



Hello World

🕒 10 minute read

The **Hello World** project is a time-honored tradition in computer programming. It is a simple exercise that gets you started when learning something new. Let's get started with GitHub!

You'll learn how to:

- Create and use a repository
- Start and manage a new branch
- Make changes to a file and push them to GitHub as commits
- Open and merge a pull request

What is GitHub?

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

This tutorial teaches you GitHub essentials like *repositories*, *branches*, *commits*, and *Pull Requests*. You'll create your own Hello World repository and learn GitHub's Pull

Request workflow, a popular way to create and review code.

No coding necessary

To complete this tutorial, you need a [GitHub.com account](#) and Internet access. You don't need to know how to code, use the command line, or install Git (the version control software GitHub is built on).

Tip: Open this guide in a separate browser window (or tab) so you can see it while you complete the steps in the tutorial.

Step 1. Create a Repository


A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. We recommend including a *README*, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository. *It also offers other common options such as a license file.*

Your `hello-world` repository can be a place where you store ideas, resources, or even share and discuss things with others.


To create a new repository

1. In the upper right corner, next to your avatar or identicon, click **+** and then select **New repository**.
2. Name your repository `hello-world`.
3. Write a short description.
4. Select **Initialize this repository with a README**.

PUBLIC



Owner

 hubot ▾

/


Repository name

hello-world ✓


Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional)

Just another 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 allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** ▾

|

Add a license: **None** ▾ 

Create repository

Click **Create repository**. 🎉

Step 2. Create a Branch

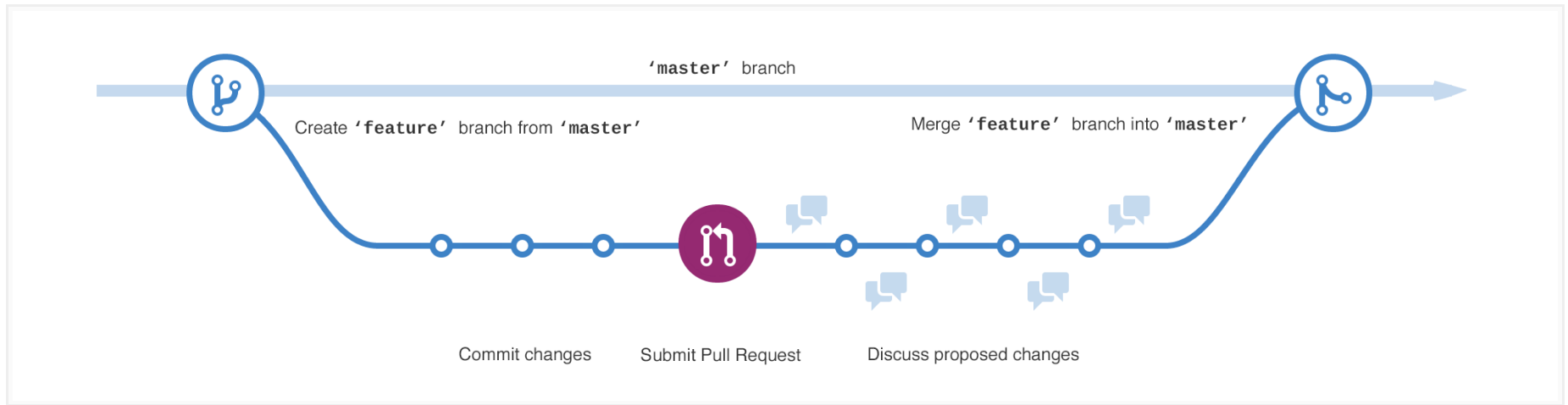
Branching is the way to work on different versions of a repository at one time.

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.

This diagram shows:

- The `master` branch
- A new branch called `feature` (because we're doing 'feature work' on this branch)
- The journey that `feature` takes before it's merged into `master`



Have you ever saved different versions of a file? Something like:

- `story.txt`
- `story-joe-edit.txt`
- `story-joe-edit-reviewed.txt`

Branches accomplish similar goals in GitHub repositories.

