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!

GitHub is a platform for hosting and collaborating on projects. You don’t have to worry about losing data on your hard drive or managing a project across multiple computers — sync from anywhere. Most importantly, GitHub is a collaborative and asynchronous workflow for building software better, together.

This guide walks you through the GitHub Essentials: **Repositories**, **Branches**, **Commits**, **Issues** and **Pull Requests**.

With these skills, you’ll go far. And with this hello-world repository you’ll have a place to store ideas (aka future repositories, like [jlord/hello-world](https://github.com/jlord/hello-world)), resources or other general notes, even discuss things with others (like [holman/feedback](https://github.com/holman/feedback)).

**Install & Code Free Zone**

We’re going to complete the tasks in this tutorial using GitHub.com, so you don’t have to know the command line or even install Git (which is the change-tracking software GitHub is built on). You don’t even need to know how to code. You will, however, need a GitHub account, so sign up if you don’t have one!

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

**Create a Repository**

A **repository** is the basic unit of GitHub, most commonly a single project. Repositories can contain folders and files, including images – anything your project needs. Because we recommend including a README, or a file describing the project, in every repository, 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, but we can skip that for now.*

**To create a new repository**

1. Click the icon next to your username, top-right.
2. Name your repository hello-world.
3. Write a short description.
4. Select **Initialize this repository with a README**.



Click **Create repository**. Boom, repository! :boom:

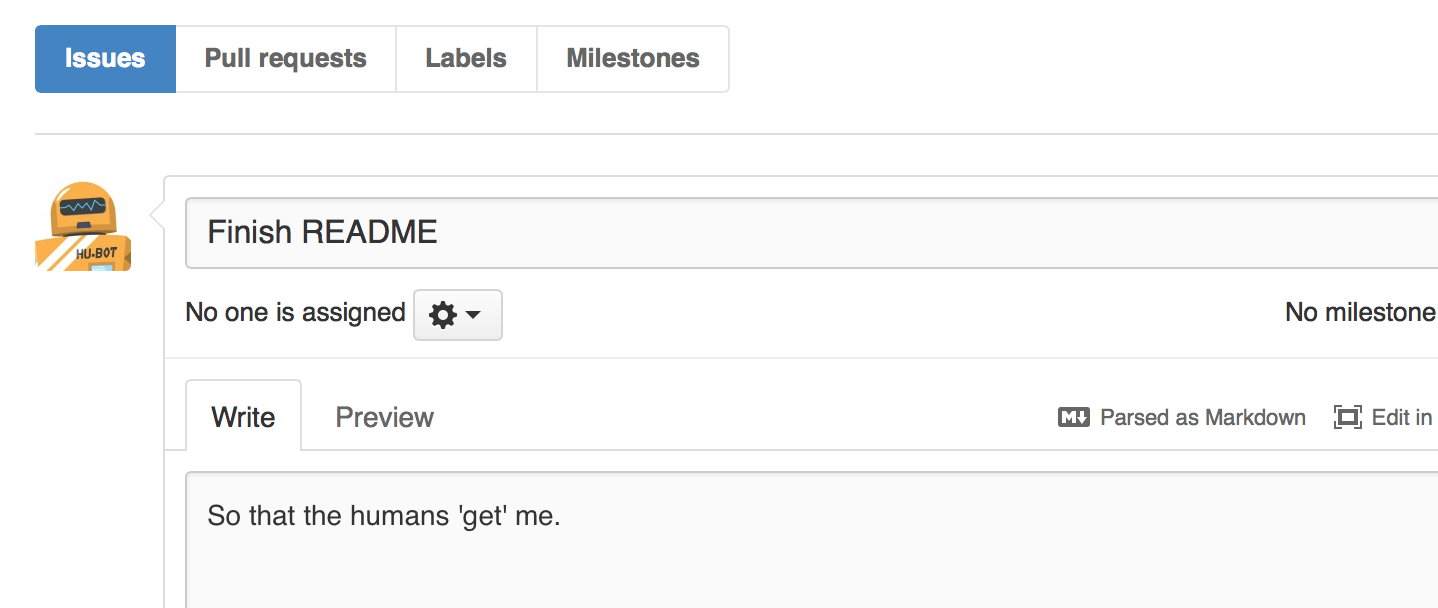
**Open an Issue**

An **Issue** is a note on a repository about something that needs attention. It could be a bug, a feature request, a question or lots of other things. On GitHub you can label, search and assign Issues, making managing an active project easier.

You’ve got a repository now, but it’s pretty bare. It could use a README with more information in it so that people know what is going on. Open an Issue!

