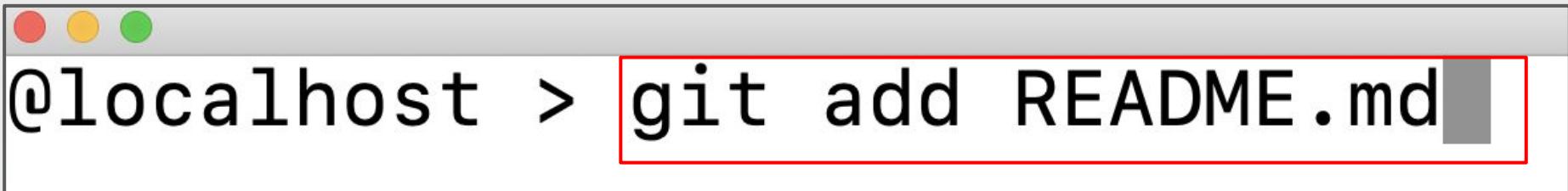
    modified:   README.md

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!

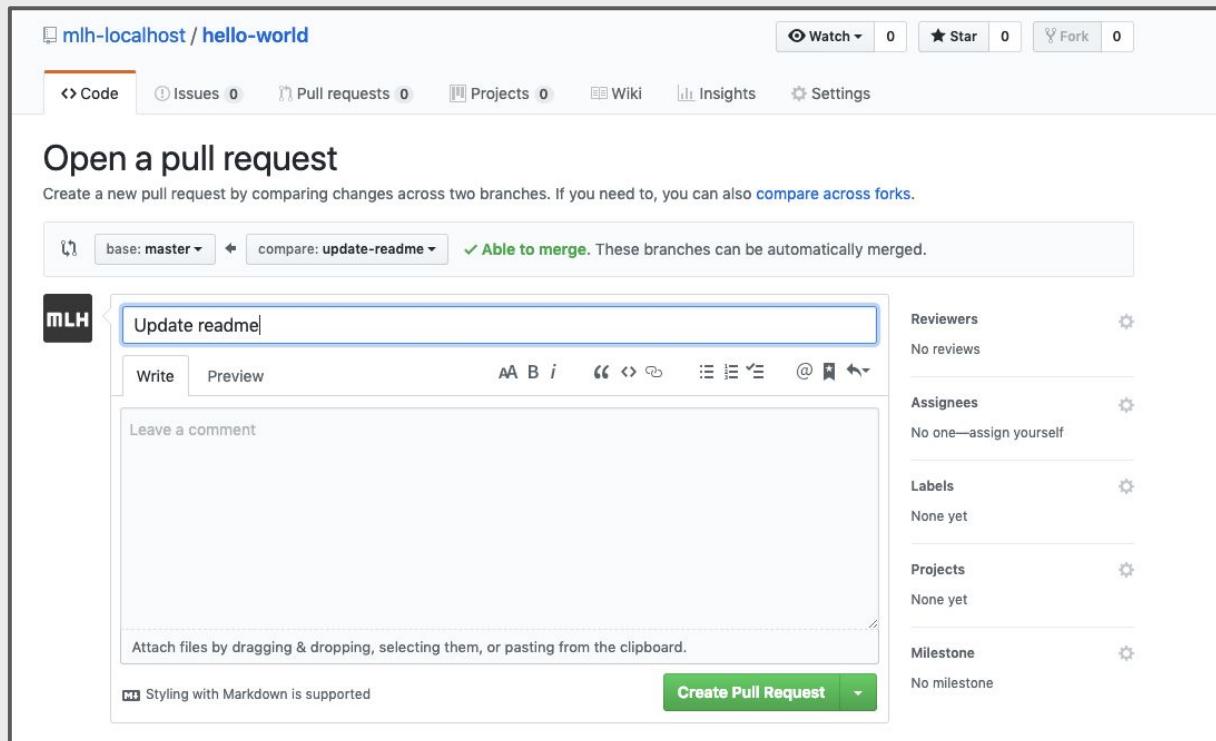
12. Select **Compare & pull request**.

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 'mlh-localhost Initial commit' for 'README.md' at 'Initial commit' on '4 minutes ago'.

