
Workshop
How to Collaborate with GitHub
MLH localhost
GitHub

Roshan **Harish**

We are your Hosts !

MLH

MAJOR LEAGUE HACKING

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!

slido

Audience Q&A Session

Head over to slido.com
And use the event code #62335

ⓘ 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

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

*Graphics taken from the Software Carpentry Git Lesson**

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

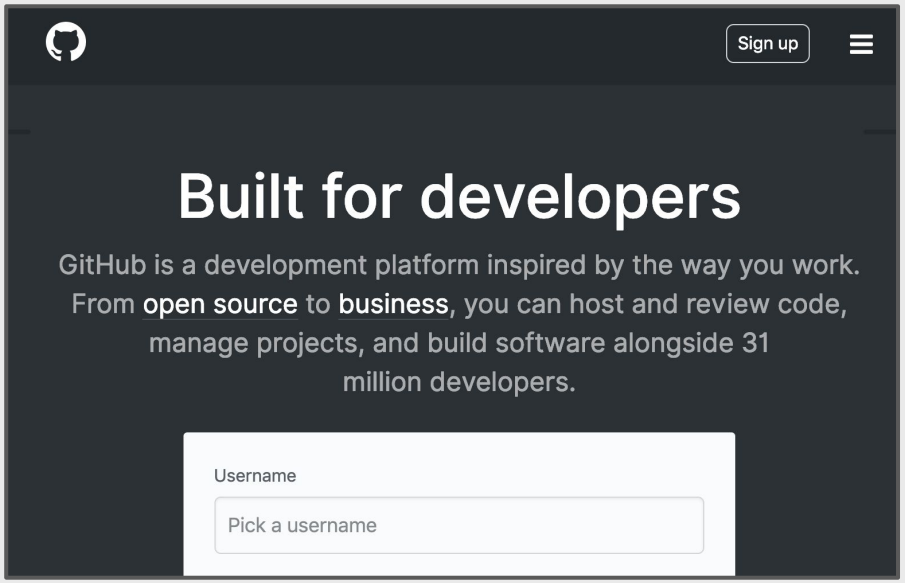
- GitHub is a platform for code collaboration!
- GitHub uses Git for version control and provides you all sorts of awesome collaboration tools.

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!

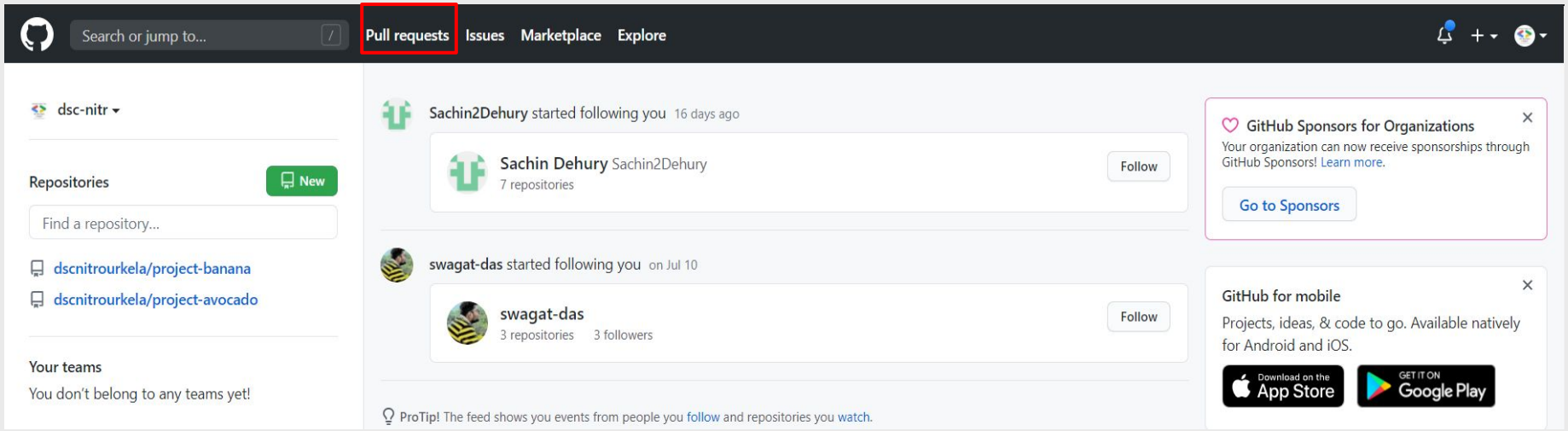
http://mlhlocal.host/github-edu

http://mlhlocal.host/github-signup



Explore GitHub

The GitHub platform provides numerous features for collaboration.



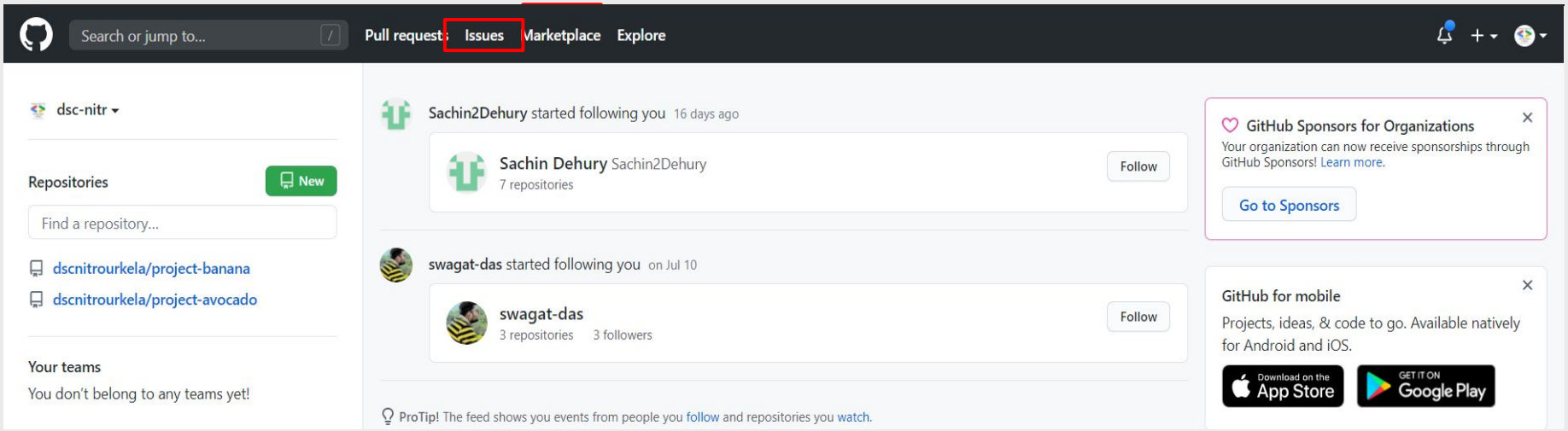
Next to the search bar, you can select Pull requests. This will show you your own pull requests against other people's repos.

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.



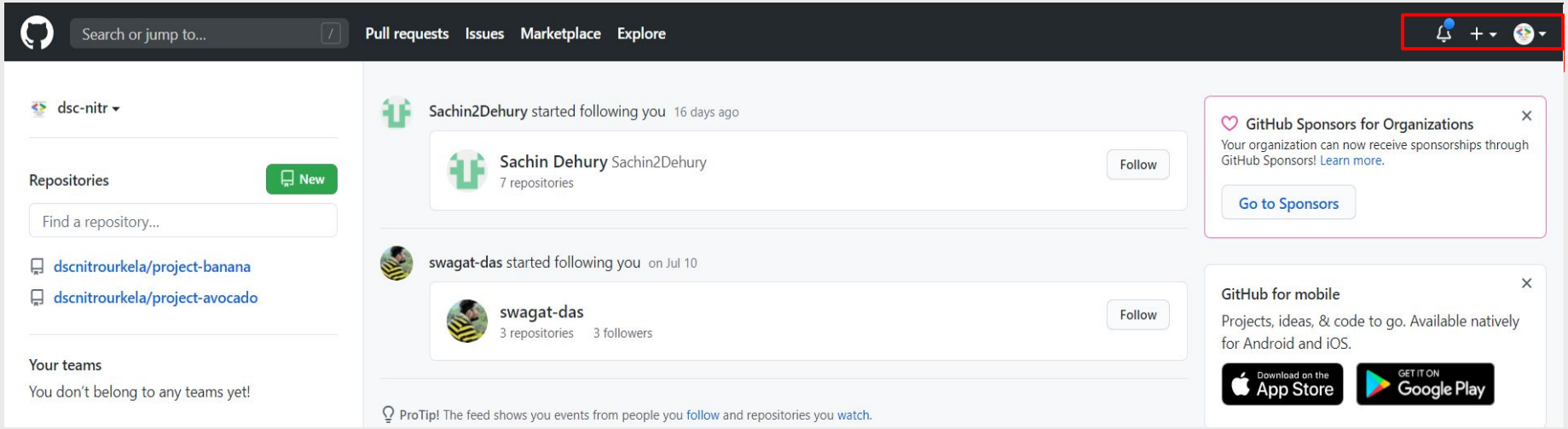
Next to pull requests, you can see any issues you've opened or worked on.

Key Term

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

Explore GitHub

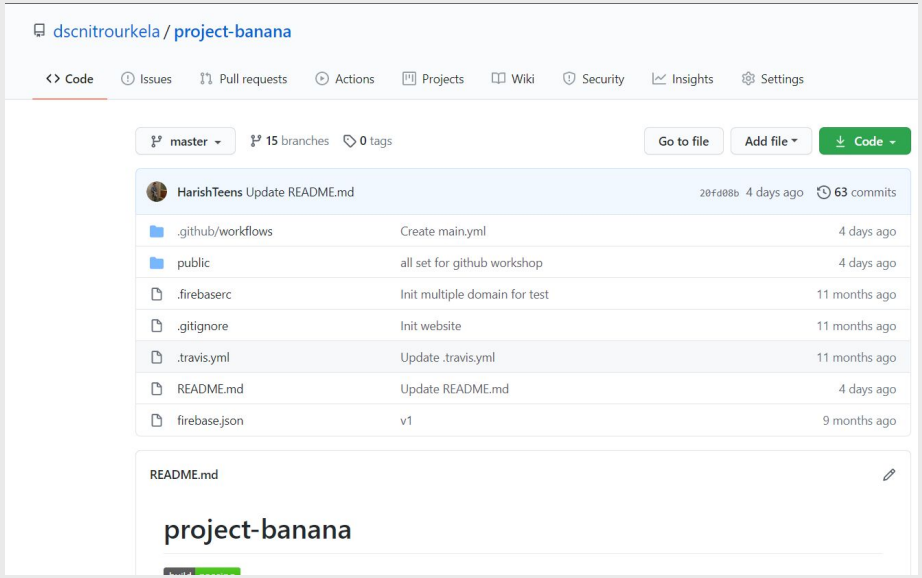
The GitHub platform provides numerous features for collaboration.



- 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!