**Open an Issue for README edits**

1. Click the **Issues** tab from the sidebar.
2. Click **New Issue**.
3. Give your Issue a title and description.



Click **Submit new Issue** when you’re done. Now this issue has a permanent home (URL) that you can reference even after it is closed.

Next, work towards editing your README and closing this issue.

**Create a Branch**

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

When you create a repository, by default it has one branch with the name master. You could keep working on this branch and have only one, that’s fine. But if you have another feature or idea you want to work on, you can create another branch, starting from master, so that you can leave master in its working state.

When you create a branch, you’re making a copy of the original branch as it was at that point in time (like a photo snapshot). If the original branch changes while you’re working on your new branch, no worries, you can always pull in those updates.

[](https://guides.github.com/activities/hello-world/branching.png)

You may have, at some point in your life, saved different versions of a file like, “story.txt”, “story-joe-edit.txt”, “story-sue-edit.txt”. Branches accomplish the same goals but are easier to manage on GitHub repositories.

At GitHub, our developers, writers, and designers use branches for keeping bug fixes and feature work separate from our master (production) branch. When a feature or fix is ready, the branch is merged 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.



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.

**Make a commit**

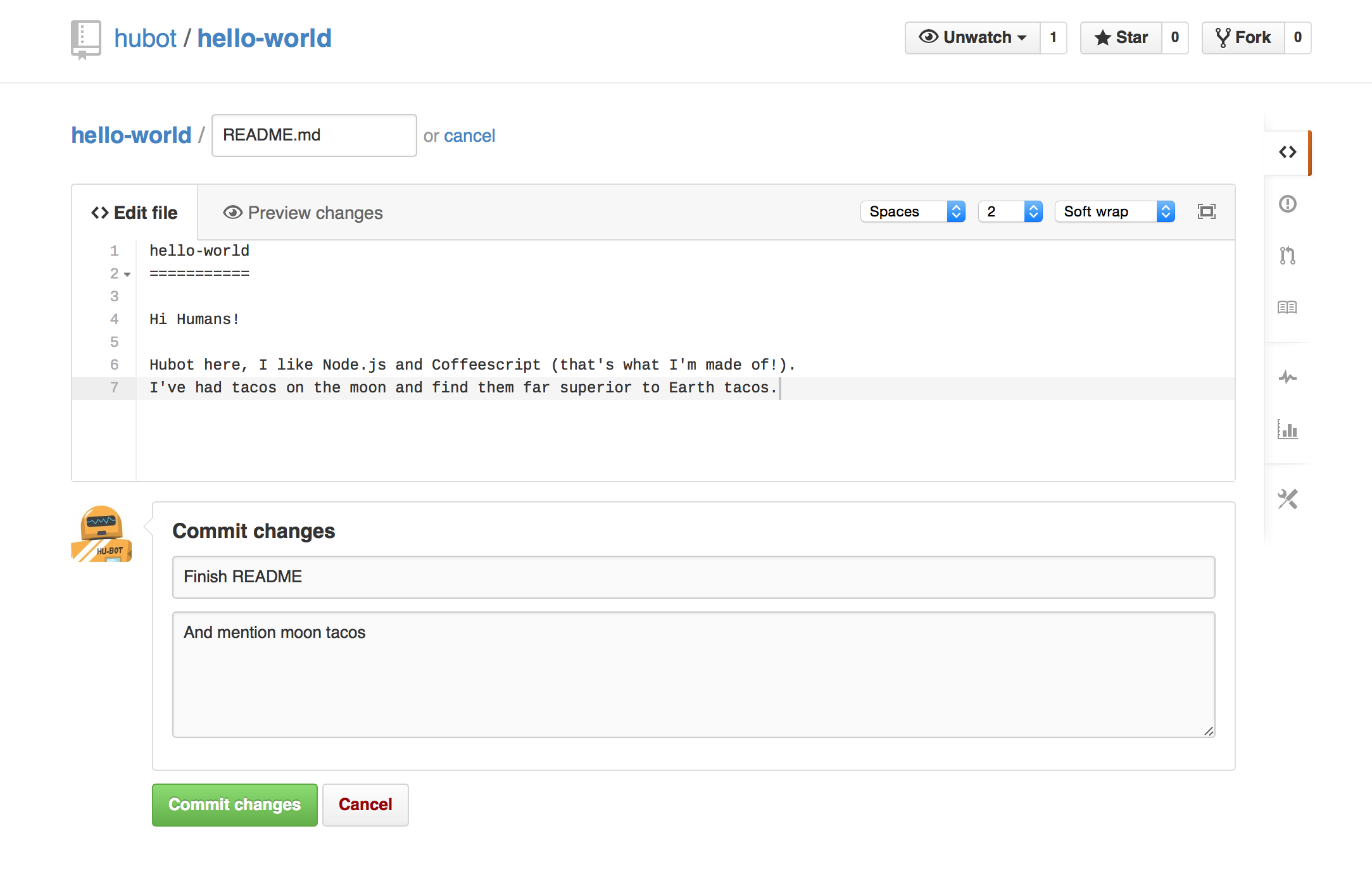
On GitHub, saved changes are called **commits**. Commits are pretty glorious, because a bunch of them together read like the history of your project.

