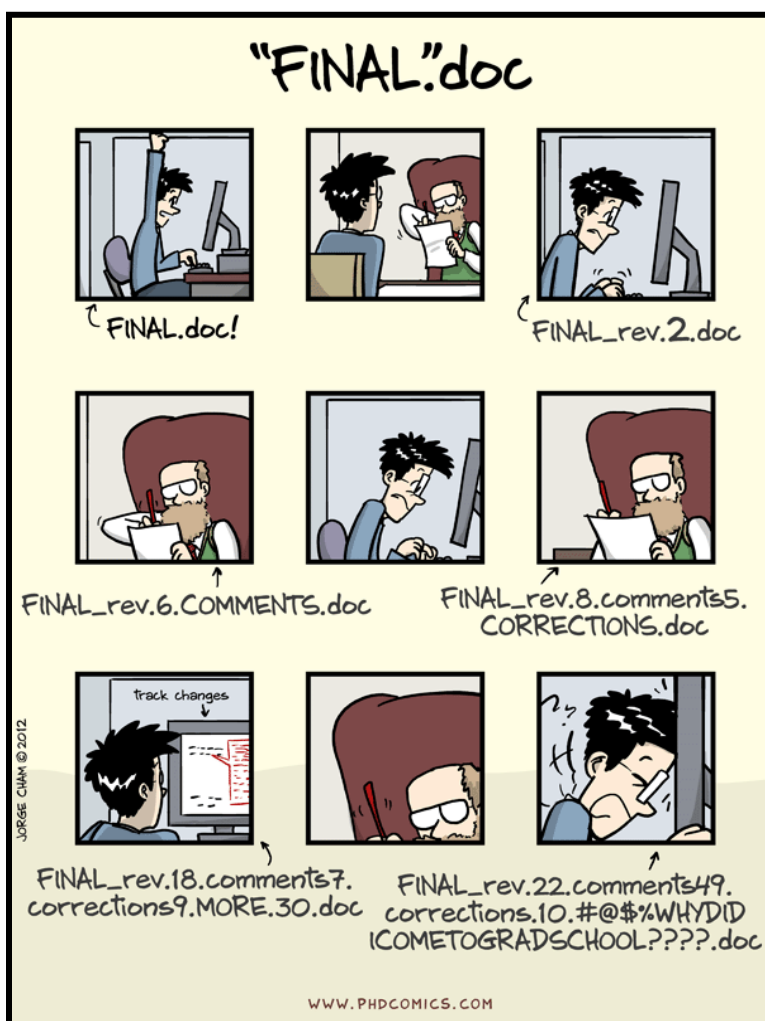


Github Workshop

What is Git?...

We often want to keep track of the different versions in case we want to go back, but this can be painful



Git vs GitHub?...

git and GitHub are not the same things. Git is an open-source, version control tool created in 2005 by developers working on the Linux operating system; GitHub is a company founded in 2008 that makes tools that integrate with git. You do not need GitHub to use git, but you cannot use GitHub without using git. There are many other alternatives to GitHub, such as GitLab, BitBucket, and "host-your-own" solutions such as gogs and gittea.

1. Installing Git

Check your version of Git

You can check your current version of Git by running the `git --version` command in a terminal (Linux, macOS) or command prompt (Windows).

For example:

```
git --version  
git version 2.7.4
```

Download and install Git

[Link](#)

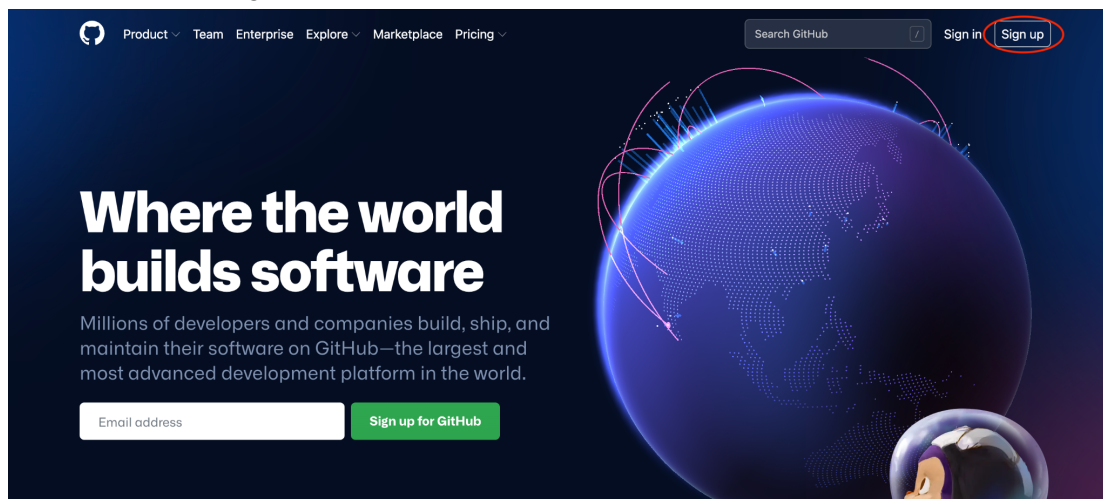
[1.5 Getting Started - Installing Git](#)