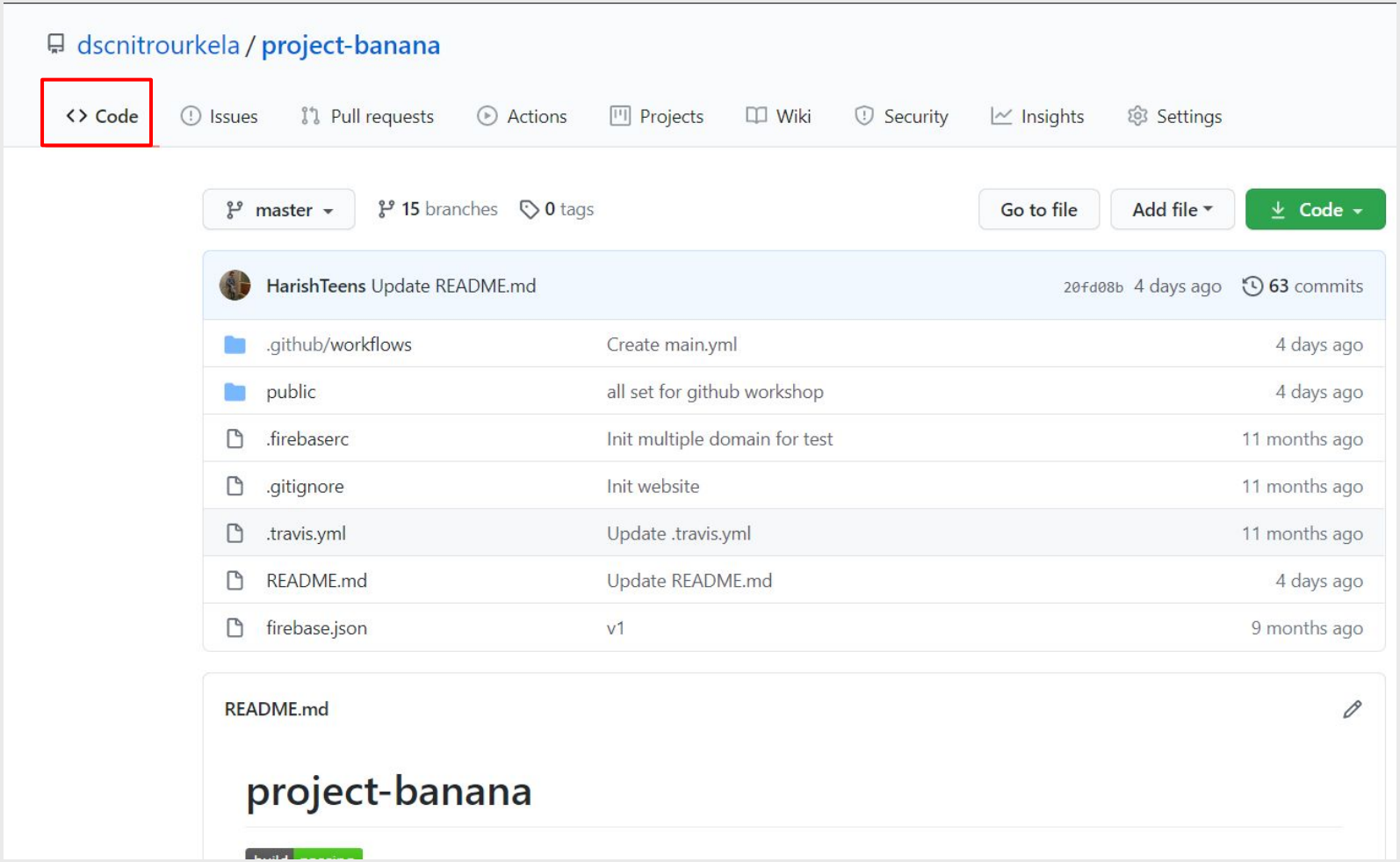
project-banana

Key Term

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

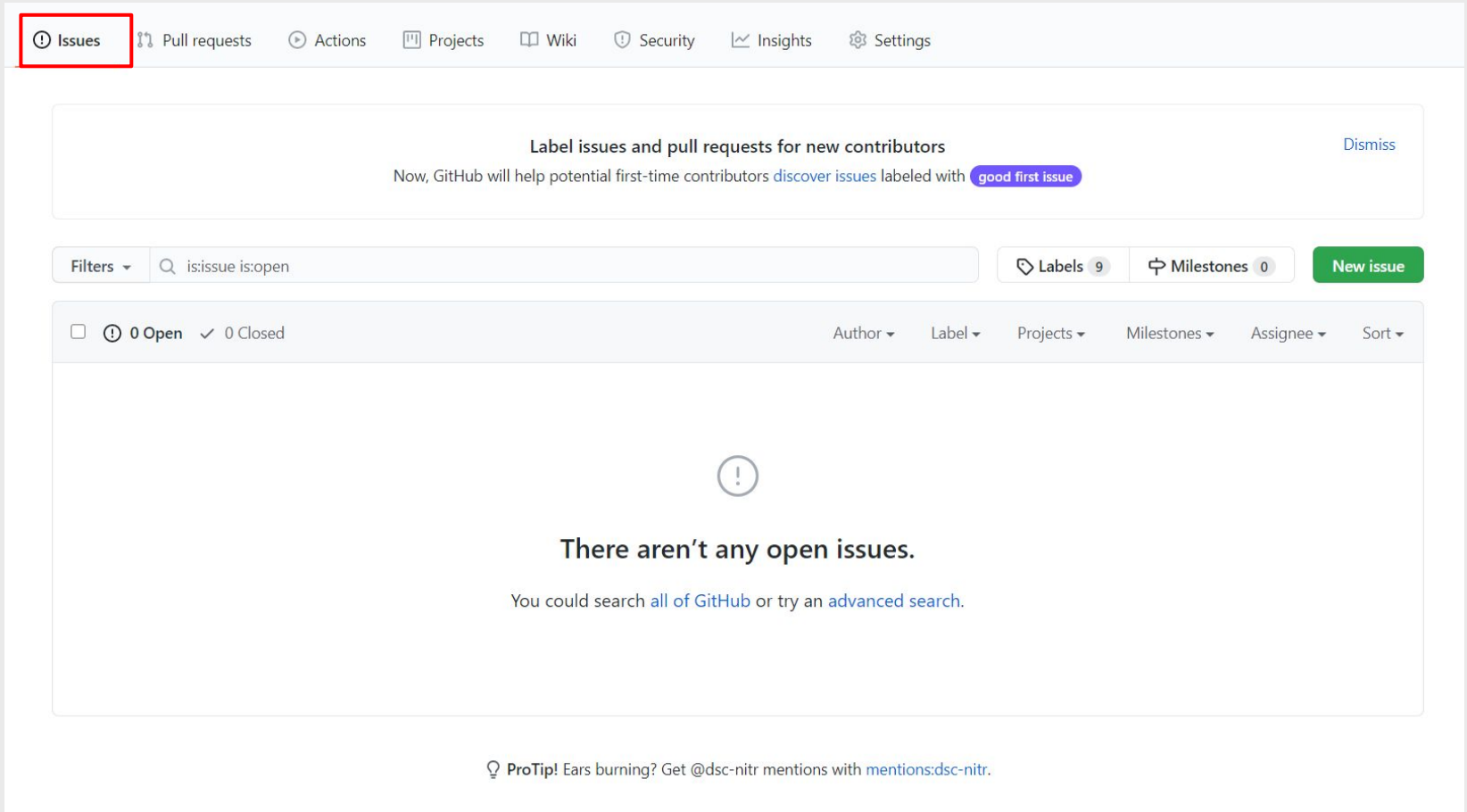
Explore a Repository

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



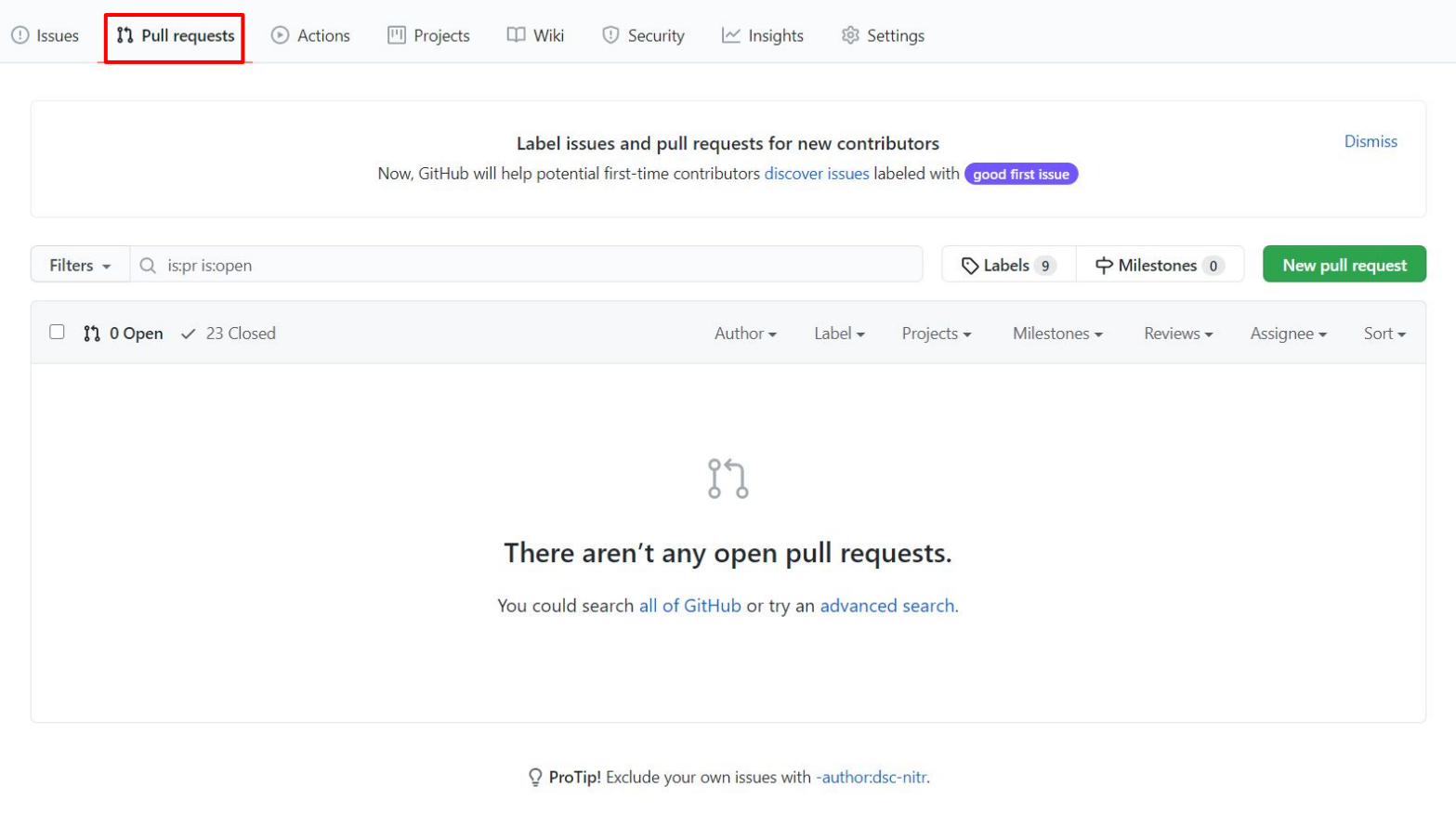
Explore a Repository

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

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.

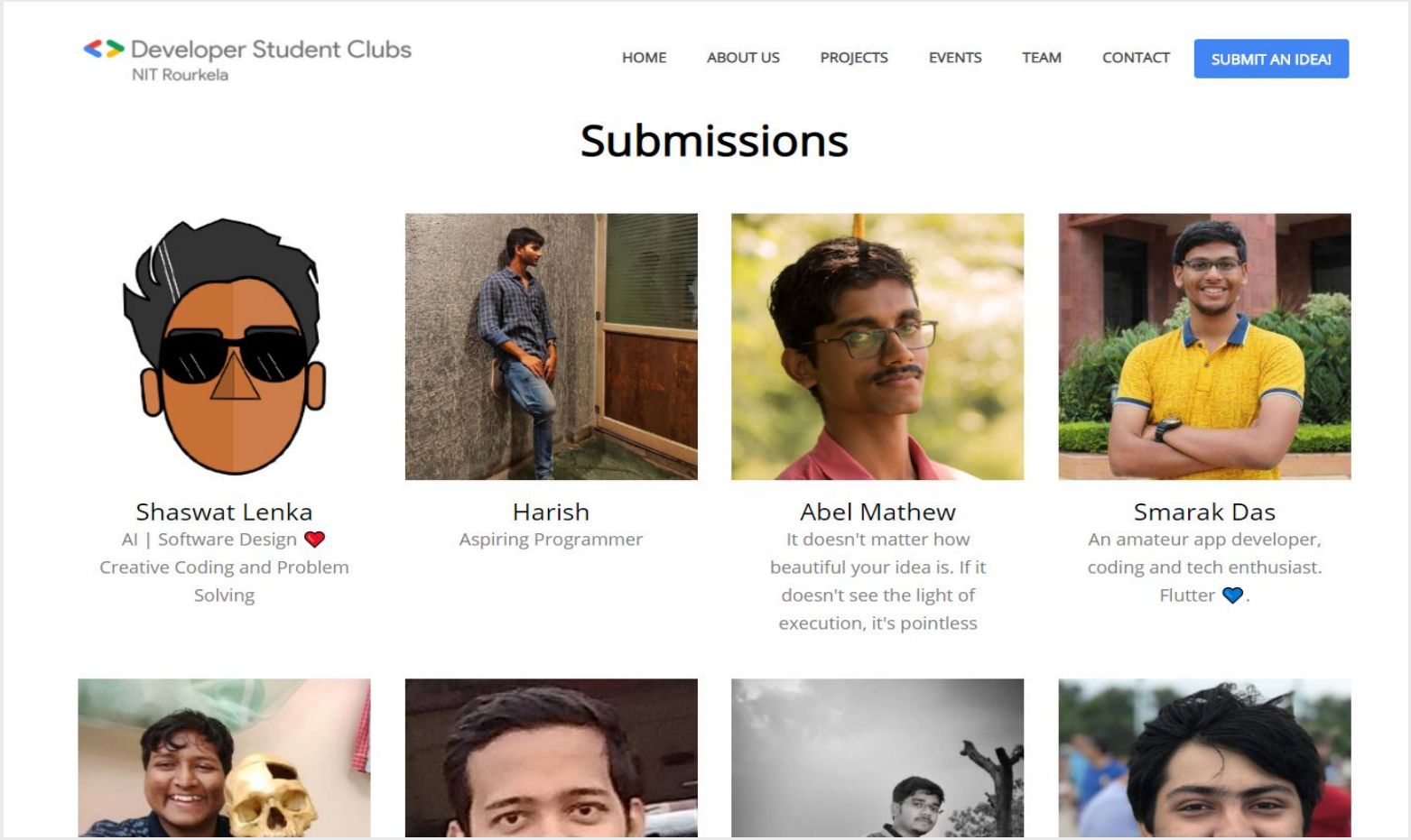
Key Terms
stage: add to a cohesive group/bundle of revisions
commit: a group of revisions you want to officially add to your branch
merge: to officially add the changes from your branch into the master branch (or another branch)