Each commit has an associated **commit message**, which is a description explaining why a particular change was made. Thanks to these messages, you and others can read through commits and understand what you’ve done and why.

You should still be on the code view for your readme-edits branch, now make some changes!

**To commit changes**

1. Click the README file.
2. Click the pencil icon in the upper right corner of the file view to edit.
3. In the editor, write some text, tell a bit about yourself.
4. Write a commit message that describes your changes.



Click **Commit changes**. Now these changes have been made to the README file on your readme-edits branch and now this branch contains different content and commits than master (as it should!).

**Open a Pull Request**

Pull Requests are the heart of collaboration on GitHub. When you make a **pull request**, you’re proposing your changes and requesting that someone pull in your contribution - aka merge them into their branch. GitHub’s Pull Request feature allows you to compare the content on two branches. The changes, additions and subtractions, are shown in green and red and called **diffs** (differences).

As soon as you make a change, you can open a Pull Request. People use Pull Requests to start a discussion about commits (code review) even before the code is finished. This way you can get feedback as you go or help when you’re stuck.

By using GitHub’s [@mention system](https://github.com/blog/821-mention-somebody-they-re-notified) 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, as we’re doing here, a Pull Request in your own repository and merge it yourself. It’s a great way to learn to process before working on larger projects.

**Create a Pull Request for changes to the README**

*Click on the image for a larger version*

| **Step** | **Screenshot** |
| --- | --- |
| Click the Pull Request icon on the sidebar, then from the Pull Request page, click the green **New pull request** button. |  |
| Select the branch you made, readme-edits, to compare with master (the original). | [branch](https://guides.github.com/activities/hello-world/pick-branch.png) |
| Look over your changes in the diffs on the Compare page, make sure they’re what you want to submit. |  |
| When you’re satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button. | [create-pull](https://guides.github.com/activities/hello-world/create-pr.png) |
| Give your pull request a title and since it relates directly to an open issue, include “fixes #” and the issue number in the title. Write a brief description of your changes. | pr-form |

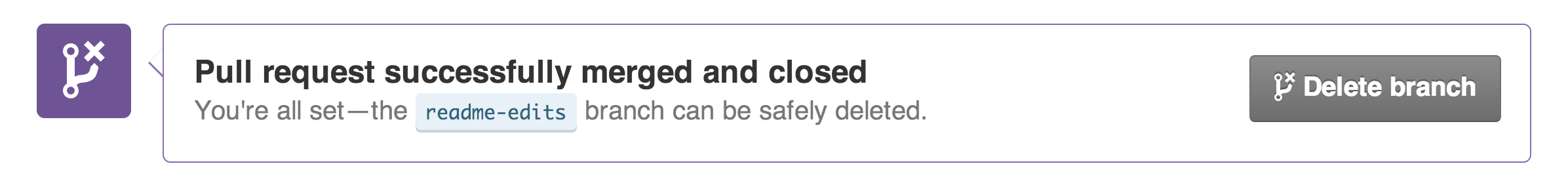
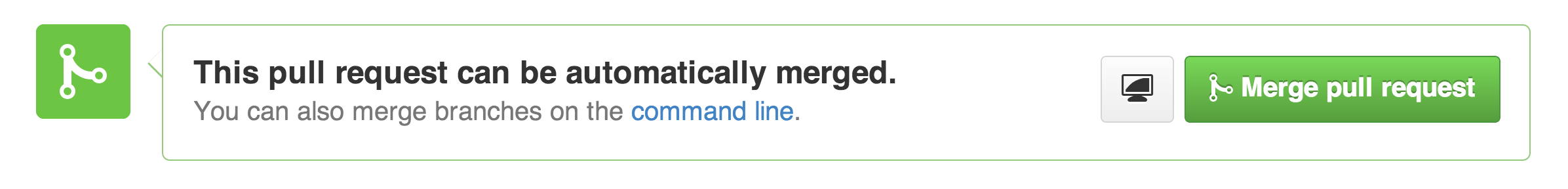
When you’re done with your message, click **Create pull request**!

**Tip**: You can use [emoji](https://github.com/blog/1289-emoji-autocomplete) and [drag and drop images and gifs](https://github.com/blog/1347-issue-attachments) onto comments and Pull Requests.