Here at GitHub, our developers, writers, and designers use branches for keeping bug fixes and feature work separate from our `master` (production) branch. When a change is ready, they merge their branch into `master`.

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.

Just another repository — Edit

🕒 1 commit

🌿 1 branch



🌿 branch: **master** ▾

hello-world /

Initial commit



hubot authored just now



README.md

Initial



README.md


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.

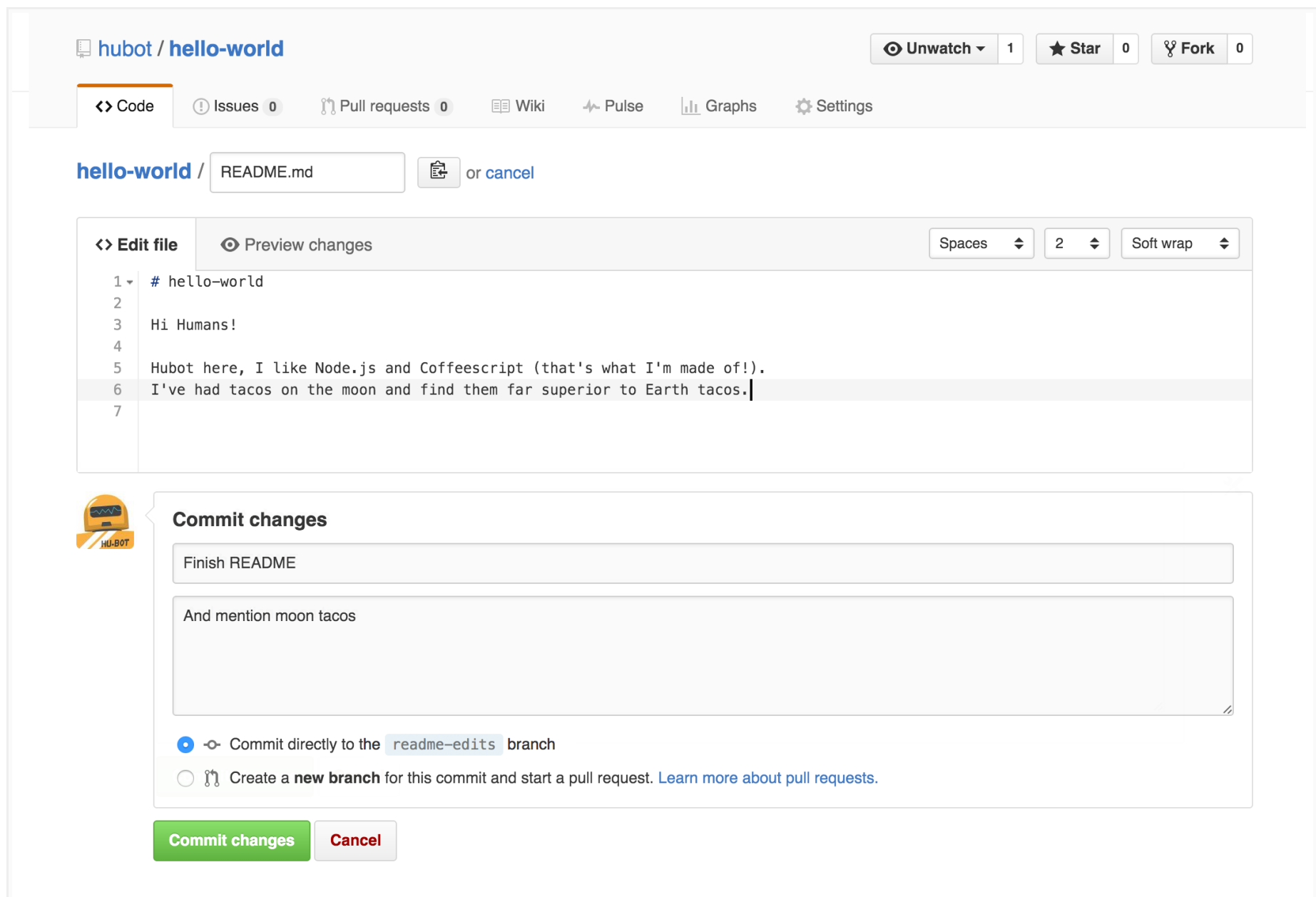
Step 3. Make and commit changes

Bravo! Now, you're on the code view for your `readme-edits` branch, which is a copy of `master`. Let's make some edits.