2. Creating a GitHub account

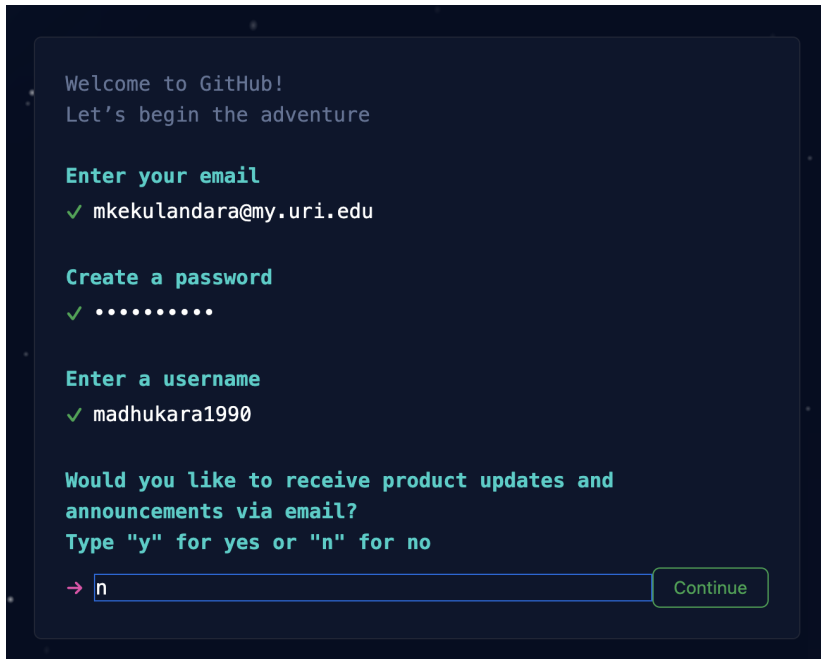
Create an account at GitHub3 (if you haven't already) using your @uri.edu (or any) email or go to <https://github.com/settings/emails> if you want to add your email to an existing account. You must verify your email address with GitHub.

[GitHub sign-up](#)

1. Click the Sign-up button.



2. Fill out the Sign-up Form

A screenshot of the GitHub sign-up form. The background is dark blue. The text is white and green. It says "Welcome to GitHub!" and "Let's begin the adventure". There are three sections: "Enter your email" with a green checkmark and the email "mkekulandara@my.uri.edu"; "Create a password" with a green checkmark and a series of dots; and "Enter a username" with a green checkmark and the username "madhukara1990". Below these is a question: "Would you like to receive product updates and announcements via email?" with the instruction "Type 'y' for yes or 'n' for no". There is a text input field with "n" and a green "Continue" button.

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ mkekulandara@my.uri.edu

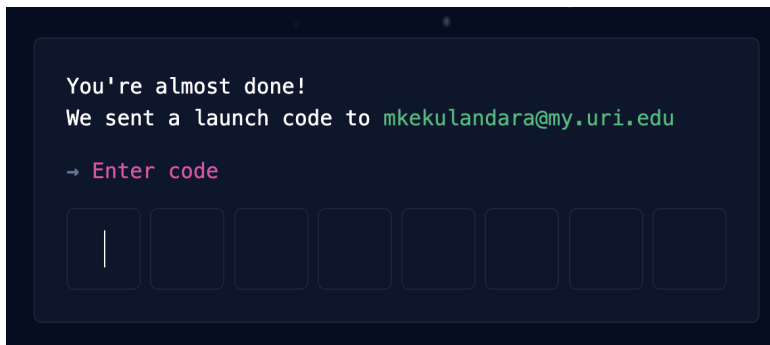
Create a password
✓

Enter a username
✓ madhukara1990

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no

→ n Continue

3. Check your email for code

A screenshot of the GitHub email code entry screen. The background is dark blue. The text is white. It says "You're almost done!" and "We sent a launch code to mkekulandara@my.uri.edu". Below this is a prompt "→ Enter code" and a row of eight input boxes. The first box has a vertical line cursor.