**Merge your Pull Request**

It’s time to bring your changes together – merge your readme-edits branch into the master branch.

1. Click the green 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.



If you revisit the issue you opened, it’s now closed! Because you included “fixes #1” in your Pull Request title, GitHub took care of closing that issue when the Pull Request was merged!

**Celebrate!**

You’ve learned to make a pull request on GitHub! :tada::octocat::zap:

For your humblebrag, here are the things you accomplished in this tutorial: created a repository, branch, issue, and pull request, then merged a pull request! Show off your new [contribution squares](https://help.github.com/articles/viewing-contributions)!

If you want to learn more about the power of pull requests, we recommend reading the [GitHub Flow Guide](http://guides.github.com/overviews/flow/). You might also visit [GitHub Explore](http://github.com/explore) and get involved in an Open Source project :octocat:

At the heart of GitHub is an open source version control system (VCS) called [*Git*](https://help.github.com/articles/github-glossary#git). Git is responsible for everything GitHub-related that happens locally on your computer.

If you're not comfortable using the command line right now, GitHub lets you complete many Git-related actions without using the command line, including:

* [Creating a repository](https://help.github.com/articles/create-a-repo)
* [Forking a repository](https://help.github.com/articles/fork-a-repo)
* [Being social](https://help.github.com/articles/be-social)

However, if you find that you need to use Git, we can help you set it up!

**Tips:**

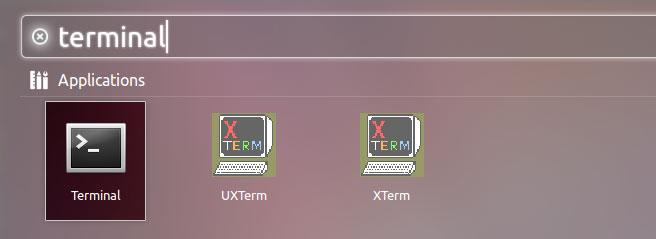
* GitHub has a [Mac](https://mac.github.com/) client! You can use it without ever touching the command line.
* To learn more about Git, see "[Getting Started - Git Basics](http://git-scm.com/book/en/Getting-Started-Git-Basics)" on the [git-scm](http://git-scm.com/) website.

**Tips:**

* GitHub has a [Windows](https://windows.github.com/) client! You can use it without ever touching the command line.
* To learn more about Git, see "[Getting Started - Git Basics](http://git-scm.com/book/en/Getting-Started-Git-Basics)" on the [git-scm](http://git-scm.com/) website.

**Tip:** to learn more about Git, see "[Getting Started - Git Basics](http://git-scm.com/book/en/Getting-Started-Git-Basics)" on the [git-scm](http://git-scm.com/) website.

#### [Input](https://help.github.com/articles/set-up-git/)

Code blocks like those on this page are part of a scripting language called Bash. To use Bash scripts, we need to use an application that comes with your Mac called Terminal, usually found at /Applications/Utilities. was installed with Git called Git Bash. Git Bash can be found in the start menu under "Git". comes with Linux called Terminal. comes with your OS called the command line.

echo 'This is input text'

# This tooltip tells you what's going on.

A line that begins with the dollar sign ($) indicates a line of Bash script you need to type. To enter it, type the text that follows the $, hitting the return key at the end of each line. You can hover your mouse over each line for an explanation of what the script is doing.

#### Output

# This is output text.

A line that does not begin with a $ is output text that is intended to give you information or tell you what to do next. We've colored output text green in these bootcamp tutorials.

#### User Specific Input

echo 'username'

# Outputs the text in the quotation marks.

Areas of yellow text represent your own personal info, repos, etc. If it is part of an input ($) line, you should replace it with your own info when you type it. If it is part of output text, it is just for your reference. It will automatically show your own info in TerminalTerminalGit Bashthe command line.

Good to know: There will be times when you type code, hit return, and all you are given is another prompt. Some actions that you execute in TerminalTerminalGit Bashthe command line don't have any output. Don't worry, if there is ever a problem with your code, TerminalTerminalGit Bashthe command line will let you know.

For security reasons, TerminalTerminalGit Bashthe command line will not display what you type when entering passwords. Just type your password and hit the return key.

