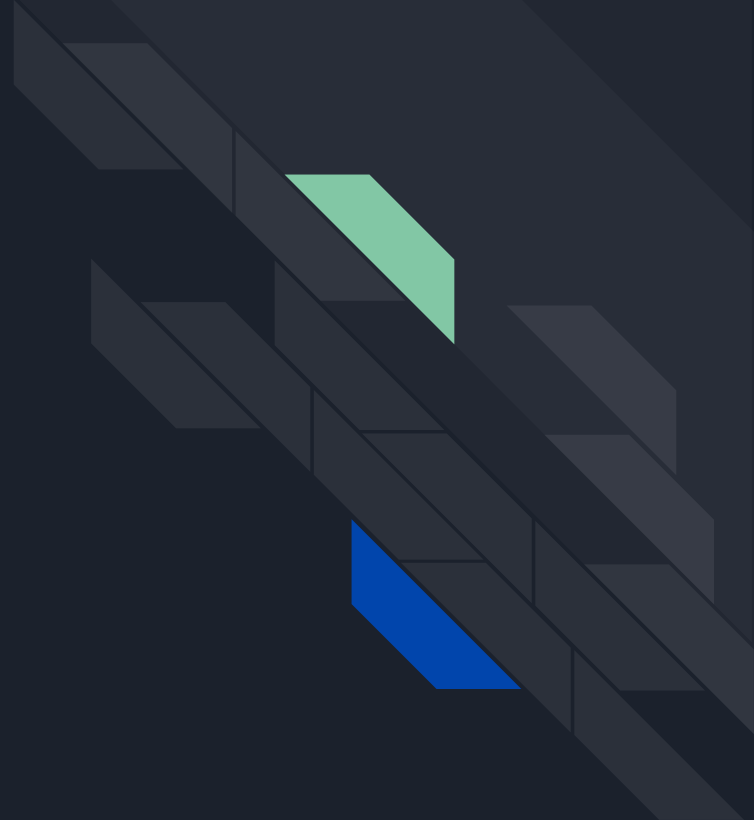




GitHub Basics



What is GitHub?

- GitHub is a commercial site that allows users to **host Git repositories** publicly and privately
- **Open source projects** host or mirror their repositories on GitHub
- Post your own code for others to use or contribute to
- Use and **learn** from the code in other people's repositories



Git vs GitHub

Linus Torvalds would kill you for this!

Git is a version control system; think of it as a series of snapshots (commits) of your code. You see a path of these snapshots, in which order they were created. You can make branches to experiment and come back to snapshots you took.

GitHub, is a web-page on which you can publish your Git repositories and collaborate with other people.




Step 1. Create a Repository

- 01 Go to your Profile. Click on Repositories and then, select New Repository
- 02 Give a name to your repo, add description and select its visibility (Public or Private)
- 03 Initialize with a README.md --not compulsory, but desired :)

Create a new repository

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

Owner

 ishitamed19 ▾

Repository name

hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **animated-happiness**.

Description (optional)

This is a sample repository!



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

NOTE: If you create a file in your repository named **.gitignore**, Git uses it to determine which files and directories to **ignore**, before you make a commit. A **.gitignore** file should be committed into your repository, in order to share the **ignore** rules with any other users that clone the repository.



Step 2. Make and Commit Changes

On GitHub, saved changes are called *commits*. Each commit has an associated *commit message*, which is a description explaining why a particular change was made.

1. Click the README.md file.
2. Click the pencil icon in the upper right corner of the file view to edit.
3. In the editor, make the necessary changes.
4. Write a commit message that describes your changes.
5. Click **Commit changes** button.

37 # Editing README.md
38 Demo to explain commit changes





Commit changes

Finish README

A sample description

ishitamediratta19@gmail.com  

-  Commit directly to the `master` branch.
-  Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

branching

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.





Branching

Branching is the way to work on different versions of a repository at one time.

By default your repository has one branch named `master` which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.

When you create a branch off the `master` branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, **you could pull in those updates.**



To create a new branch

1. Go to your new repository `hello-world`.
2. Click the drop down at the top of the file list that says **branch: master**.
3. Type a branch name, `readme-edits`, into the new branch text box.
4. Select the blue **Create branch** box or hit “Enter” on your keyboard.



Push & Pull

Now you have two branches, master and readme-edits. They look exactly the same, but not for long! Next we'll add our changes to the new branch.

Edit your README file again and this time select the "readme-edits" branch while committing the changes.

These changes will be made to just the README file on your readme-edits branch, so now this branch contains content that's different from master

- Commit directly to the `readme-edits` branch
- Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

ONE DOES NOT SIMPLY

**MERGE INTO MASTER WITHOUT
PULL REQUEST AND CODE REVIEW**



Step 4. Open Pull Request

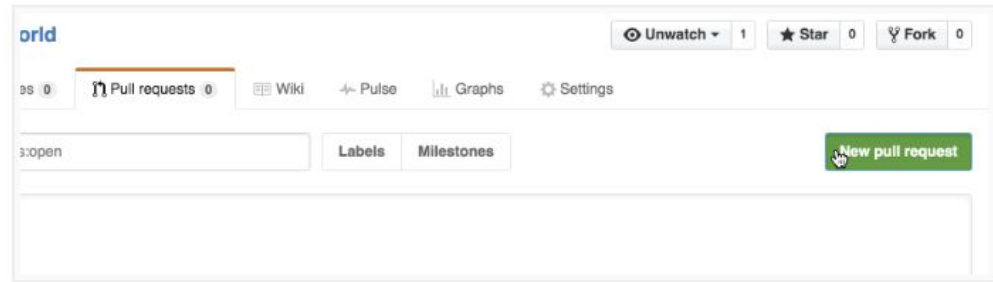
Pull Requests are the heart of collaboration on GitHub.