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



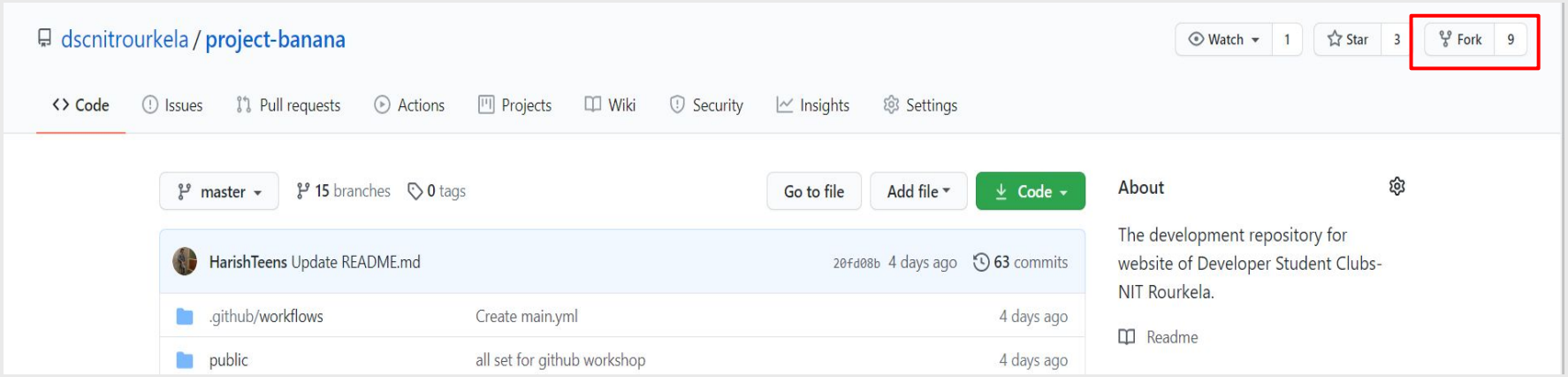
Activity Workflow

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-

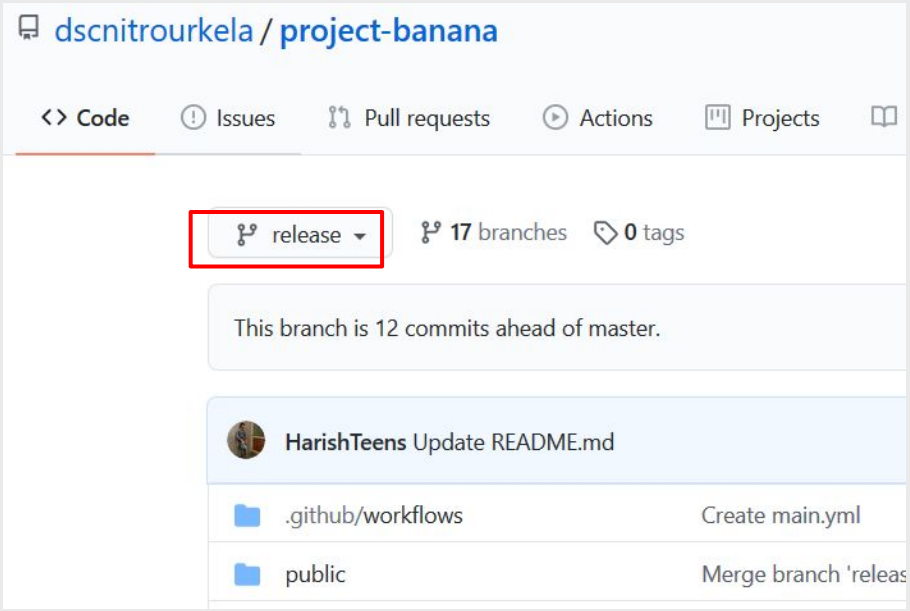
https://github.com/dscnitrourkela/project-banana

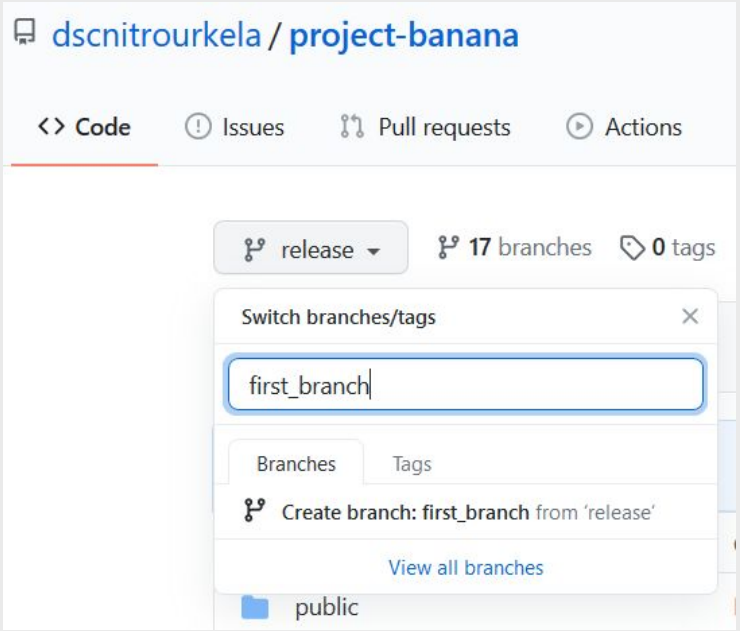
1. **Fork** the repository you want to contribute to.



Activity Workflow

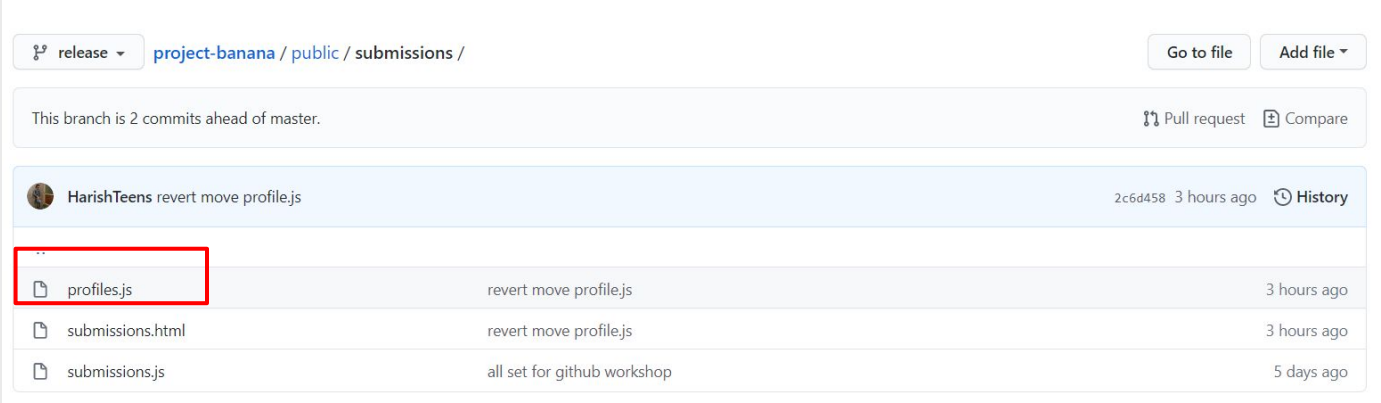
2. Go to the **Release branch** then Create a **branch**. You should always try to name branches with your name and what you're doing, or follow the conventions set by your team.