### Setting up Git

1. [Download and install the latest version of Git](http://git-scm.com/downloads).
2. On your computer, open the **Terminal** application.
3. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
4. git config --global user.name "YOUR NAME"
5. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
6. git config --global user.email "YOUR EMAIL ADDRESS"
7. Download and install the latest version of [GitHub for Windows](https://windows.github.com/). This will automatically install Git and keep it up-to-date for you.
8. On your computer, open the **Git Shell** application.
9. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
10. git config --global user.name "YOUR NAME"
11. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
12. git config --global user.email "YOUR EMAIL ADDRESS"
13. [Download and install the latest version of Git](http://git-scm.com/downloads).
14. On your computer, open the **Terminal** application.
15. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
16. git config --global user.name "YOUR NAME"
17. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
18. git config --global user.email "YOUR EMAIL ADDRESS"
19. [Download and install the latest version of Git](http://git-scm.com/downloads).
20. On your computer, open your command line application.
21. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
22. git config --global user.name "YOUR NAME"
23. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
24. git config --global user.email "YOUR EMAIL ADDRESS"

### Next steps: Authenticating with GitHub from Git

When you connect to a GitHub repository from Git, you'll need to authenticate with GitHub using either HTTPS or SSH.

#### Connecting over HTTPS (recommended)

If you [clone with HTTPS](https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-https-recommended), you can [cache your GitHub password in Git](https://help.github.com/articles/caching-your-github-password-in-git) using a credential helper.

#### Connecting over SSH

If you [clone with SSH](https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-ssh), you must [generate SSH keys](https://help.github.com/articles/generating-ssh-keys) on each computer you use to push or pull from GitHub.

### Celebrate

Congratulations, you now have Git and GitHub all set up! What do you want to do next?

* **Set Up Git**
* [Create A Repository](https://help.github.com/articles/create-a-repo)
* [Fork A Repository](https://help.github.com/articles/fork-a-repo)
* [Be Social](https://help.github.com/articles/be-social)

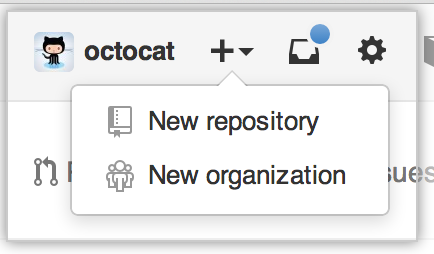
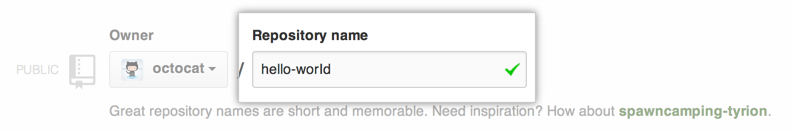
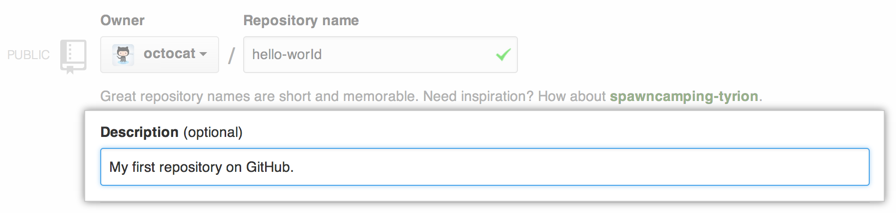
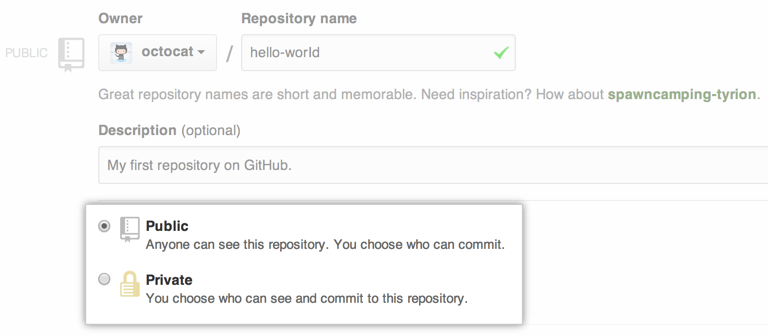
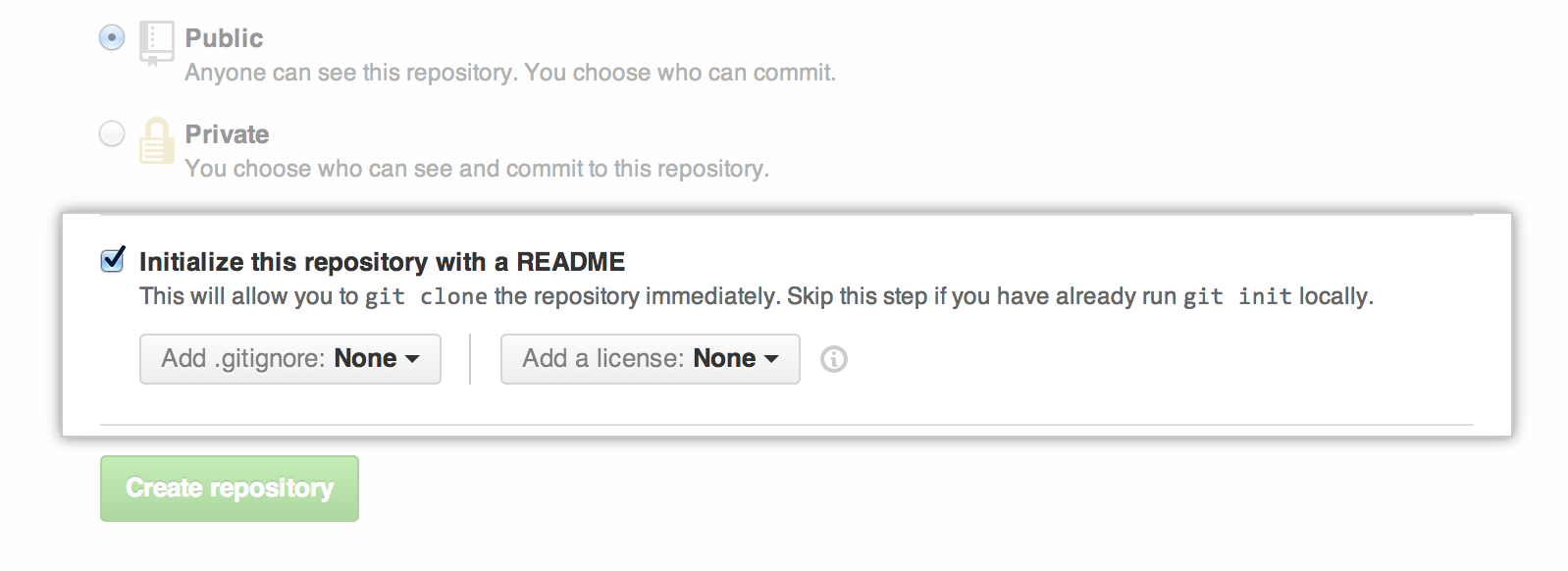
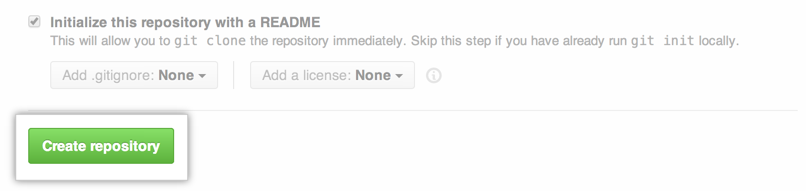
## Create A Repo