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

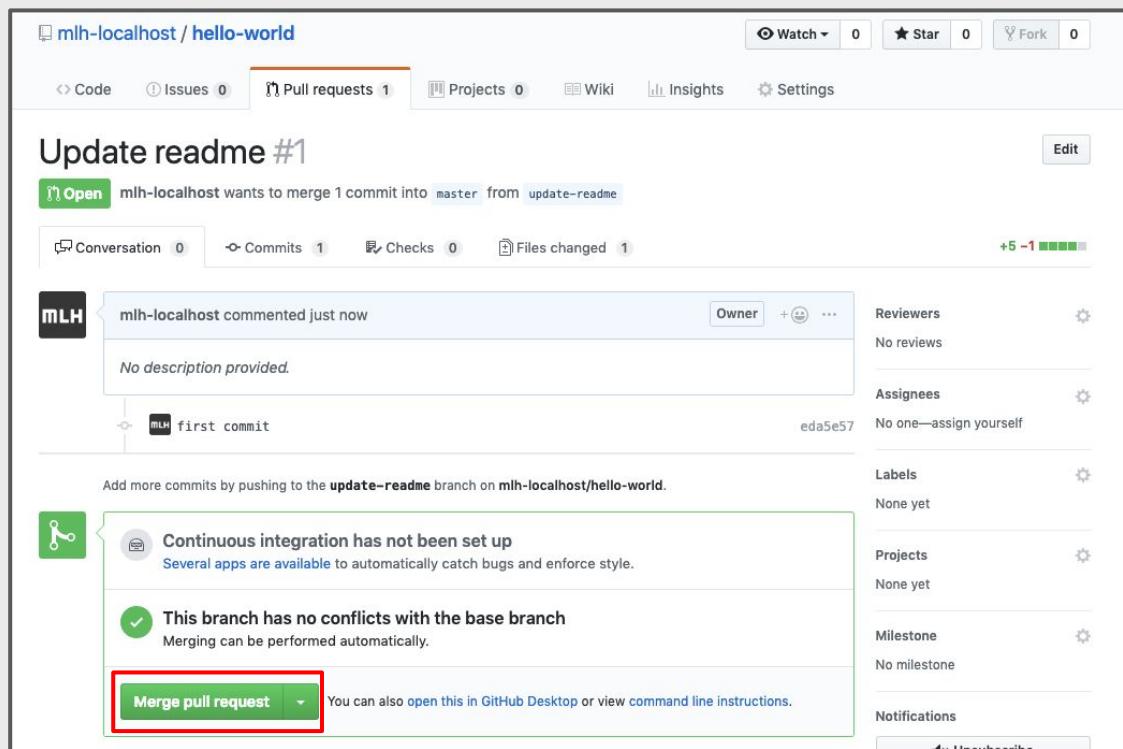
Let's make a change!

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'.
```

A screenshot of a macOS terminal window. The title bar shows the path: "localhost ~ mlh-hackathon-flask-starter/hello-world -bash 89x16". The terminal window contains a command-line session. The command "git checkout master" is entered by the user, followed by its execution message "Switched to branch 'master'". Below that, a status message indicates "Your branch is up to date with 'origin/master'". The line "git checkout master" is highlighted with a red rectangle.

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!

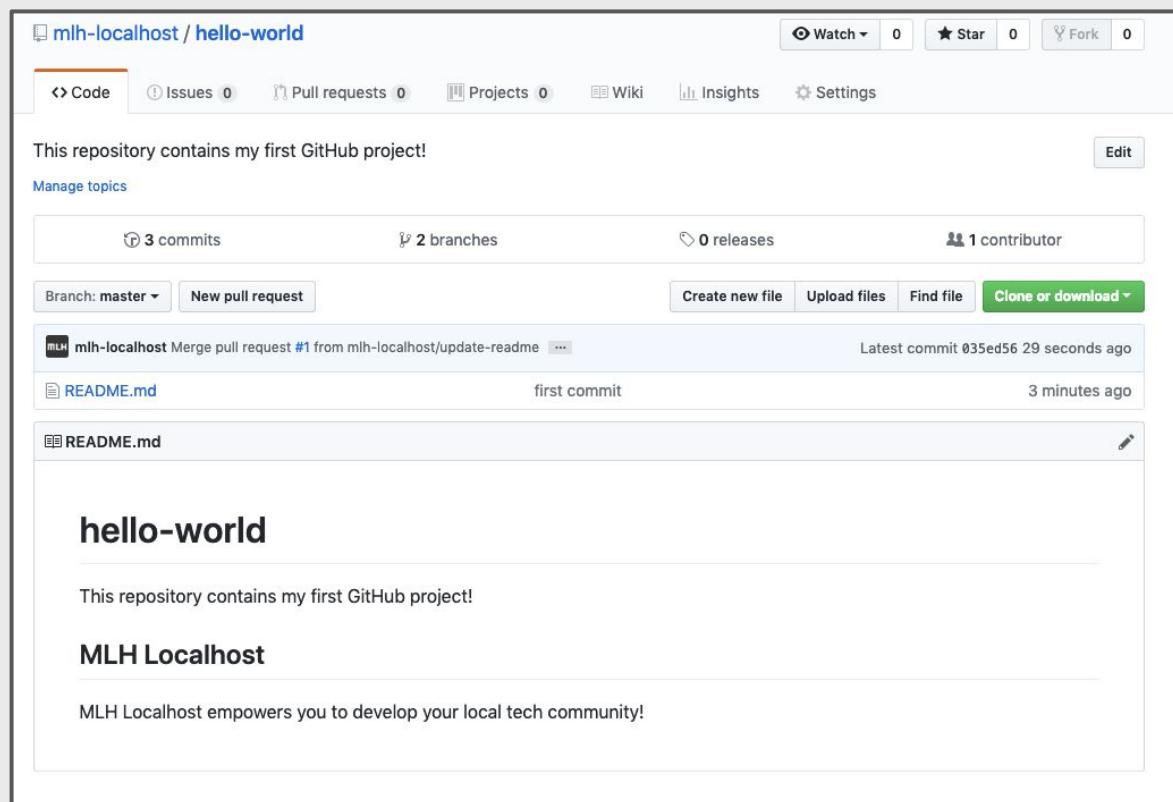


Table of Contents

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

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

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!

A photograph of three people 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 text on it. They appear to be in a workshop setting.

How to Collaborate with GitHub

Workshop

MLH localhost

GitHub