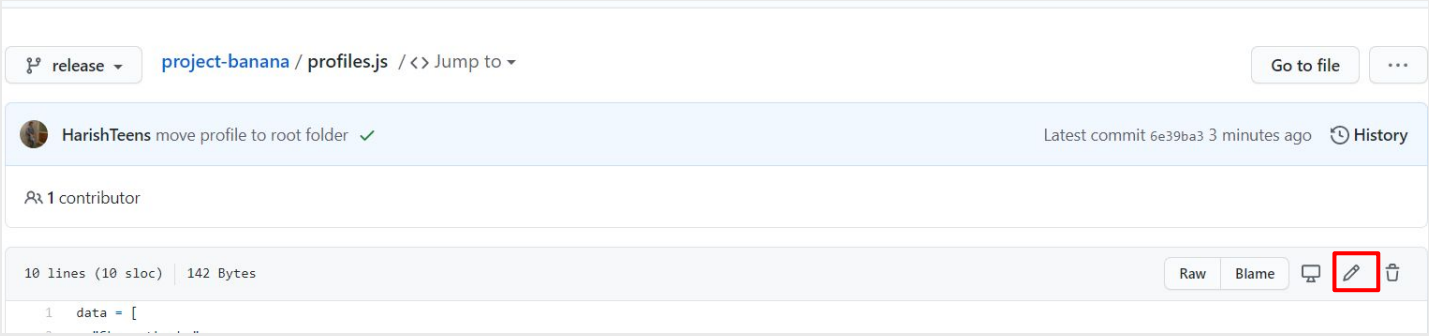


Activity Workflow

3. Open `profiles.js`

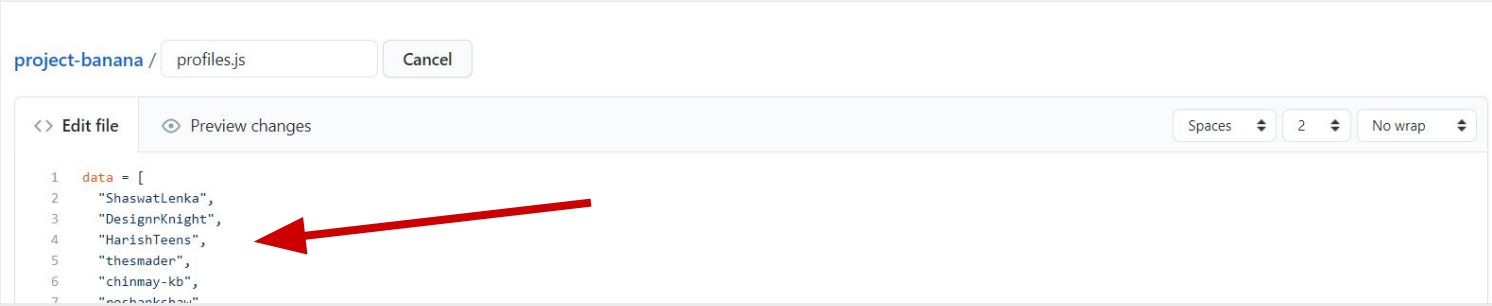


4. Select the edit icon. It looks like a pencil.

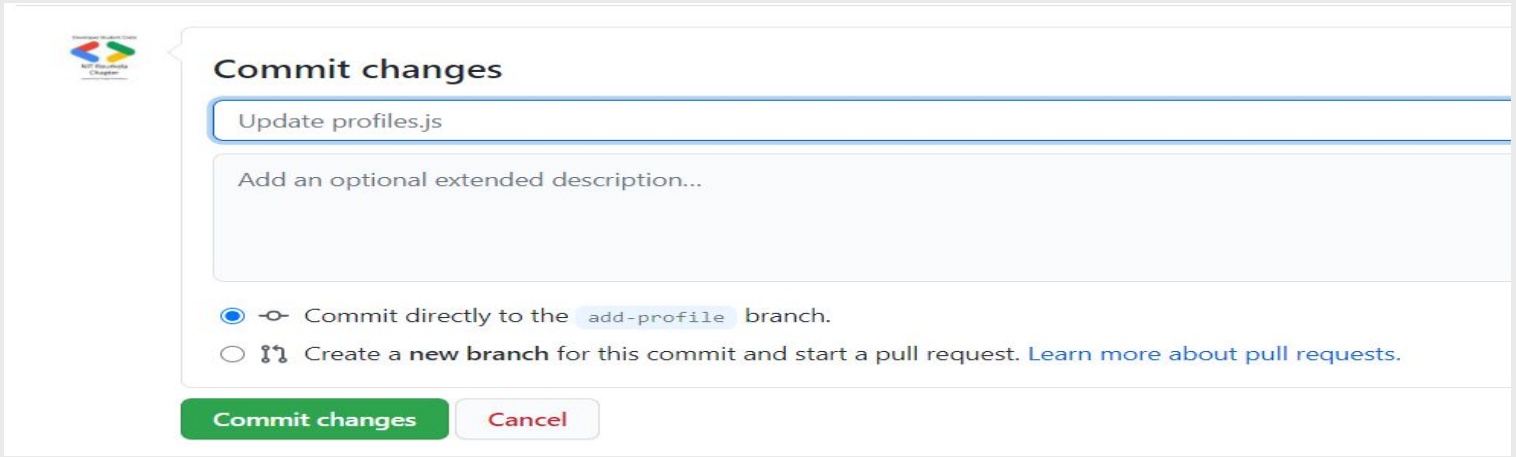


Activity Workflow

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

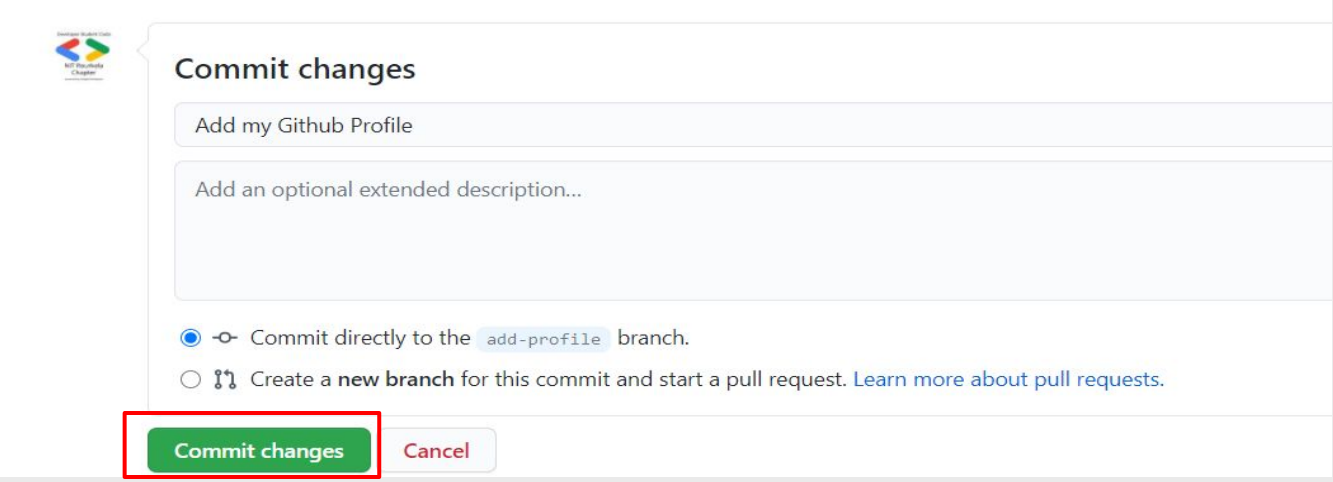


6. Scroll to the bottom to the Commit changes section.



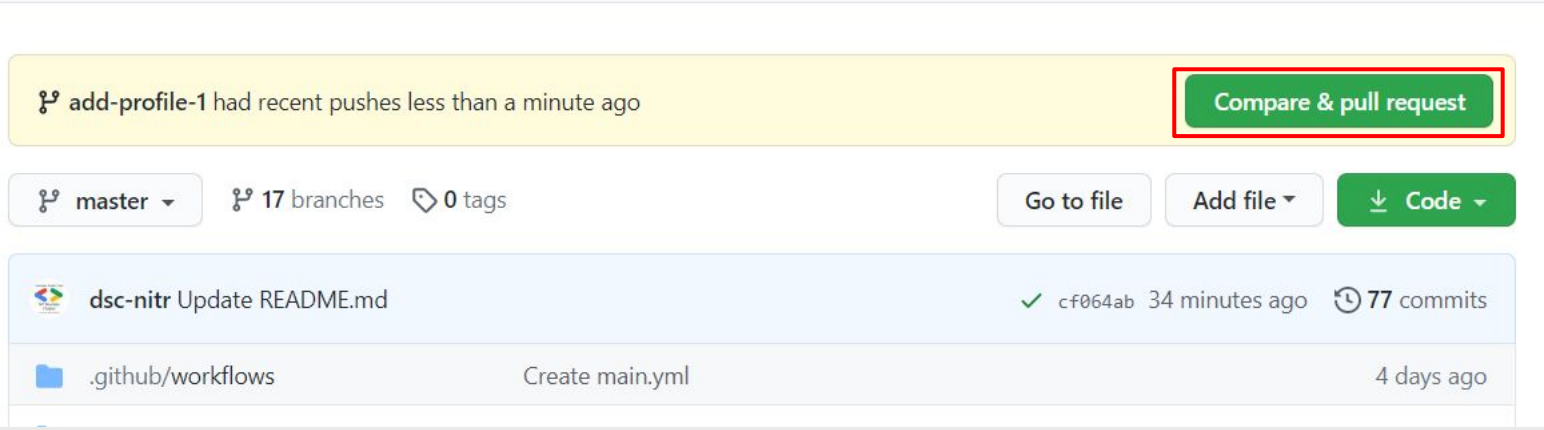
Activity Workflow

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



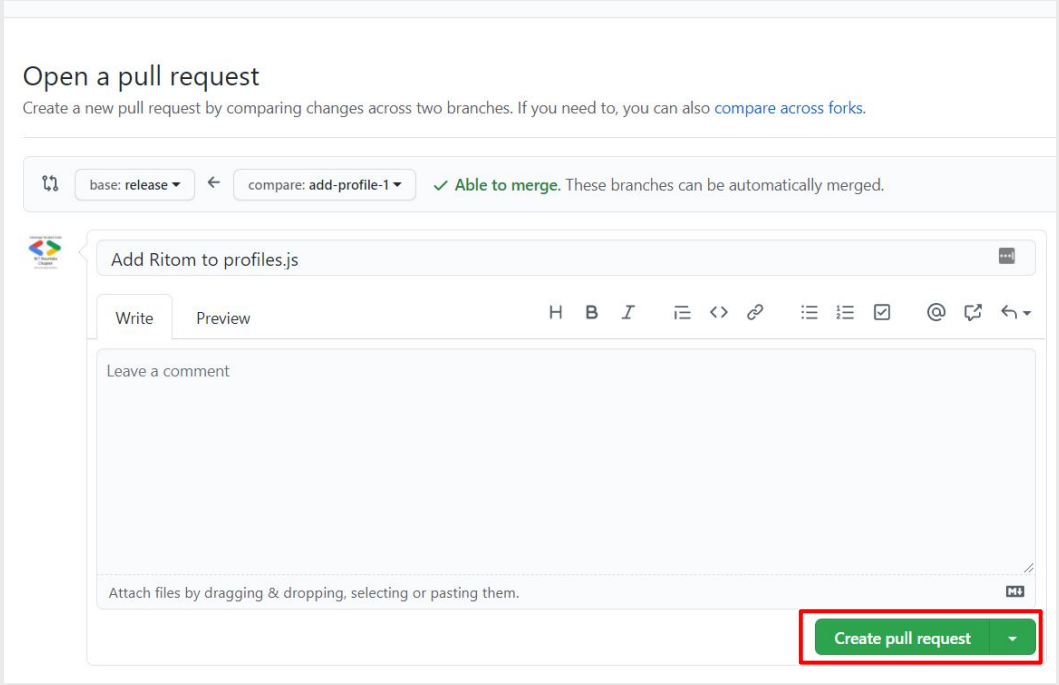
Activity Workflow

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



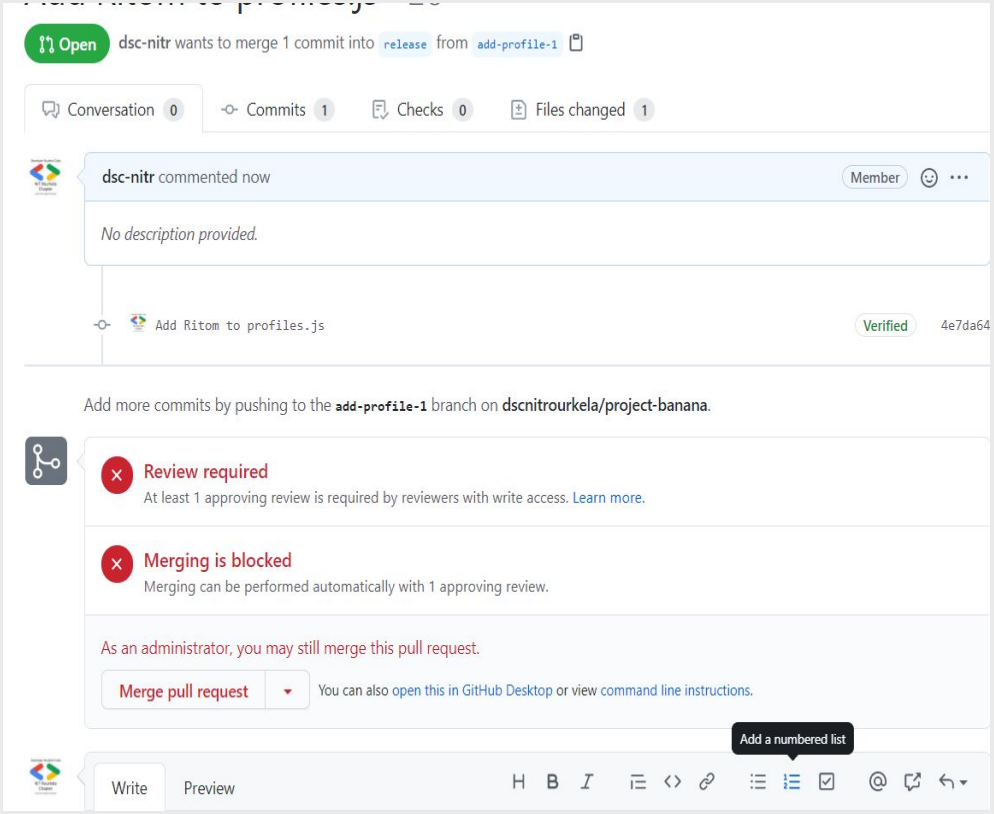
Activity Workflow

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.



Activity Workflow

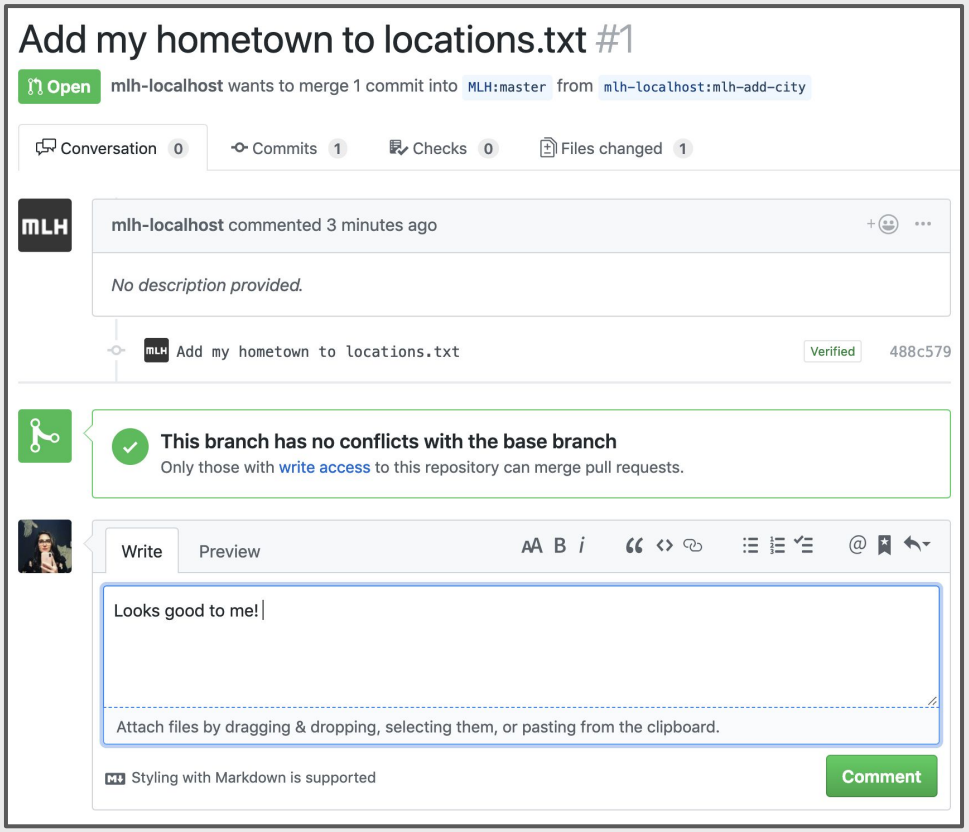
11. Wait for your pull request to be reviewed by a reviewer.The reviewer is usually the owner or a maintainer of the repository.



Activity Workflow

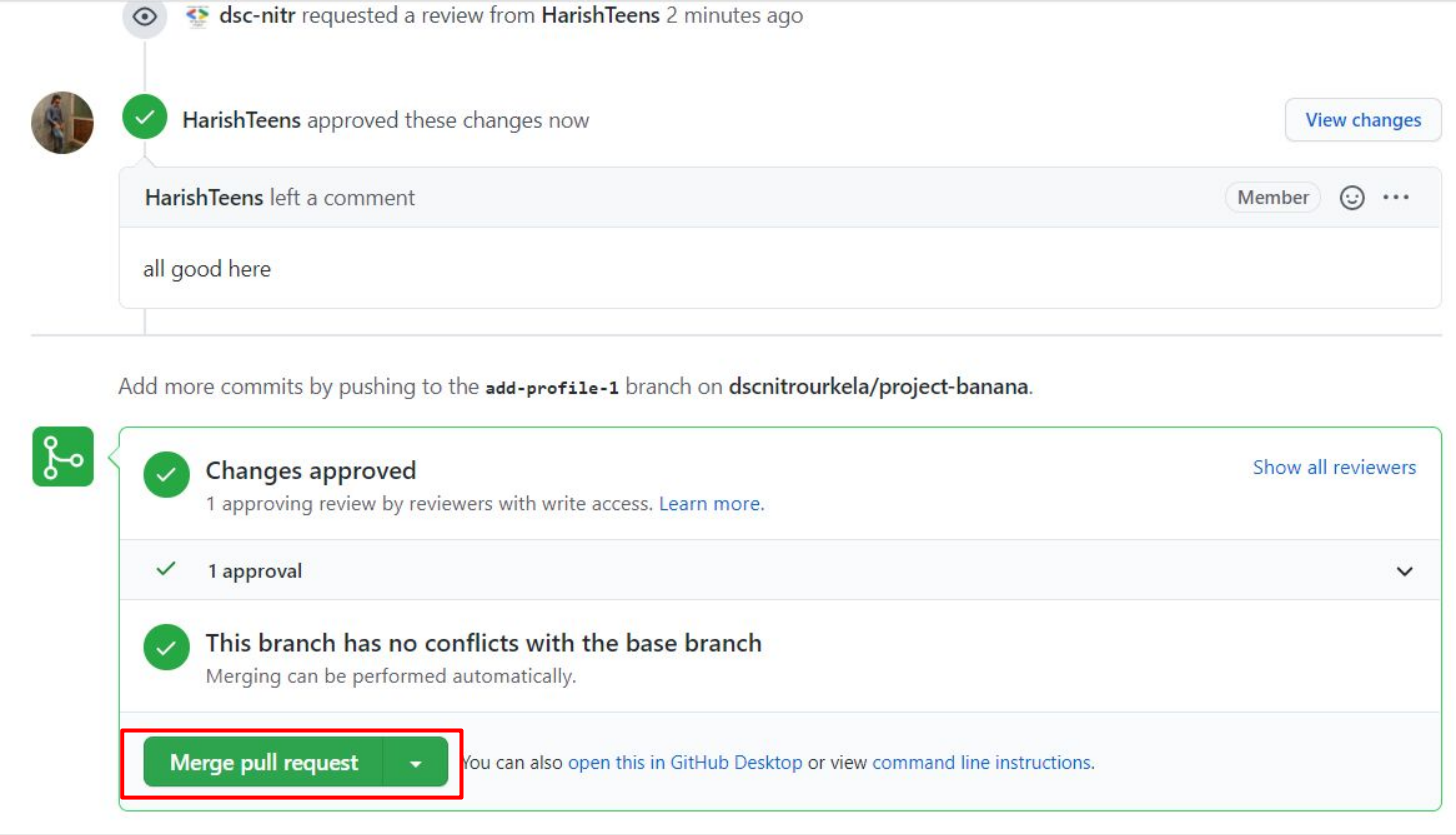
12. Ask someone else to comment on your PR. Comment on someone else's PR, too!