* [mac](https://help.github.com/articles/create-a-repo/#platform-mac)
* [windows](https://help.github.com/articles/create-a-repo/#platform-windows)
* [linux](https://help.github.com/articles/create-a-repo/#platform-linux)
* [all](https://help.github.com/articles/create-a-repo/#platform-all)

To put your project up on GitHub, you'll need a repository for it to live in. Let's create one!

On GitHub, you can store all kinds of projects in [*repositories*](https://help.github.com/articles/github-glossary#repository). Personal repositories belong to user accounts, so after you've [signed up for GitHub](https://help.github.com/articles/signing-up-for-a-new-github-account), you can create your first repository!

### Create a new repository on GitHub

1. In the upper-right corner of any page, click , and then click **New repository**.
2. Create a short, memorable name for your repository. For example, "hello-world".
3. Optionally, add a description of your repository. For example, "My first repository on GitHub."
4. Choose between creating a public or private repository.
   * **Public** repositories are a great choice for getting started! They're visible to any user on GitHub, so you can benefit from a collaborative community.
   * **Private** repositories require a little more setup. They're only available to you, the repository owner, as well as any collaborators you choose to share with. Private repositories are only available for paid accounts. For more information, see "[What plan should I choose?](https://help.github.com/articles/what-plan-should-i-choose)."
5. Select **Initialize this repository with a README**.
6. Click **Create repository**.