You're almost done!
We sent a launch code to mkekulandara@my.uri.edu

→ Enter code

|

4. Select your account type.
 5. Add the features you want.
 6. Select Paid or free version.
-

3. Create a new repository

Using Git:

```
$ cd /Users/user/my_project # Navigate to the project directory
$ git init # Create the repository
```

Using GitHub:

1. Click the Create repository button on your GitHub dashboard.

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

[Create repository](#)[Import repository](#)

2. Add a name, description, and visibility.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *



madhukara1990 ▾

Repository name *

gitWorkshop ✓

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

Description (optional)

This repository is a tutorial about using github



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

3. Configure the repository and click create.

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

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)
.gitignore template: **None** ▾

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)
License: **None** ▾

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

4. Repository homepage.

The screenshot shows the GitHub repository homepage for `madhukara1990/gitWorkshop`. The repository is public and has 1 branch (main) and 0 tags. The commit history shows an initial commit by madhukara1990. The README file is visible, titled `gitWorkshop`, with the content: "This repository is a tutorial about using github". The right sidebar shows repository statistics: 0 stars, 1 watching, and 0 forks. There are also buttons for Pin, Unwatch, Fork, and Star.

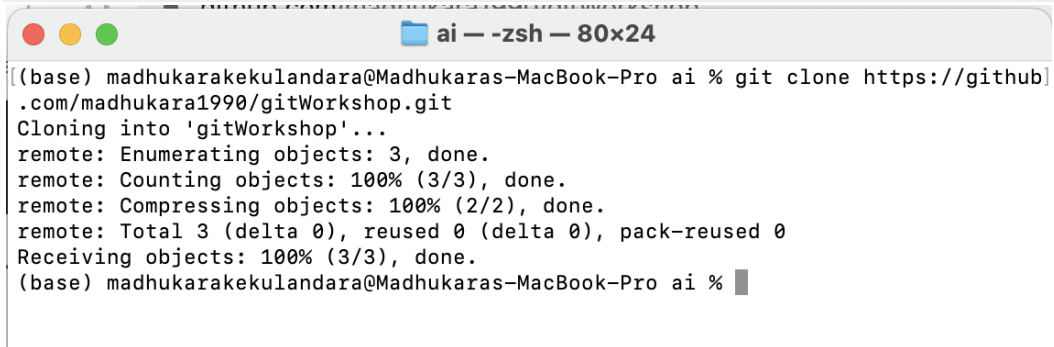
4. Clone the repository

1. Click the code button and copy the repository link.

The screenshot shows the 'Code' dropdown menu from the GitHub repository page. It includes options for cloning the repository using HTTPS, SSH, or GitHub CLI (marked as 'New'). The HTTPS URL is displayed as `https://github.com/madhukara1990/gitWo`. Below the URL, there is a note: "Use Git or checkout with SVN using the web URL." There are also options to "Open with GitHub Desktop" and "Download ZIP".

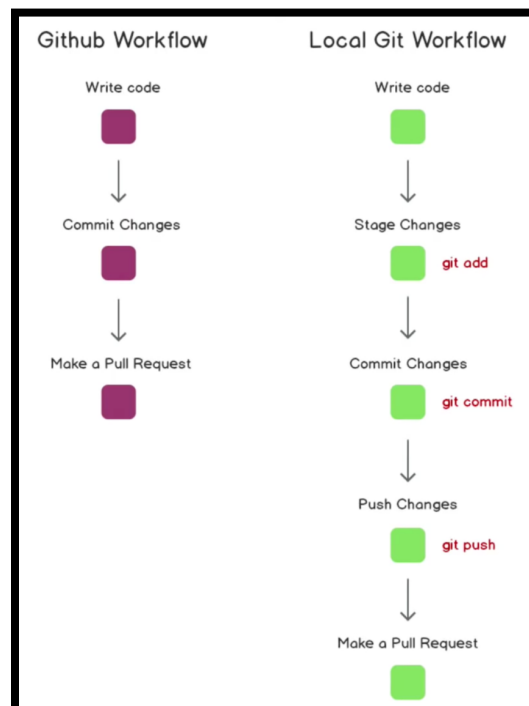
2. Open a terminal window and run the following code.

```
$ git clone 'Paste the link here'
```



```
ai — -zsh — 80x24
(base) madhukarakekulandara@Madhukaras-MacBook-Pro ai % git clone https://github.com/madhukara1990/gitWorkshop.git
Cloning into 'gitWorkshop'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
(base) madhukarakekulandara@Madhukaras-MacBook-Pro ai %
```

5. Adding a new file



1. Create a new file in your repository folder.

```
madhukarakekulandara@Madhukaras-MacBook-Pro ai % touch newfile.txt
madhukarakekulandara@Madhukaras-MacBook-Pro ai % ls
README.md      newfile.txt
```