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



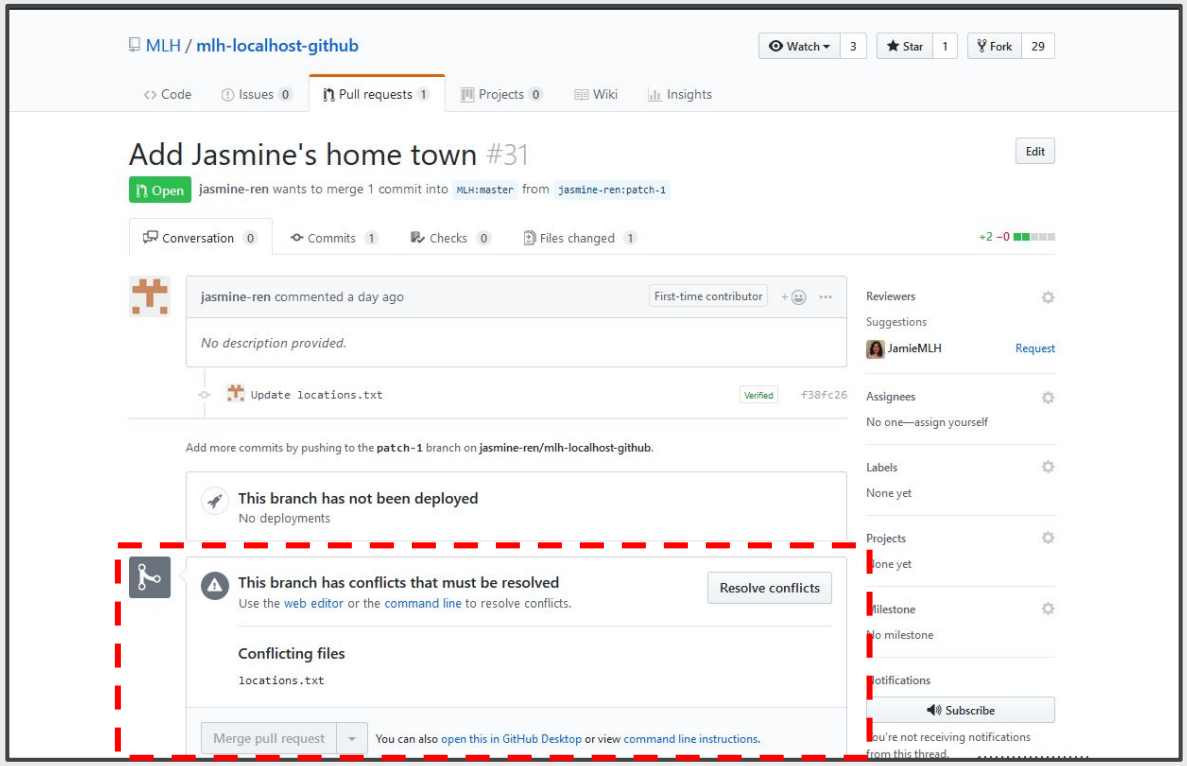
Activity Workflow

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!



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.