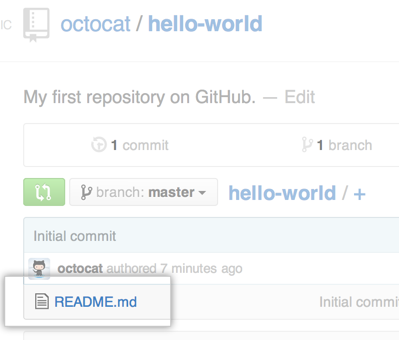
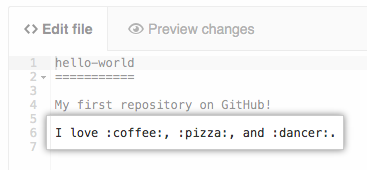
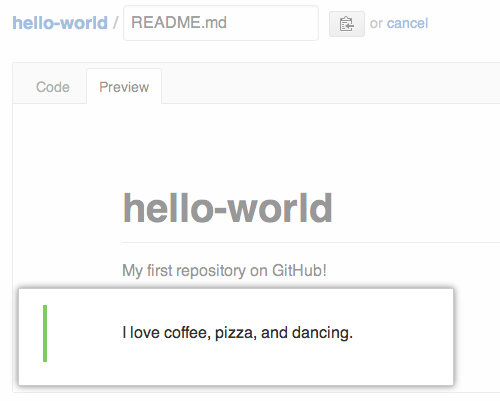
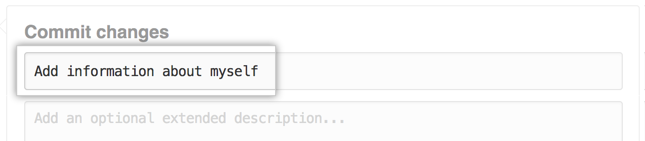
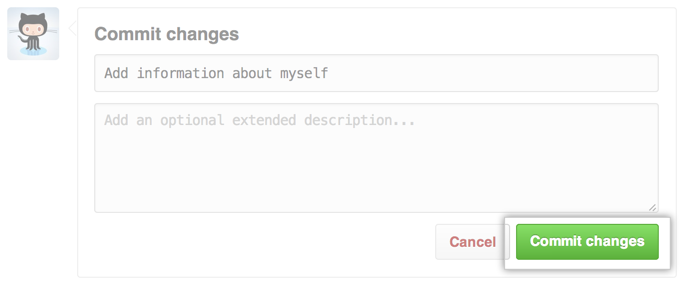
Congratulations! You've successfully created your first repository, and initialized it with a README file.

### Commit your first change

A [*commit*](https://help.github.com/articles/github-glossary#commit) is like a snapshot of all the files in your project at a particular point in time.

When you created your new repository, you initialized it with a README file. README files are a great place to describe your project in more detail, or add some documentation such as how to install or use your project. The contents of your README file are automatically shown on the front page of your repository.

Let's commit a change to the README file.

1. In your repository's list of files, click **README.md**.
2. Above the file's content, click .
3. On the **Edit file** tab , type some information about yourself.
4. Above the new content, click **Preview changes**.
5. Review the changes you made to the file. You'll see the new content in green.
6. At the bottom of the page, under "Commit changes," type a short, meaningful commit message that describes the change you made to the file.
7. 

## Set Up Git

* [mac](https://help.github.com/articles/set-up-git/#platform-mac)
* [windows](https://help.github.com/articles/set-up-git/#platform-windows)
* [linux](https://help.github.com/articles/set-up-git/#platform-linux)
* [all](https://help.github.com/articles/set-up-git/#platform-all)

At the heart of GitHub is an open source version control system (VCS) called *[Git](https://help.github.com/articles/github-glossary" \l "git)*. Git is responsible for everything GitHub-related that happens locally on your computer.

If you're not comfortable using the command line right now, GitHub lets you complete many Git-related actions without using the command line, including:

* [Creating a repository](https://help.github.com/articles/create-a-repo)
* [Forking a repository](https://help.github.com/articles/fork-a-repo)
* [Being social](https://help.github.com/articles/be-social)

However, if you find that you need to use Git, we can help you set it up!

**Tips:**

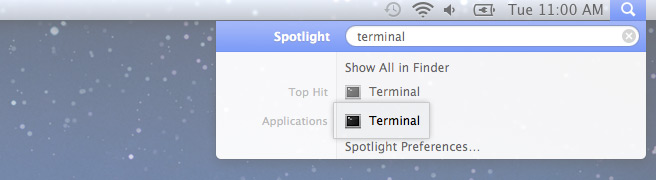
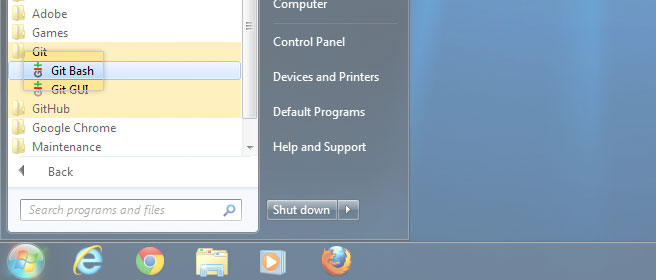
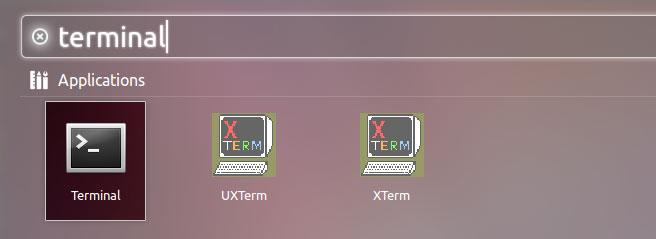
* GitHub has a [Mac](https://mac.github.com/) client! You can use it without ever touching the command line.
* To learn more about Git, see "[Getting Started - Git Basics](http://git-scm.com/book/en/Getting-Started-Git-Basics)" on the [git-scm](http://git-scm.com/) website.