2. After creating the new file, you can use the git status command to see which files git knows exist.

```
madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git status
On branch main
Your branch is up to date with 'origin/main'.

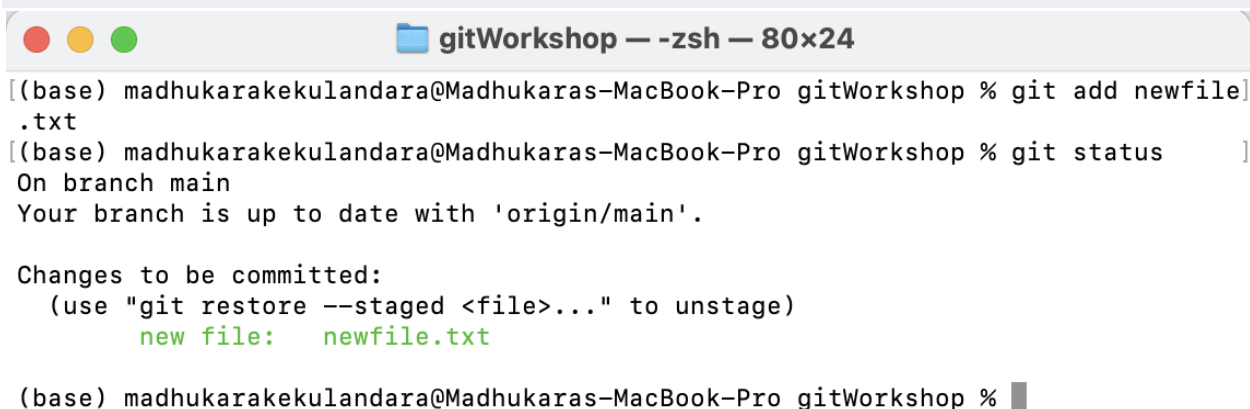
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    newfile.txt

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

What this basically says is, "Hey, we noticed you created a new file called newfile.txt, but unless you use the 'git add' command we aren't going to do anything with it."

3. Add a file to the staging environment using the git add command.

```
git add <filename>
```



```
gitWorkshop — -zsh — 80x24
[(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git add newfile]
.txt
[(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   newfile.txt

(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop %
```

6. Committing Your Changes

Now that your staging area is set up the way you want it, you can commit your changes. Remember that anything that is still unstaged — any files you have created or modified that you haven't run git add on since you edited them — won't go into this commit. They will stay as modified files on your disk.

1. Commit using git commit -m "commit message"

```
madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git commit -m "committing the
newfile.txt"
[main 51b4f5e] committing the newfile.txt
```


2. Check the status

```
madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

3. Use the git log command to check your commit.

```
madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git log
commit 51b4f5e7b610ad1754c2f829c0d9a1b69821cfdc (HEAD -> main)
Author: Madhukara Kekulandara <madhukarakekulandara@Madhukaras-MacBook-Pro.local>
Date:   Sun Jun 19 14:41:06 2022 -0400

    committing the newfile.txt

commit c58bef3319d2be7b24edc3cd7a17d99213a00f54 (origin/main, origin/HEAD)
Author: madhukara1990 <107808230+madhukara1990@users.noreply.github.com>
Date:   Sun Jun 19 13:15:03 2022 -0400

    Initial commit
```

7. Pushing to Your Remotes

When you have your project at a point that you want to share, you have to push it upstream. The command for this is simple: `git push <remote> <branch>`.

```
$ git push origin main
Username for 'https://github.com': <username>
Password for 'https://your-username@github.com':
remote: Support for password authentication was removed on August 13, 2021.
Please use a personal access token instead.
remote: Please see
https://github.blog/2020-12-15-token-authentication-requirements-for-git-op
erations/ for more information.
fatal: Authentication failed for
'https://github.com/<username>/repo-name.git/'
```

When you try to push your code using Username and Password you will get the above error. To fix it you need to generate an access token. Use the following steps to fix it.

- Click on your GitHub profile icon on the top right corner
- Click Settings
- From the menu shown on the left, click Developer Settings
- Click Personal access tokens
- Click Generate new token
- Add a note that will help you identify the scope of the access token to be generated
- Choose the Expiration period from the drop-down menu (Ideally you should avoid choosing the No Expiration option)
- Finally, select the scopes you want to grant the corresponding access to the generated access token. Make sure to select the minimum required scopes otherwise, you will still have trouble performing certain Git Operations.
- Finally click Generate Token

Change the remote URL

```
git remote set-url origin  
https://<githubtoken>@github.com/<username>/<repositoryname>.git
```

Now we have successfully configured token-based authentication for a specific repository on GitHub. To test that everything went to plan, simply try to push the changes you've made locally to the remote host. For example,