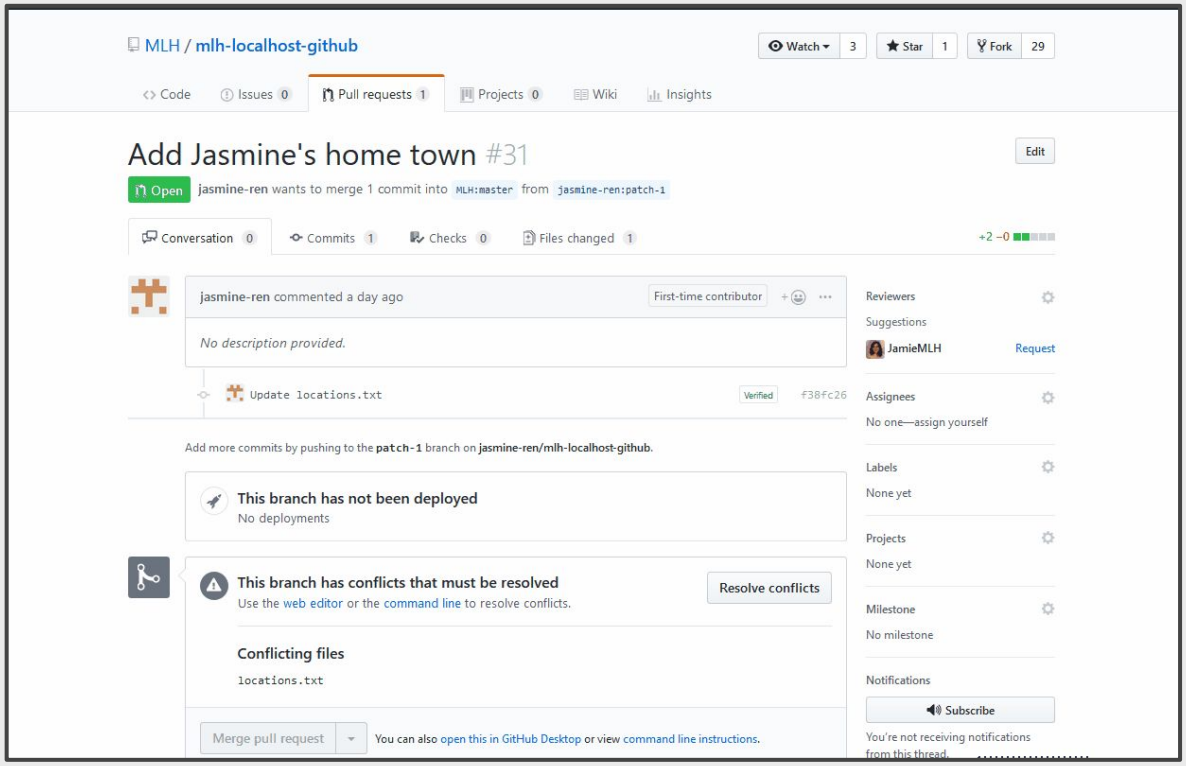


Conflict Resolution

A conflict is marked by

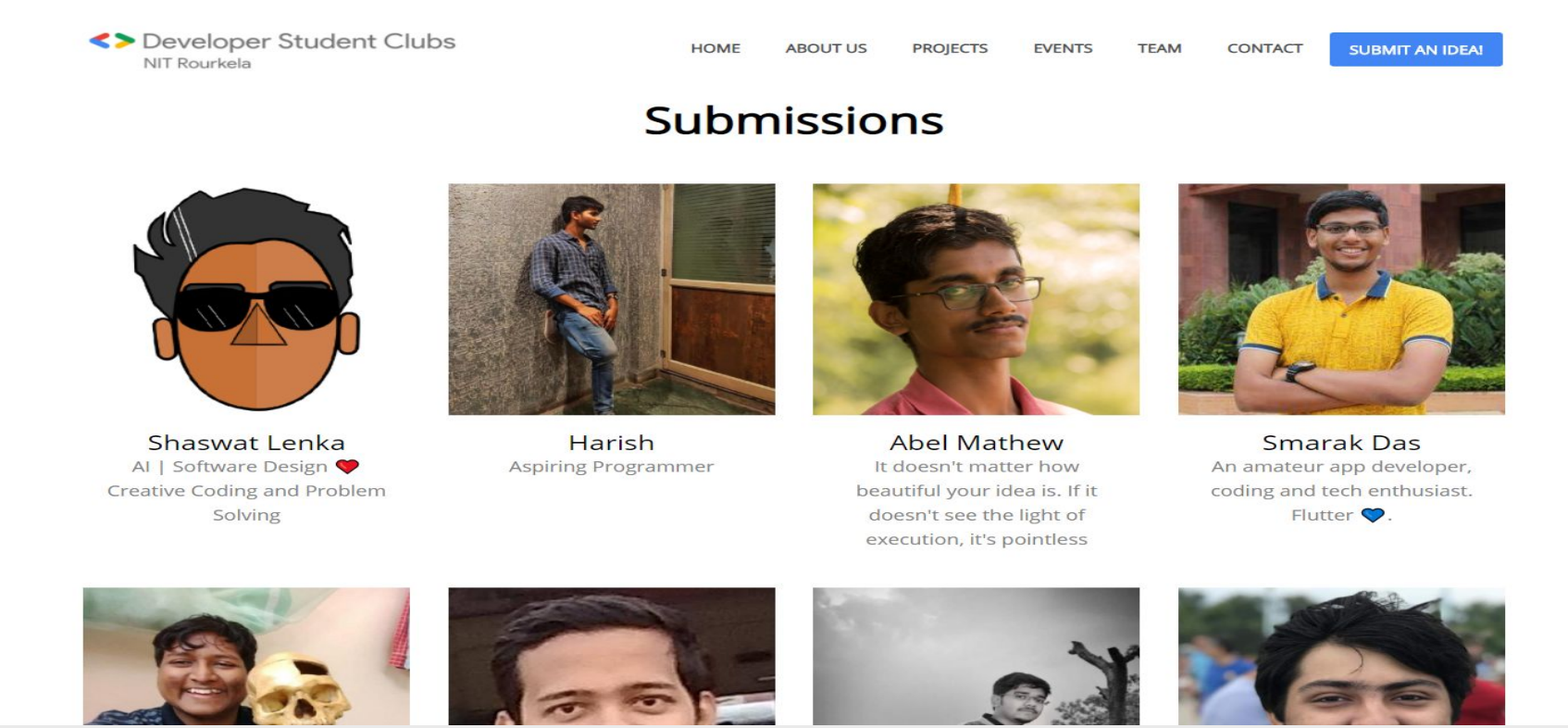
```
<<<<<<< new branch

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.



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?

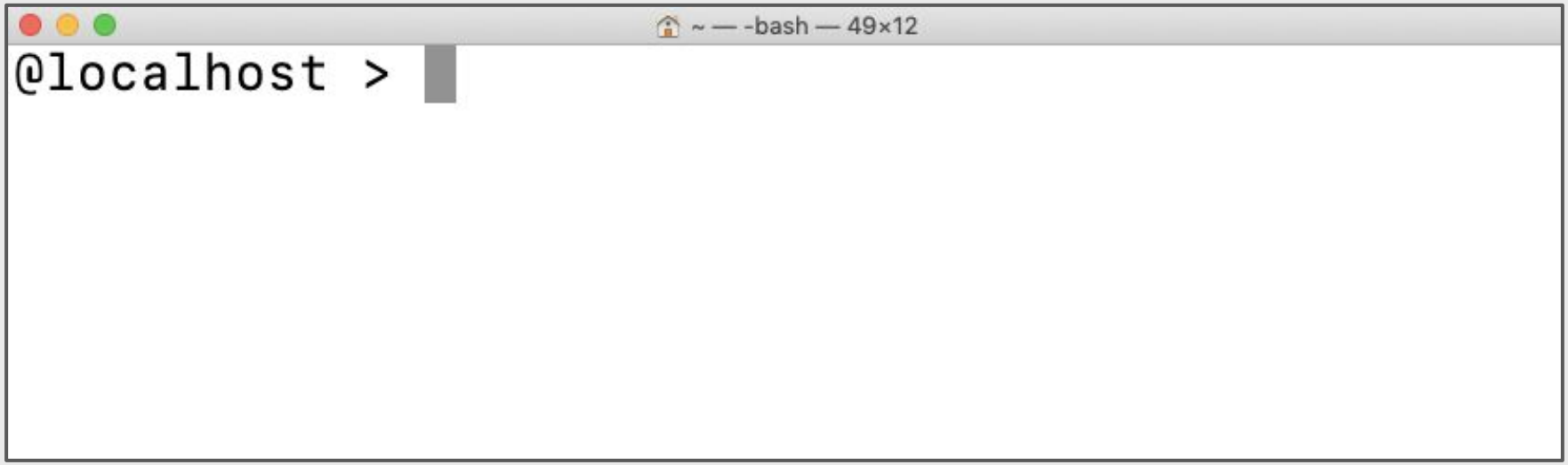
We created a fun quiz to test your knowledge and see what you learned from this workshop.

http://mlhlocal.host/quiz

Table of Contents

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!

Install Git

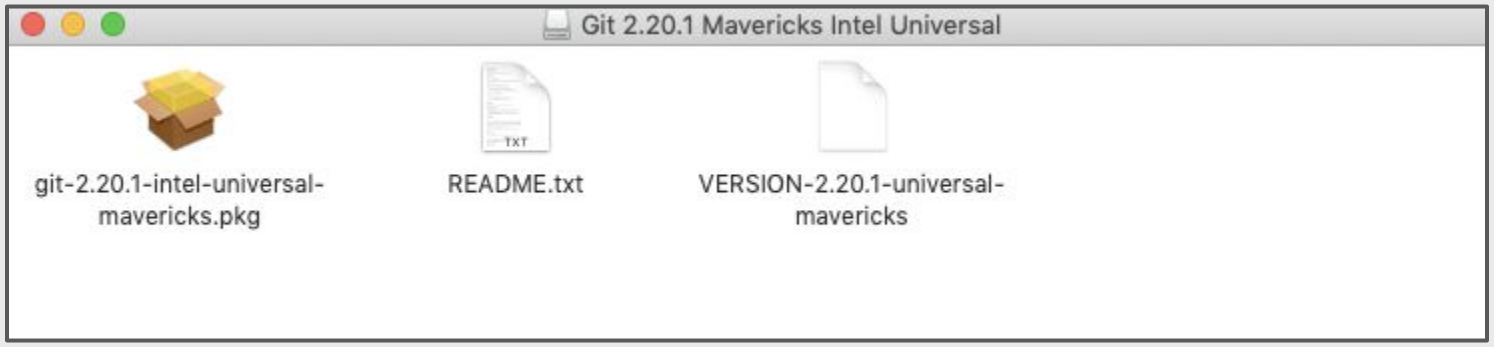
http://mlhlocal.host/install-git

2. If you do not have git installed, navigate to the URL above.

3. Select your operating system and the installer will download.

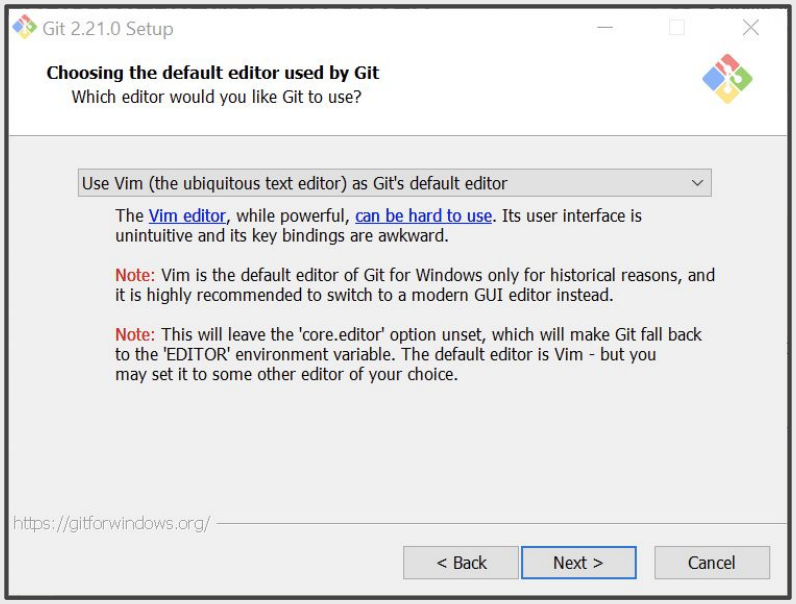
Install Git: MacOS

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.

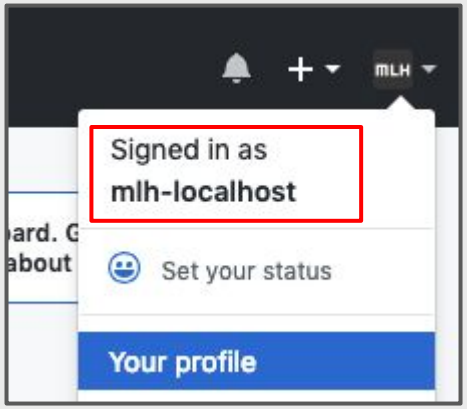
6. Run the command below again. Raise your hand if this is not what you see.

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

Configure Git

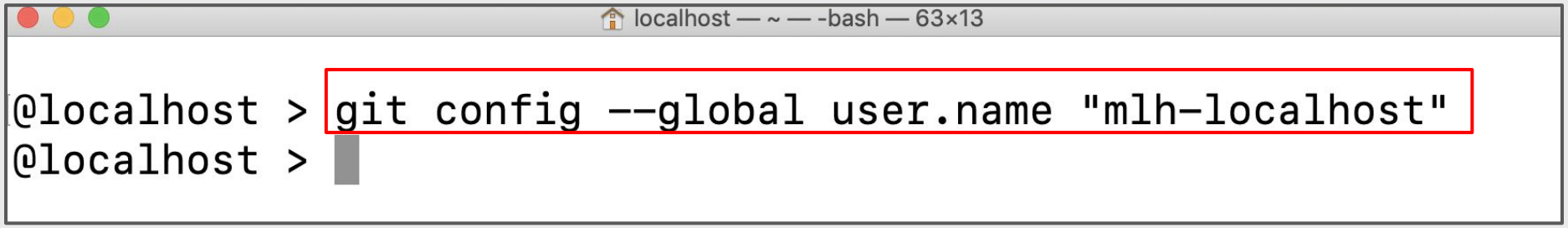
We want to configure our local environments so that the correct GitHub account is associated with our commits.

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"

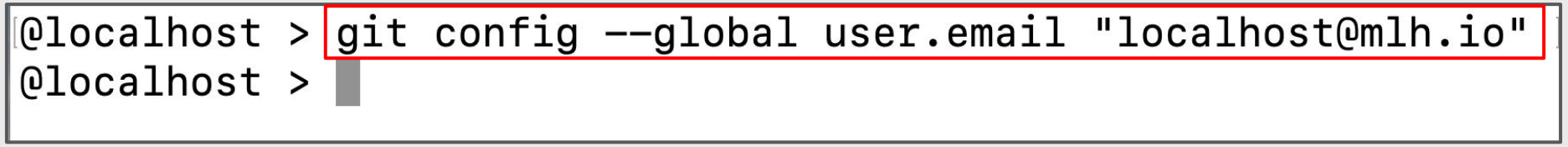


Configure Git

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



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



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.

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.