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. Commit messages capture the history of your changes, so other contributors can understand what you've done and why.

Make and commit changes

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, write a bit about yourself.
4. Write a commit message that describes your changes.
5. Click **Commit changes** button.



The screenshot shows the GitHub web interface for the repository `hubot / hello-world`. At the top, there are buttons for `Unwatch` (1), `Star` (0), and `Fork` (0). Below this is a navigation bar with links for `Code`, `Issues` (0), `Pull requests` (0), `Wiki`, `Pulse`, `Graphs`, and `Settings`.

The main content area shows the `hello-world / README.md` file. The file content is as follows:

```
1 # hello-world
2
3 Hi Humans!
4
5 Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
6 I've had tacos on the moon and find them far superior to Earth tacos.
7
```

Below the file editor is the **Commit changes** dialog. It features a text input for the commit message with the placeholder "Finish README" and a larger text area for a description with the placeholder "And mention moon tacos". There are two radio button options for the commit destination:

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

At the bottom of the dialog are two buttons: **Commit changes** (green) and **Cancel** (grey).

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`.

Step 4. Open a Pull Request

Nice edits! Now that you have changes in a branch off of `master`, you can open a *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.


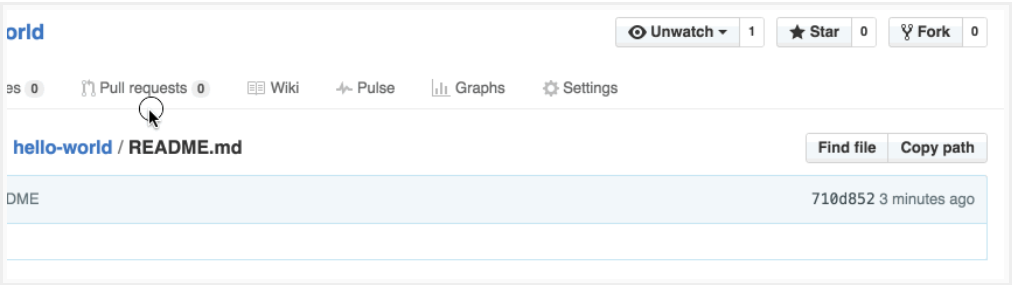

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

By using GitHub's [@mention system](#) in your pull request message, you can ask for feedback from specific people or teams, whether they're down the hall or 10 time zones away.

You can even open pull requests in your own repository and merge them yourself. It's a great way to learn the GitHub Flow before working on larger projects.

Open a Pull Request for changes to the README

Click on the image for a larger version

Step	Screenshot
Click the  Pull Request tab, then from the Pull Request page, click the green New pull request button.	
Select the branch you made, <code>readme-edits</code> , to compare with <code>master</code> (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.

base: master

compare: readme-edits

Readme edits

Write

Preview

Content for non-telepathic human.

When you're done with your message, click **Create pull request**!

Tip: You can use [emoji](#) and [drag and drop images and gifs](#) onto comments and Pull Requests.

Step 5. Merge your 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

Celebrate!

By completing this tutorial, you've learned to create a project and make a pull request on GitHub! 🎉 🐙 ⚡

Here's what you accomplished in this tutorial:

- Created an open source repository
- Started and managed a new branch
- Changed a file and committed those changes to GitHub
- Opened and merged a Pull Request

Take a look at your GitHub profile and you'll see your new [contribution squares](#)!

If you want to learn more about the power of Pull Requests, we recommend reading the [GitHub Flow Guide](#). You might also visit [GitHub Explore](#) and get involved in an Open Source project 🐙

Tip: Check out our other [Guides](#) and [YouTube Channel](#) for more GitHub how-tos.

Last updated April 7, 2016



[GitHub](#) is the best way to build and ship software.

Powerful collaboration, code review, and code management for open source and private projects.