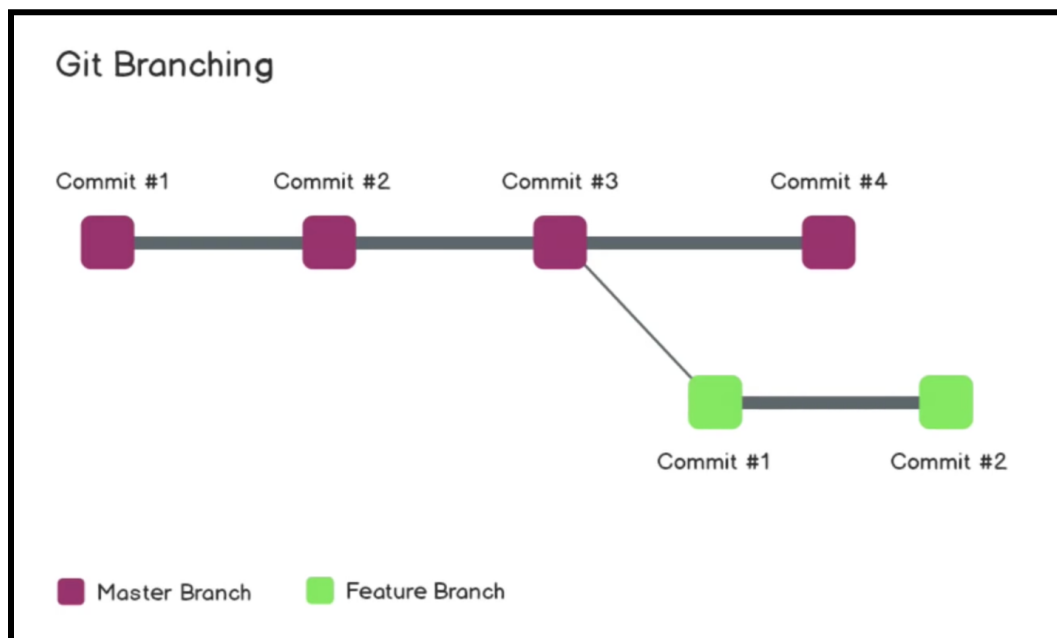
```
$ git push origin main
```

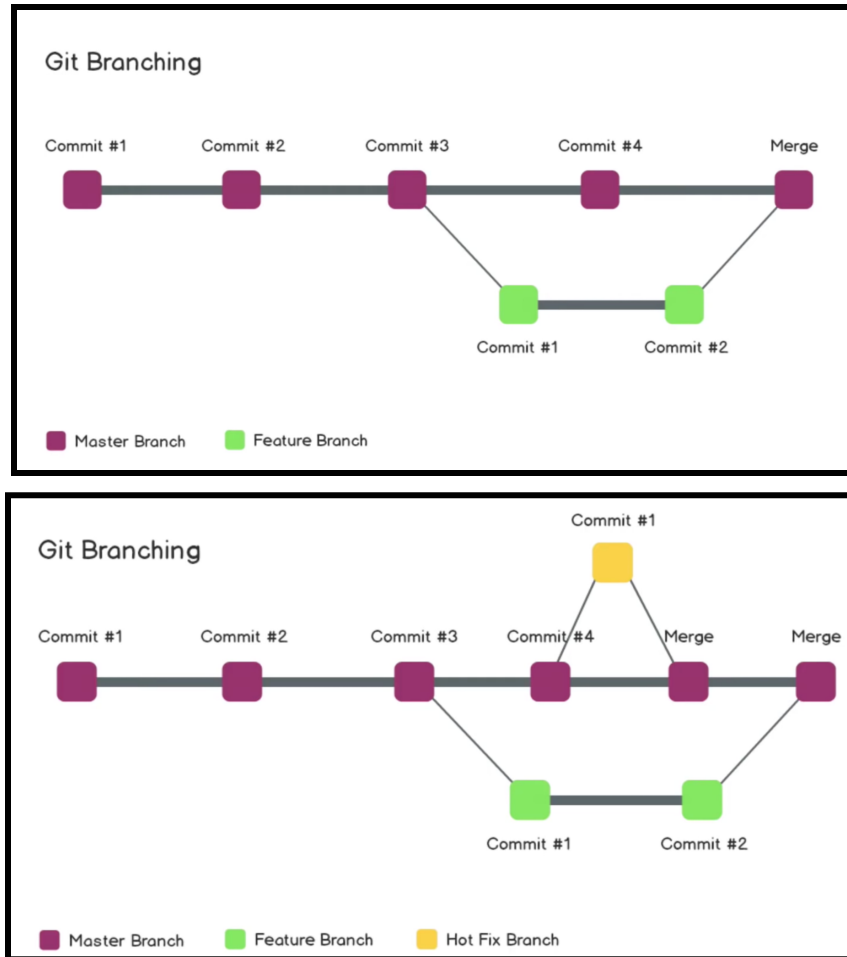
You might be wondering what that "origin" word means in the command above. What happens is that when you clone a remote repository to your local machine, git creates an alias for you. In nearly all cases this alias is called "origin." It's essentially shorthand for the remote repository's URL.

8. Create a new branch

Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where git branches come in.