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

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

Three phases of Git

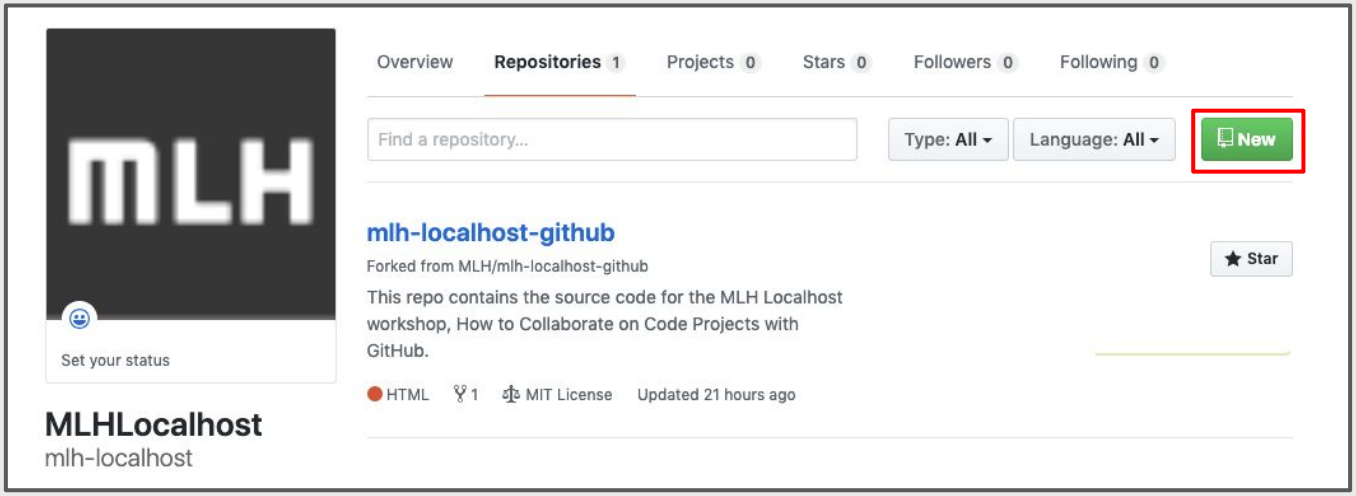
Use the staging area to build a commit

Great! Let's try it.

Create a Repo

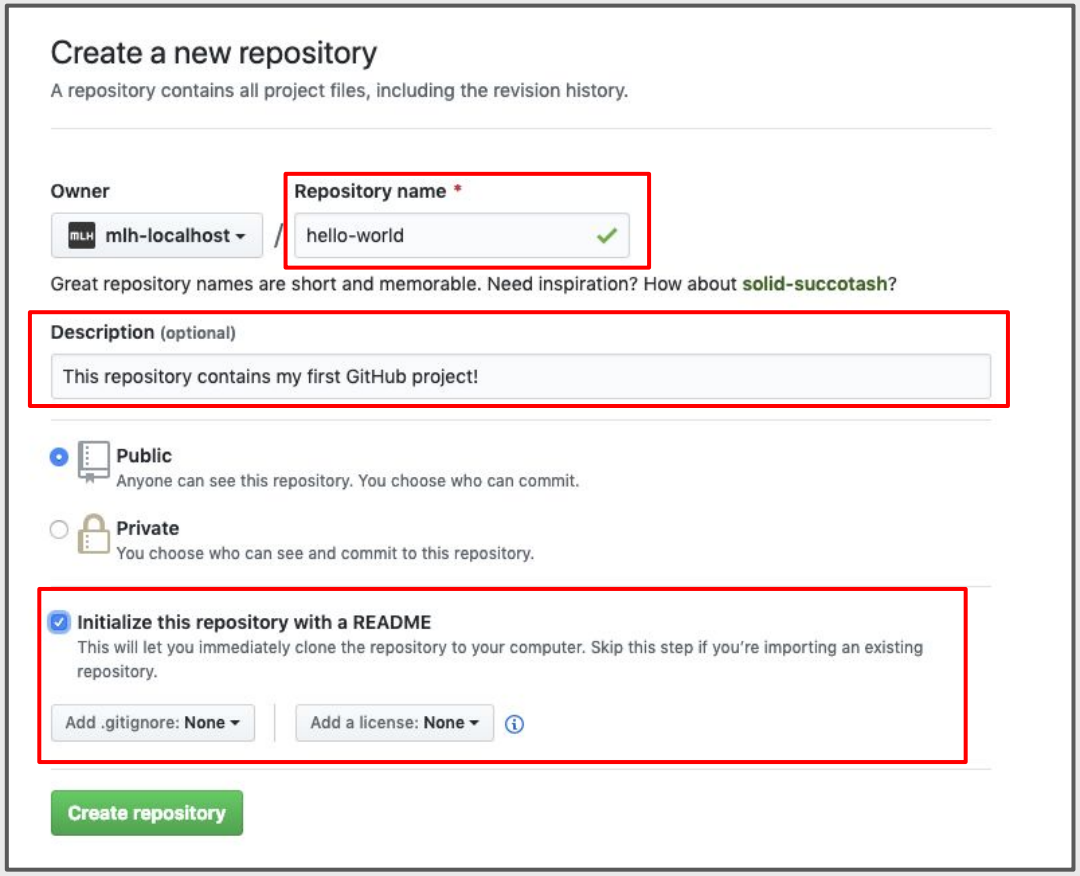
Let's create a new repository on GitHub and work with it locally.

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



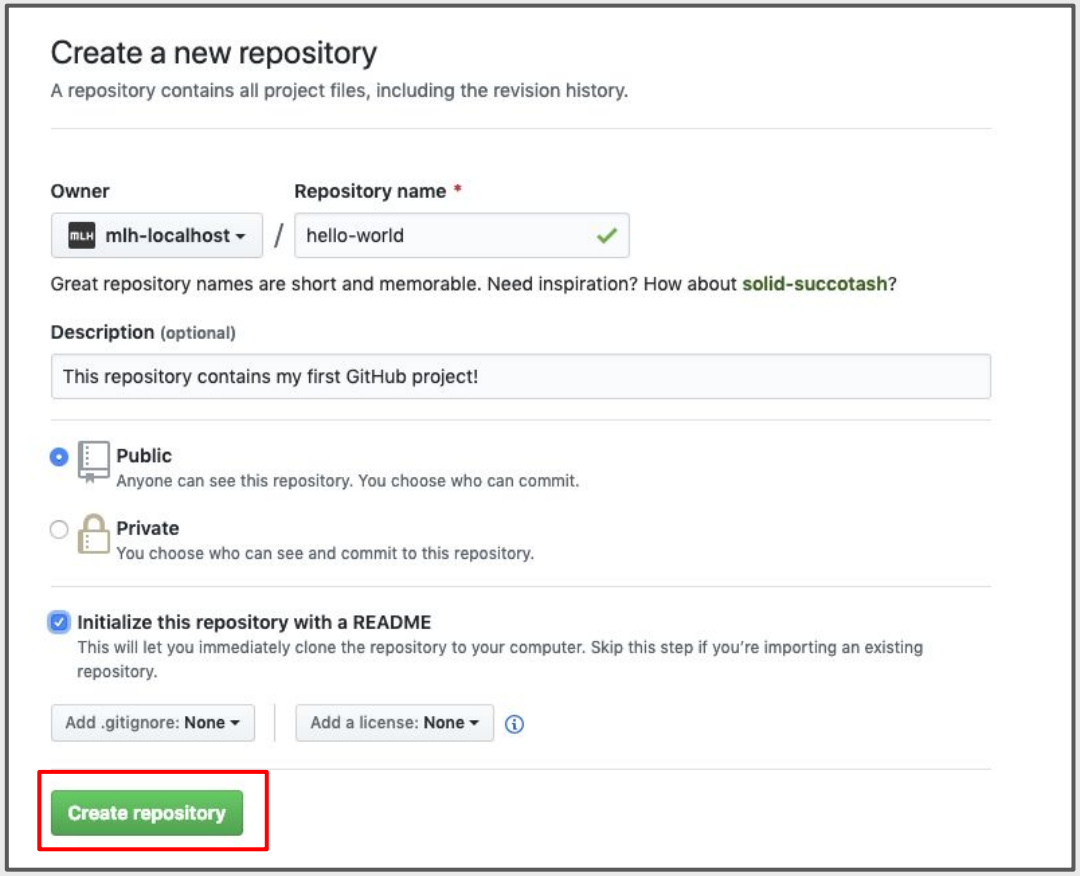
Create a Repo

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**.



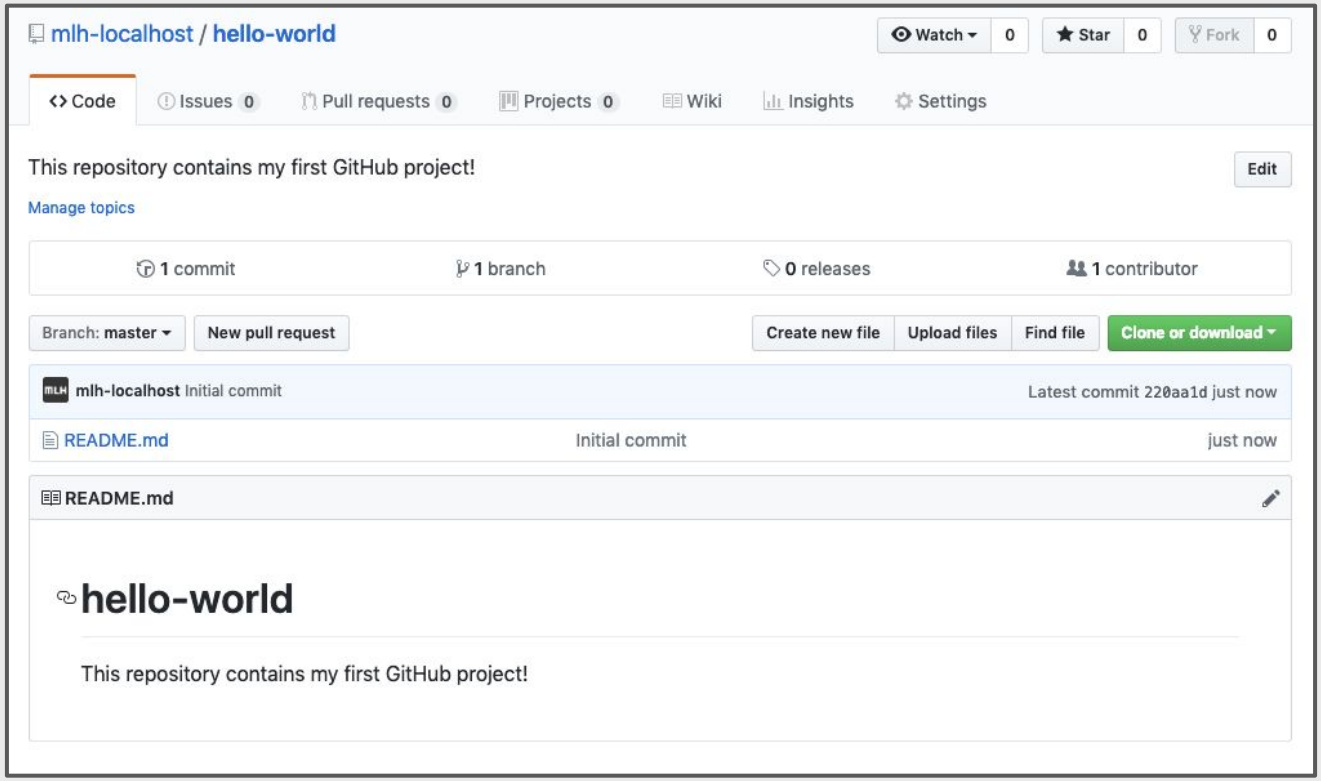
Create a Repo

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.**



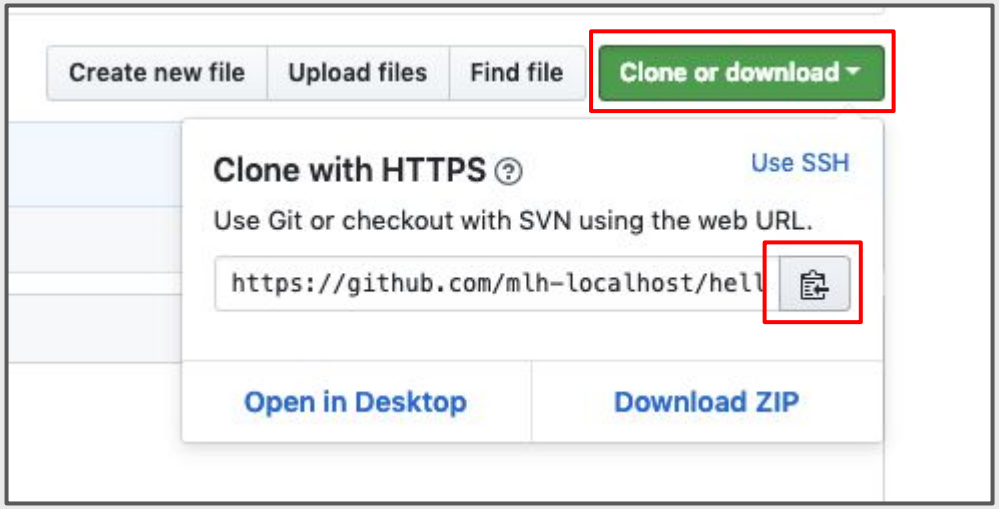
Create a Repo

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



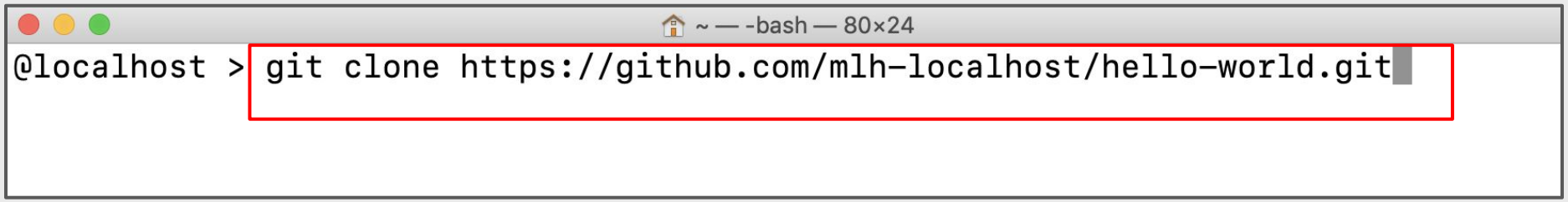
Clone Your Repo

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

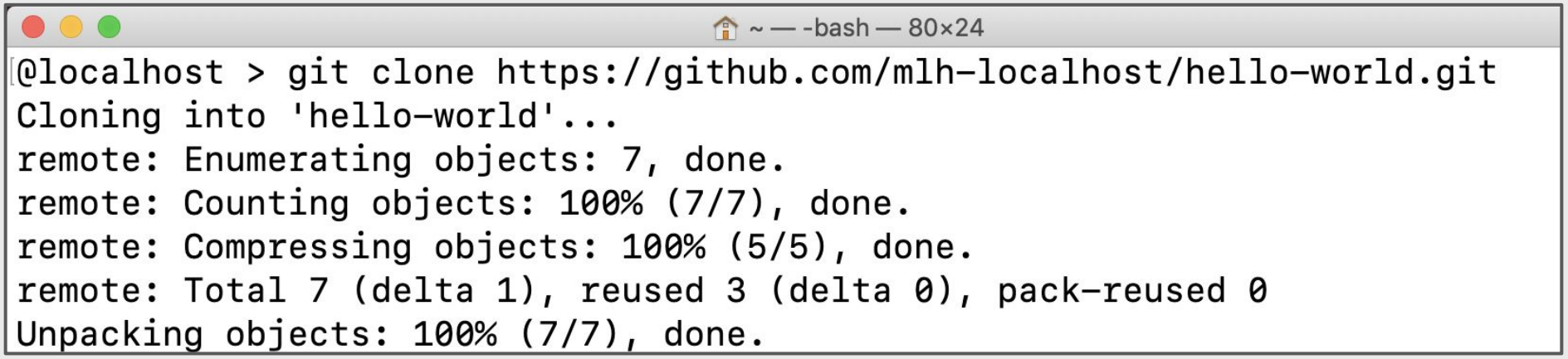


Clone Your Repo

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



You will see output like this:



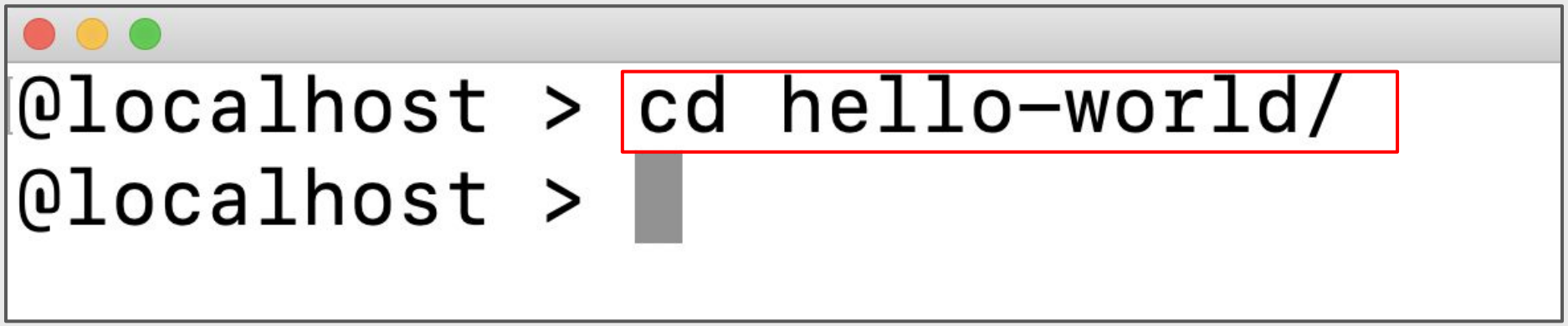
Explore Your Repo

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.

Let's make a change!

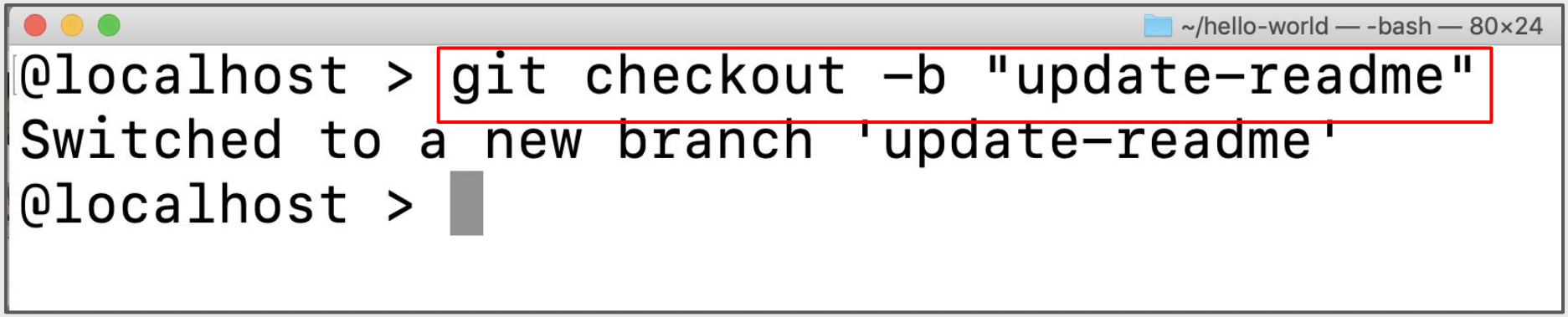
1. Change directory into the repo you cloned by typing the command below.

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



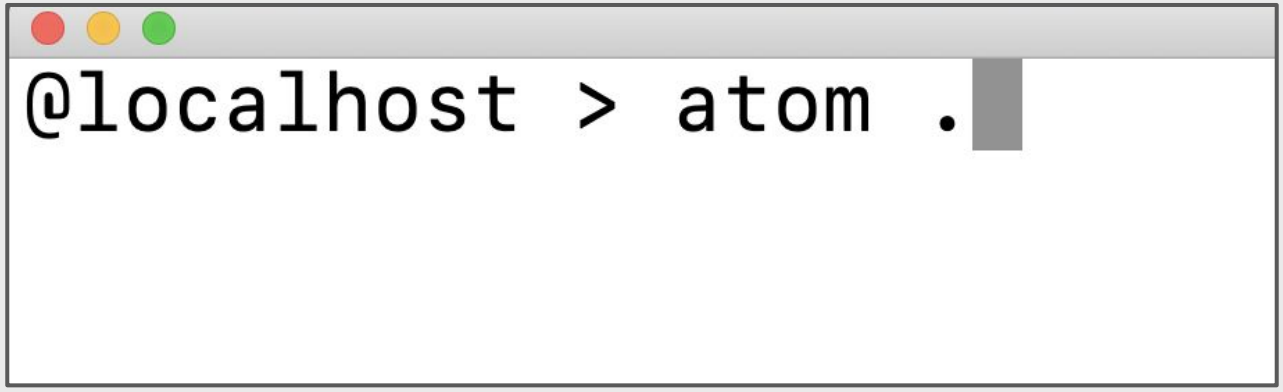
Let's make a change!

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.



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