When you open a *pull request*, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch.

Pull requests show *diffs*, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

You can even open pull requests in your own repository and merge them yourself

Click the  **Pull Request** tab, then from the Pull Request page, click the green **New pull request** button.



In the **Example Comparisons** box, select the branch you made, `readme-edits`, to compare with `master` (the original).



Look over your changes in the diffs on the Compare page, make sure they're what you want to submit.

1 commit 1 file changed

Commits on Oct 27, 2014

hubot Finish README ...

Showing 1 changed file with 1 addition and 1 deletion.

2 README.md

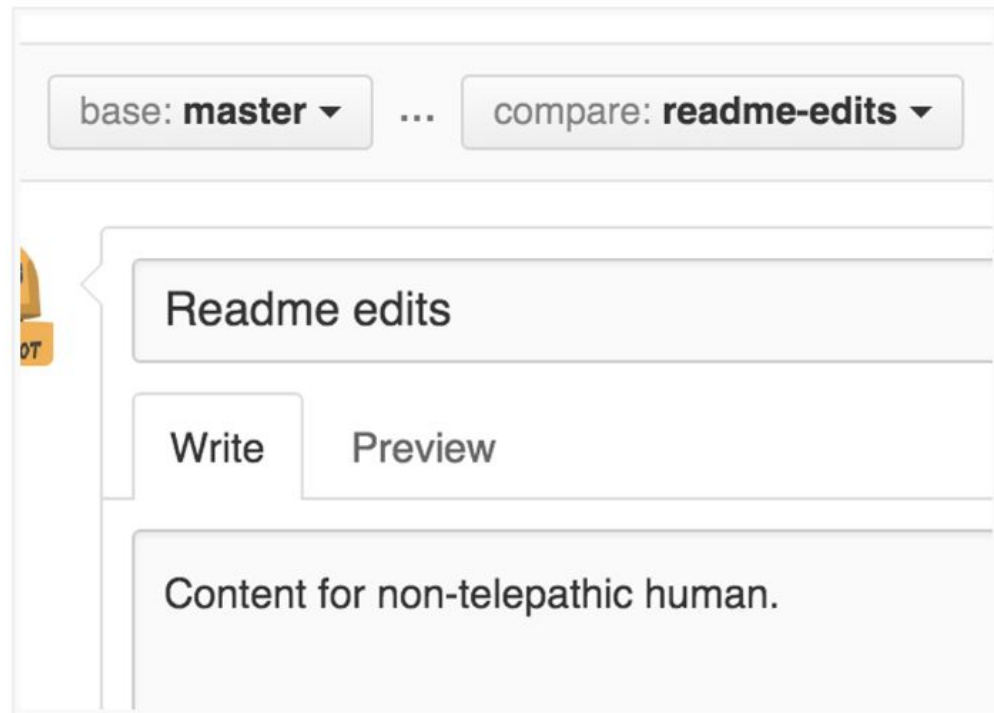
...	...	@@ -1,4 +1,4 @@
1	1	hello-world
2	2	=====
3	3	
4		-Just another repository
	4	+Hubot here, I like Node.js and Coffee them far superior to Earth tacos.

When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.

base: master ... compare: readme-ed

Create pull request Discuss and review the

Give your pull request a title and write a brief description of your changes.



The image shows a screenshot of a pull request editor interface. At the top, there are two dropdown menus: "base: master" and "compare: readme-edits". Below these, there is a yellow icon with the number "07". The main content area has a title field containing "Readme edits" and two tabs, "Write" and "Preview". The "Write" tab is active, and the description field contains the text "Content for non-telepathic human."



Step 5. Merging Pull Request

In this final step, it's time to bring your changes together – merging your readme-edits branch into the master branch.

1. Click the green **Merge pull request** button to merge the changes into master.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.



This branch has no conflicts with the base branch

Merging can be performed automatically.



Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



Pull request successfully merged and closed

You're all set—the `readme-edits` branch can be safely deleted.



Delete branch



**YOU DON'T NEED TO SUBMIT A
PULL REQUEST**

IF YOU PUSH TO MASTER



What's Next?

There are MILLIONS of public repositories on GitHub

If you want to use or contribute to a repository, you can **fork** it.

Fork, in the GitHub context, doesn't extend Git. It only allows clone on the server side.



- After you fork and clone a repository all pushed changes will go to your fork
- These changes will not affect the original repository
- if you would like to get your changes to be incorporated into the original repo, you can submit a pull request



Fork vs Clone

Yes Fork is a clone. It emerged because, **you cannot push to others' copies without their permission**. What they do is make a **copy** of it for you (*fork*), where you will have write permission as well.

In the future if the actual owner or others users with a fork like your changes they can pull it back to their own repo. Alternatively you can send them a "pull-request".



More Terms (TAKE'EM LITE!)

- *Issues*

Use **issues** to track ideas, enhancements, tasks, or bugs for work on **GitHub**. **Issues** can act as more than just a place to report software bugs. As a project maintainer, you can use **Issues** to organize tasks you'd like to accomplish, such as adding new features or auditing old ones.



- ***Merge Conflicts***

Sometimes there are competing changes that Git can't resolve without your help. Often, merge conflicts happen when people make different changes to the same line of the same file, or when one person edits a file and another person deletes the same file.

You must resolve all merge conflicts before you can merge a pull request on GitHub. If you have a merge conflict between the compare branch and base branch in your pull request, you can view a list of the files with conflicting changes above the Merge pull request button. The Merge pull request button is deactivated until you've resolved all conflicts between the compare branch and base branch.