Branches allow you to move back and forth between 'states' of a project. Official git docs describe branches this way: 'A branch in Git is simply a lightweight movable pointer to one of these commits.' For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can merge your changes from your branch into the primary branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files.





Let's say you are on the primary branch and want to create a new branch to develop your web page. Here's what you'll do: Run `git checkout -b <my branch name>`. This command will automatically create a new branch and then 'check you out on it, meaning git will move you to that branch, off of the primary branch.

After running the above command, you can use the `git branch` command to confirm that your branch was created:

```
gitWorkshop — -zsh — 92x24
[(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git checkout -b newbranch ]
Switched to a new branch 'newbranch'
[(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git branch ]
main
* newbranch
(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop %
```

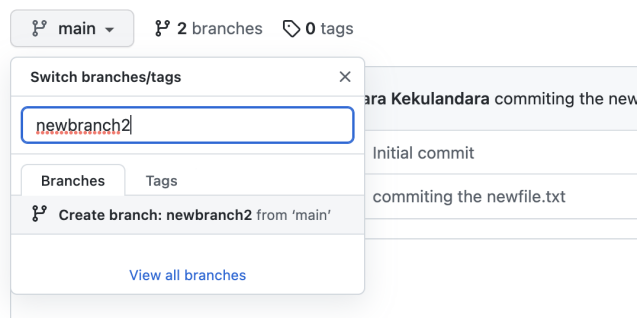

Push a branch to GitHub

Now we'll push the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the primary branch.

To push changes onto a new branch on GitHub, you'll want to run `git push origin yourbranchname`. GitHub will automatically create the branch for you on the remote repository:

```
git push origin newbranch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'newbranch' on GitHub by visiting:
remote:
https://github.com/madhukara1990/gitWorkshop/pull/new/newbranch
remote:
To https://github.com/madhukara1990/gitWorkshop.git
 * [new branch]      newbranch -> newbranch
```

Create a branch from GitHub



Fetching the new branch to local repository

```
$ git branch
* main
  newbranch
$ git fetch --prune
From https://github.com/madhukara1990/gitWorkshop
 * [new branch]      newbranch2 -> origin/newbranch2
$ git branch
* main
  newbranch
```



```
$ git checkout newbranch2
Branch 'newbranch2' set up to track remote branch 'newbranch2' from
'origin'.
Switched to a new branch 'newbranch2'
$ git branch
  main
  newbranch
* newbranch2
```

9. Create a pull request (PR)

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the primary branch.

Let's create a new branch and add a new file to the new branch first. And then learn how to create a pull request.

```
$ git checkout -b prbranch
Switched to a new branch 'prbranch'
(base) madhukarakekulandara@Madhukaras-MacBook-Pro gitWorkshop % git branch
  main
  newbranch
  newbranch2
* prbranch
$ touch mychanges.txt
$ git status
On branch prbranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    mychanges.txt

nothing added to commit but untracked files present (use "git add" to
track)
$ git add mychanges.txt
$ git status
On branch prbranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
```



```
new file:   mychanges.txt
```

```
$ git commit -m "commit mychanges file"
[prbranch 3850167] commit mychanges file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mychanges.txt
$ git log
commit 38501675a8d1fa712dff827ec0496aecf19f73a (HEAD -> prbranch)
Author: madhukara <mkekulandara@my.uri.edu>
Date:   Mon Jun 20 16:50:53 2022 -0400
```

```
commit mychanges file
```

```
commit 51b4f5e7b610ad1754c2f829c0d9a1b69821cfdc (origin/newbranch2,
origin/newbranch, origin/main, origin/HEAD, newbranch2, newbranch, main)
Author: Madhukara Kekulandara
<madhukarakekulandara@Madhukaras-MacBook-Pro.local>
Date:   Sun Jun 19 14:41:06 2022 -0400
```

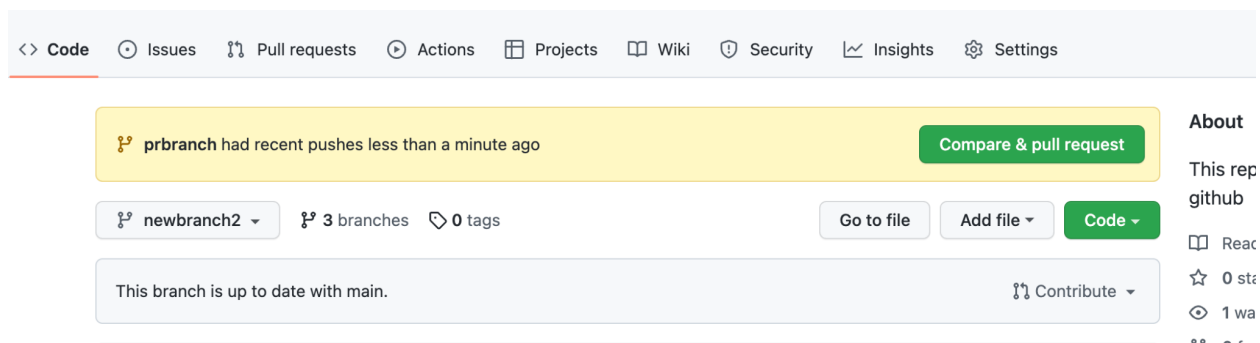
```
committing the newfile.txt
```

```
commit c58bef3319d2be7b24edc3cd7a17d99213a00f54
Author: madhukara1990 <107808230+madhukara1990@users.noreply.github.com>
Date:   Sun Jun 19 13:15:03 2022 -0400
```