**Tips:**

* GitHub has a [Windows](https://windows.github.com/) client! You can use it without ever touching the command line.
* To learn more about Git, see "[Getting Started - Git Basics](http://git-scm.com/book/en/Getting-Started-Git-Basics)" on the [git-scm](http://git-scm.com/) website.

**Tip:** to learn more about Git, see "[Getting Started - Git Basics](http://git-scm.com/book/en/Getting-Started-Git-Basics)" on the [git-scm](http://git-scm.com/) website.

#### [Input](https://help.github.com/articles/set-up-git/)

Code blocks like those on this page are part of a scripting language called Bash. To use Bash scripts, we need to use an application that comes with your Mac called Terminal, usually found at /Applications/Utilities. was installed with Git called Git Bash. Git Bash can be found in the start menu under "Git". comes with Linux called Terminal. comes with your OS called the command line.

echo 'This is input text'

# This tooltip tells you what's going on.

A line that begins with the dollar sign ($) indicates a line of Bash script you need to type. To enter it, type the text that follows the $, hitting the return key at the end of each line. You can hover your mouse over each line for an explanation of what the script is doing.

#### Output

# This is output text.

A line that does not begin with a $ is output text that is intended to give you information or tell you what to do next. We've colored output text green in these bootcamp tutorials.

#### User Specific Input

echo 'username'

# Outputs the text in the quotation marks.

Areas of yellow text represent your own personal info, repos, etc. If it is part of an input ($) line, you should replace it with your own info when you type it. If it is part of output text, it is just for your reference. It will automatically show your own info in TerminalTerminalGit Bashthe command line.

Good to know: There will be times when you type code, hit return, and all you are given is another prompt. Some actions that you execute in TerminalTerminalGit Bashthe command line don't have any output. Don't worry, if there is ever a problem with your code, TerminalTerminalGit Bashthe command line will let you know.

For security reasons, TerminalTerminalGit Bashthe command line will not display what you type when entering passwords. Just type your password and hit the return key.

### Setting up Git

1. [Download and install the latest version of Git](http://git-scm.com/downloads).
2. On your computer, open the **Terminal** application.
3. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
4. git config --global user.name "YOUR NAME"
5. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
6. git config --global user.email "YOUR EMAIL ADDRESS"
7. Download and install the latest version of [GitHub for Windows](https://windows.github.com/). This will automatically install Git and keep it up-to-date for you.
8. On your computer, open the **Git Shell** application.
9. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
10. git config --global user.name "YOUR NAME"
11. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
12. git config --global user.email "YOUR EMAIL ADDRESS"
13. [Download and install the latest version of Git](http://git-scm.com/downloads).
14. On your computer, open the **Terminal** application.
15. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
16. git config --global user.name "YOUR NAME"
17. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
18. git config --global user.email "YOUR EMAIL ADDRESS"
19. [Download and install the latest version of Git](http://git-scm.com/downloads).
20. On your computer, open your command line application.
21. Tell Git your name so your commits will be properly labeled. Type everything after the $ here:
22. git config --global user.name "YOUR NAME"
23. Tell Git the email address that will be associated with your Git commits. The email you specify should be the same one found in your [email settings](https://help.github.com/articles/adding-an-email-address-to-your-github-account/). To keep your email address hidden, see "[Keeping your email address private](https://help.github.com/articles/keeping-your-email-address-private)".
24. git config --global user.email "YOUR EMAIL ADDRESS"

### Next steps: Authenticating with GitHub from Git

When you connect to a GitHub repository from Git, you'll need to authenticate with GitHub using either HTTPS or SSH.

#### Connecting over HTTPS (recommended)

If you [clone with HTTPS](https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-https-recommended), you can [cache your GitHub password in Git](https://help.github.com/articles/caching-your-github-password-in-git) using a credential helper.

#### Connecting over SSH

If you [clone with SSH](https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-ssh), you must [generate SSH keys](https://help.github.com/articles/generating-ssh-keys) on each computer you use to push or pull from GitHub.