```
# hello-world
This repository contains my first GitHub project!

## MLH Localhost

MLH Localhost empowers you to develop your local tech community!

```

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.

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!

8. 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!

9. 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':
```

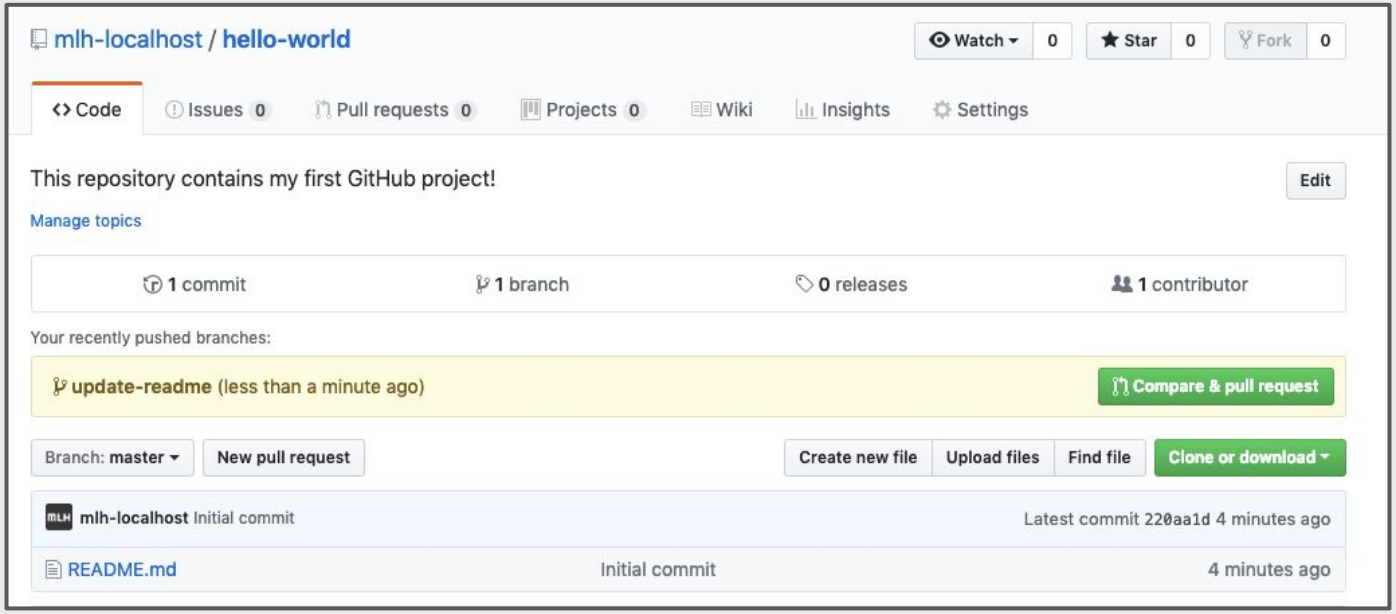
Let's make a change!

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 >
```

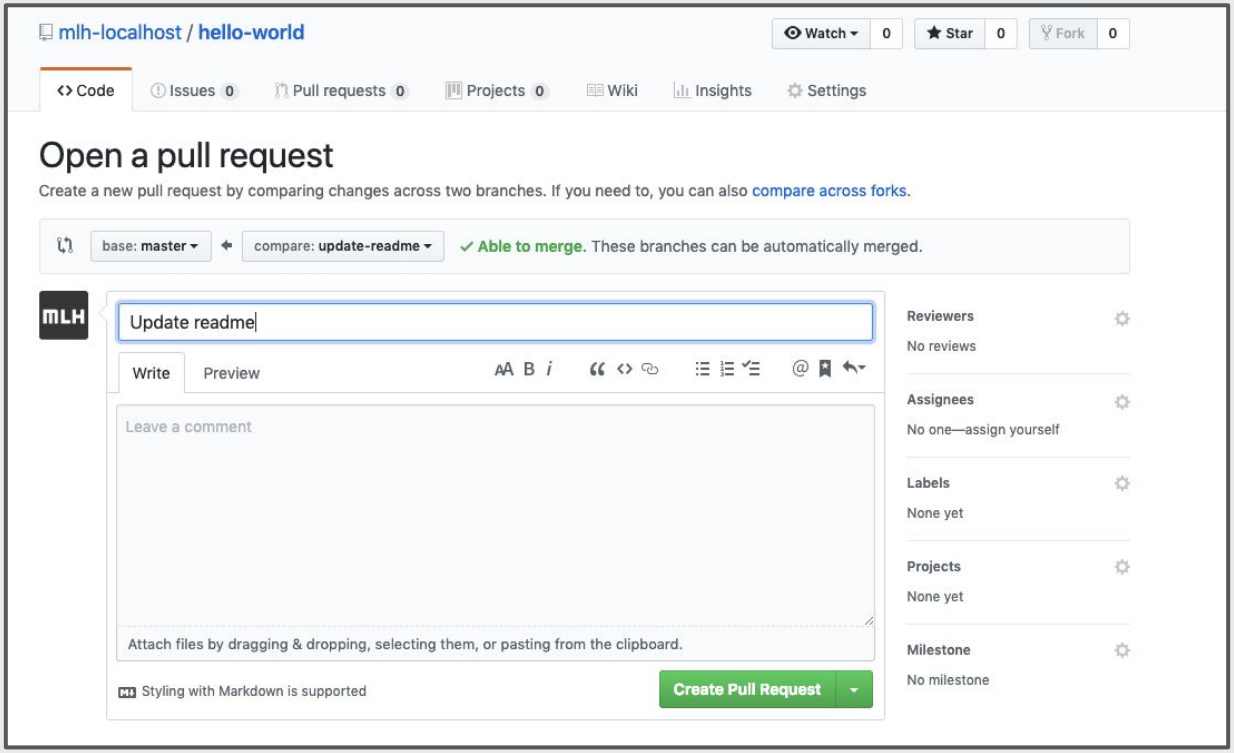
Let's make a change!

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



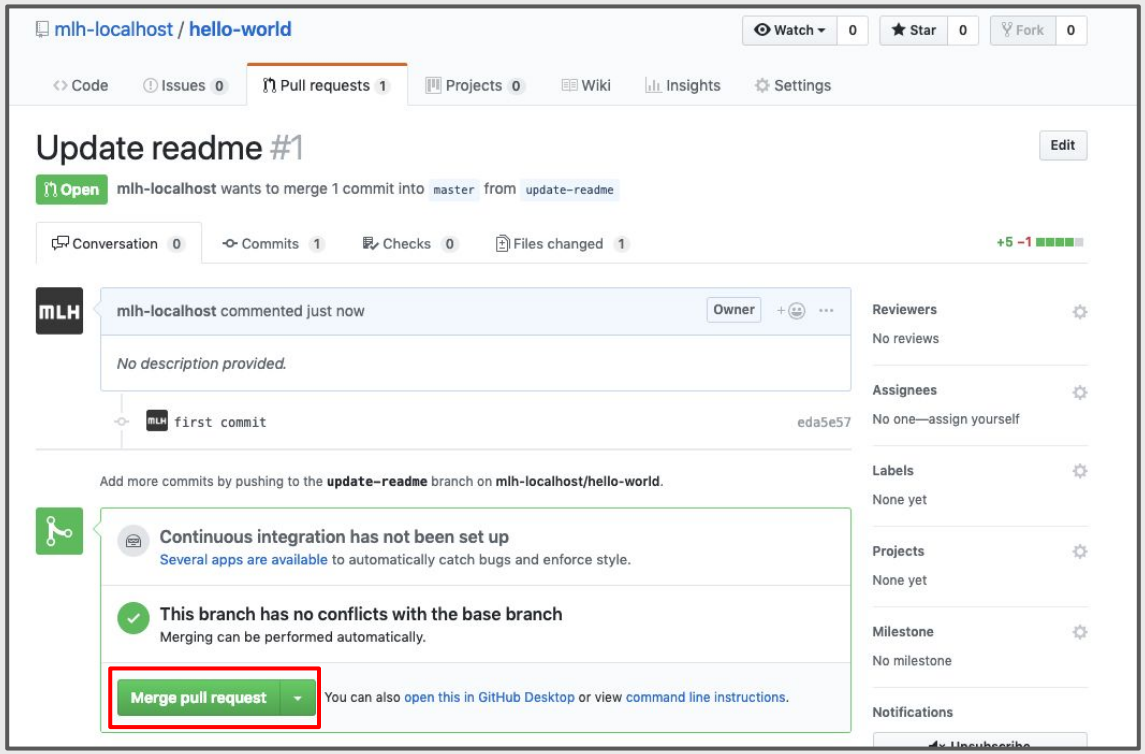
Let's make a change!

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



Let's make a change!

14. 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.

Let's make a change!

16. 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!

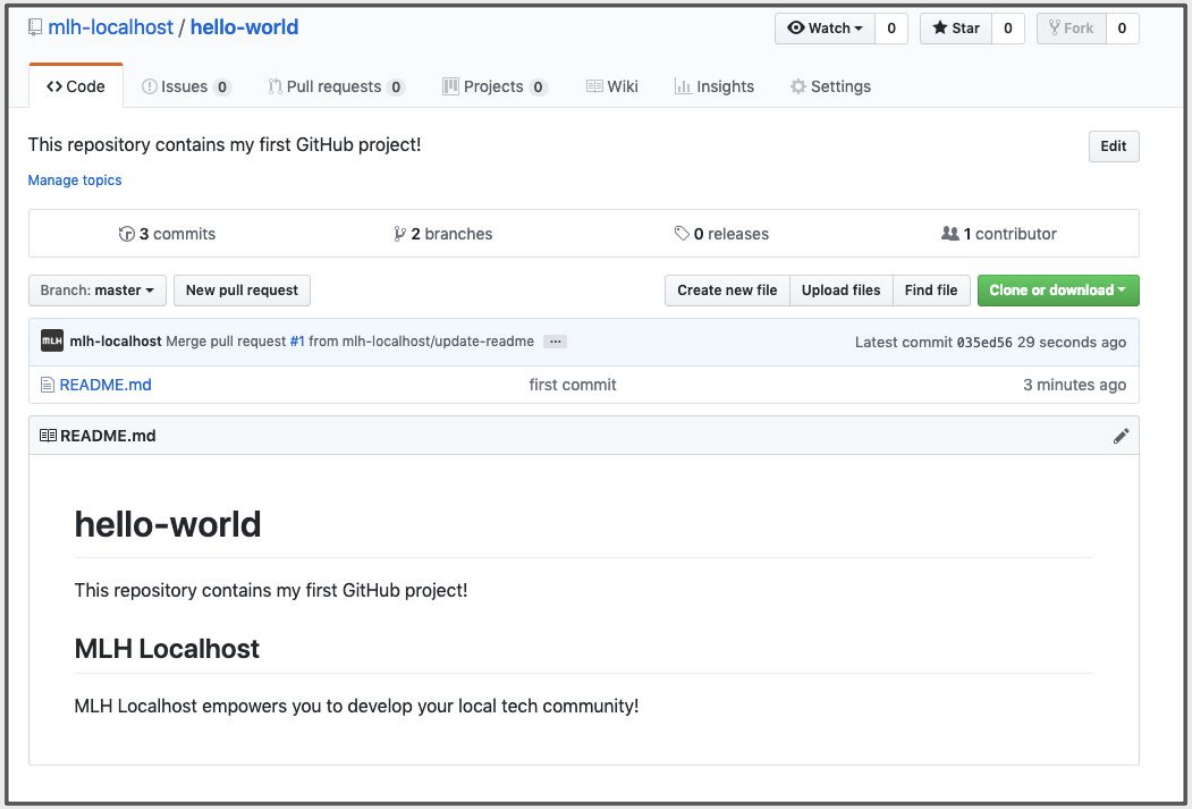


Table of Contents

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

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!


Workshop
How to Collaborate with GitHub
MLH localhost
GitHub
v.0.0.1