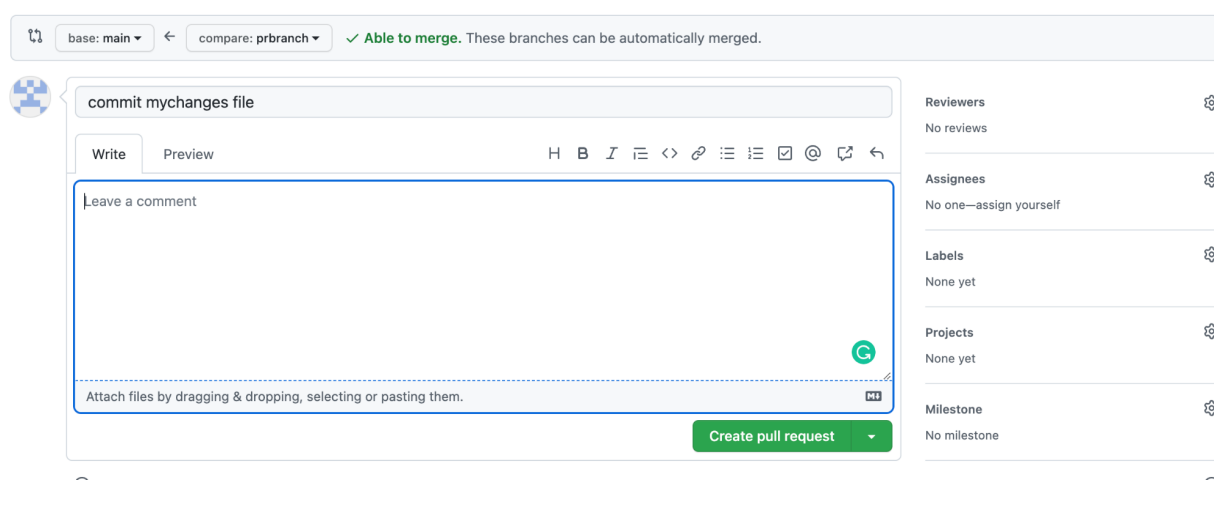
```
Initial commit
```

```
$ git push origin prbranch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 289 bytes | 289.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'prbranch' on GitHub by visiting:
remote:   https://github.com/madhukara1990/gitWorkshop/pull/new/prbranch
remote:
To https://github.com/madhukara1990/gitWorkshop.git
* [new branch]      prbranch -> prbranch
```


GitHub will alert the addition of the newbranch as below.

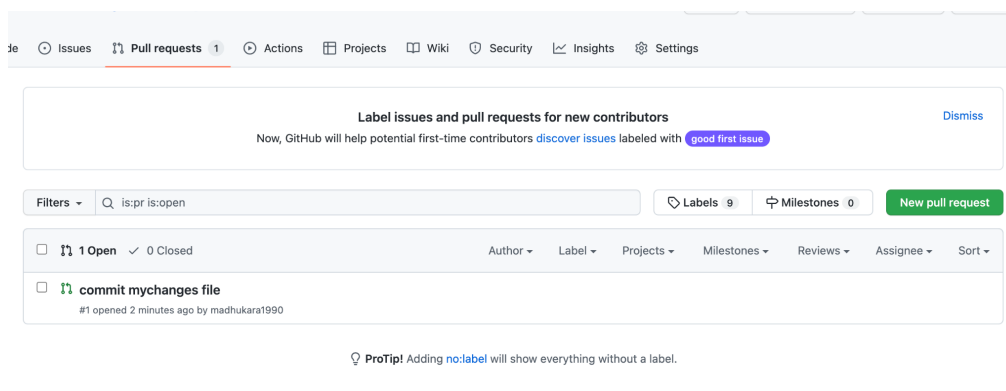


Click Compare & pull request to create a new PR

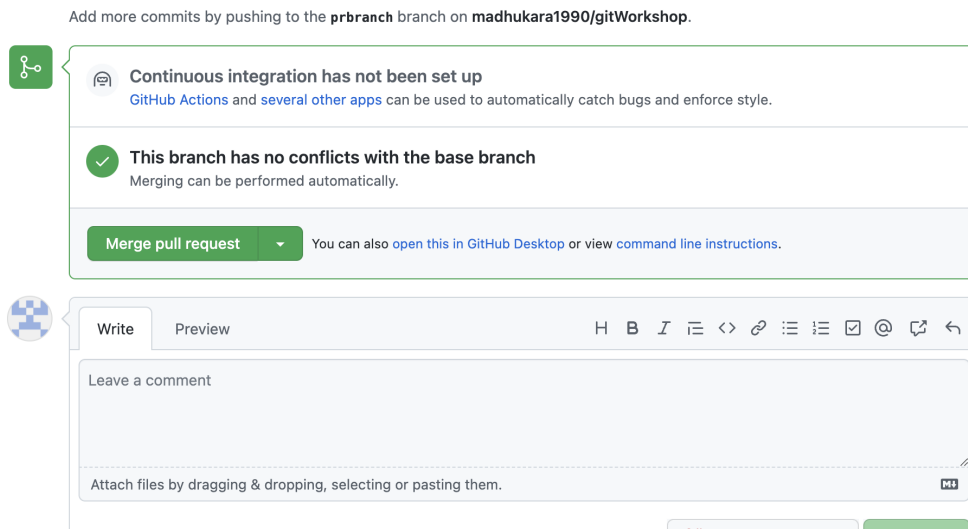


10. Merge a PR

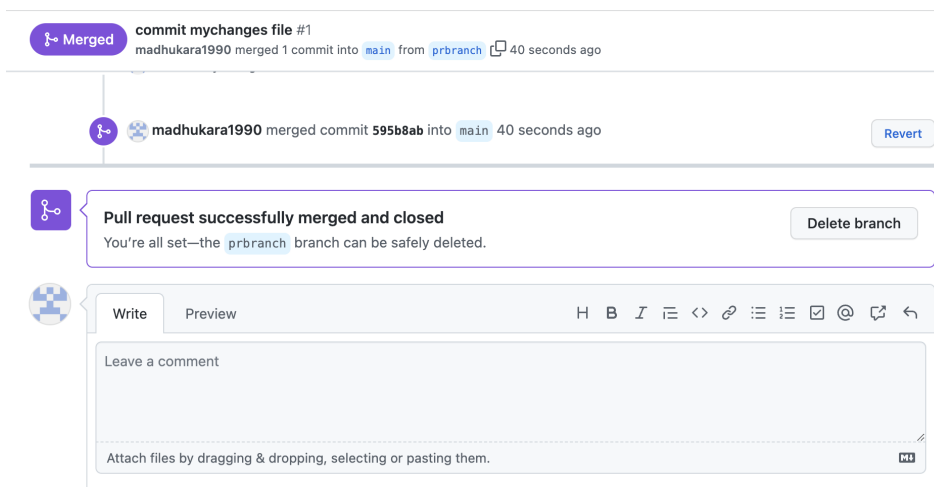
To merge a pull request go to the Pull request tab in your gitHub repository page and open the pull request you want to merge.



Click on the Merge pull request button to merge your pull request.



Finally, click on Confirm Merge button to merge.



You can delete your branch by clicking on the Delete branch button.

11: Get changes on GitHub back to your computer

Right now, the repo on GitHub looks a little different than what you have on your local machine. For example, the commit you made in your branch and merged into the primary branch doesn't exist in the primary branch on your local machine.

In order to get the most recent changes that you or others have merged on GitHub, use the `git pull origin master` command (when working on the primary branch). In most cases, this can be shortened to “`git pull`”.

And use the `git log` command to check the updates.

12. Undo changes in Git

To unstage a staged file you can use the `git reset` command

```
$git reset <filename>
```

To unstage your changes into a specific commit in your repository use the `git reset` as follows

```
$git reset <commit hash>
```

To undo (remove) all the changes to a specific commit in your repository use the `--hard` along with `git reset`

```
$git reset --hard <commit hash>
```

Useful Links:

- <https://training.github.com/>
Github's official git cheat sheets! Handy for remembering the everyday commands you'll use.
- <https://learngitbranching.js.org/>
Confused or intrigued by git's branch system? That just means you're human! It's one of the deepest parts of git, but also arguably the most powerful. Understanding the branch model gives you git superpowers, and this tutorial gives you a way to learn git branches in a visual, intuitive way.
- <https://git-school.github.io/visualizing-git>
Another tool for exploring git visually. This one is more of an open-ended sandbox than learngitbranching.js.org
- <https://github.com/jlord/git-it-electron>
A desktop application that helps you learn git through challenges you have to solve. It has a series of levels, each requiring you to use git commands to arrive at a correct answer.
- <https://github.com/Gazler/githug>
If you liked git-it, Githug is another puzzle-based tutorial designed to give you a practical way of learning git.