

SWEN90016

Software Processes & Project Management

Configuration
and
Version Control



Become familiar with
Continuous Integration
&
GIT

<https://github.com/microsoft>





The Version Control System Continuous Integration (CI)

(not be confused with CI- configuration items –lecture)

- Practice of automating the integration of code changes from multiple contributors into a single software project.
- [DevOps best practice](#), developers merge code changes into a central repository where builds and tests then run.
- Automated tools are used to assert the new code's correctness before integration.

So what is Git?

- Version Control System

who, what, where, when of code

- Made by Linus Torvalds!

famous guru - transformed technology twice

Created kernel for Linux OSs , Android and Chrome OS

Created Git, the source code management system



meet Tux, slightly overweight penguin



- WHAT: identify artefacts under versioning control
- WHERE: items stored in REPO: version control repository
- WHO: tags items with your name (collection remains consistent)
- WHEN: tags item with date (versions exist)
- WHY: quality code sharing, back up, roll-back, parallel branching

The “why” is recorded by your comment on the commit!

- Distributed (everyone has their own code repository local to them!)
- Open Source (everyone likes open source code 😊)
- Bomb proof

GitHub hit by Massive DDoS Attack From China

Friday, March 27, 2015 Mohit Kumar

[Tweet](#) [G+ Share](#) [Share](#) 29 [in Share](#) [f Share](#) [Share](#)



1.git init

Initialize local repository called **master**

1.git add *filename*

Track changes made to the contents of this file

1.git commit -m “*message*”

Save this file in local repository

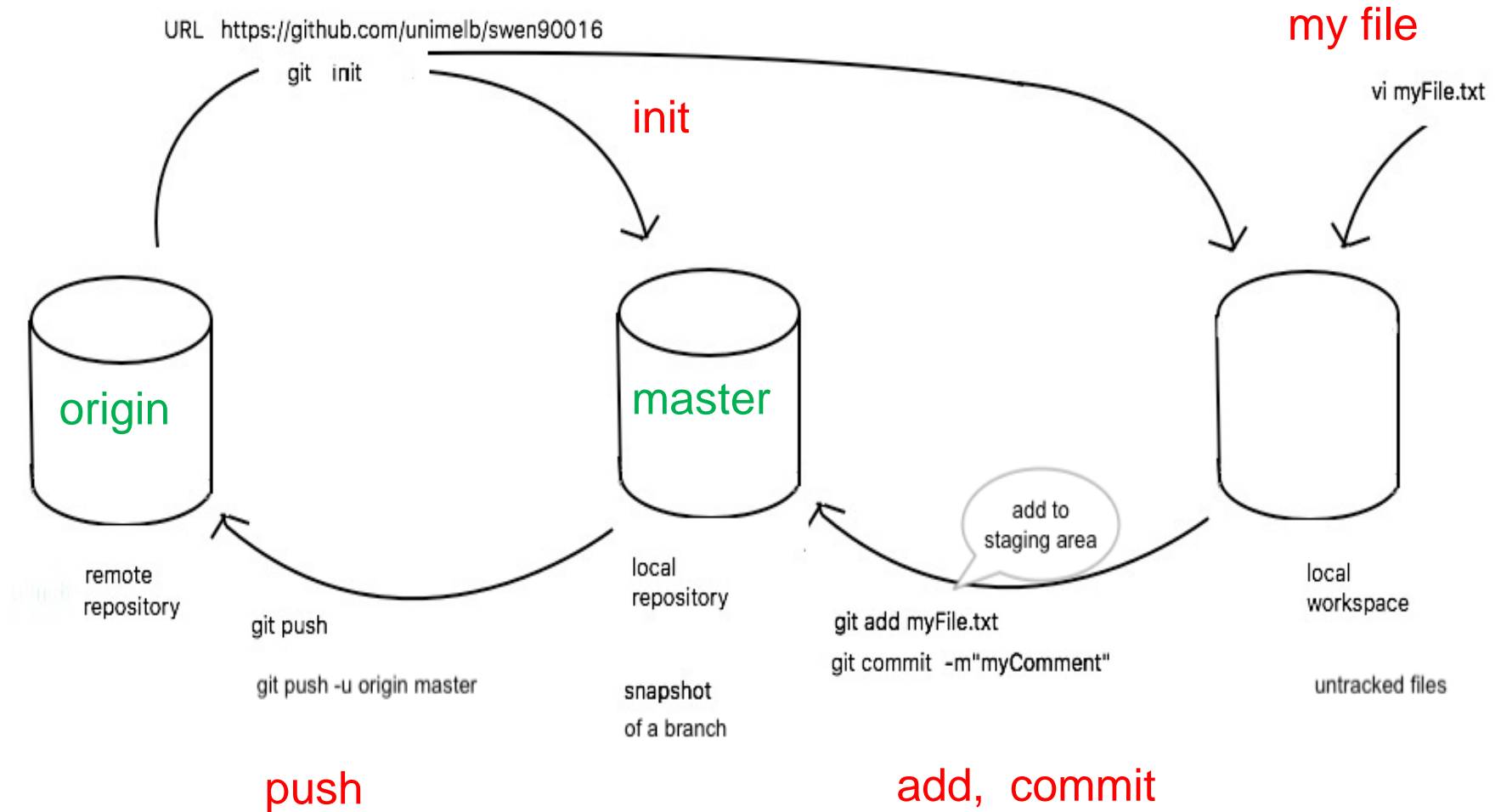
1.git push

Save this collection

- to remote repository called **origin**
- from local repository called **master**



REPOURNA



No Coding Necessary

To complete this tutorial, you need a <https://github.com/microsoft> 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 organise a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.

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.

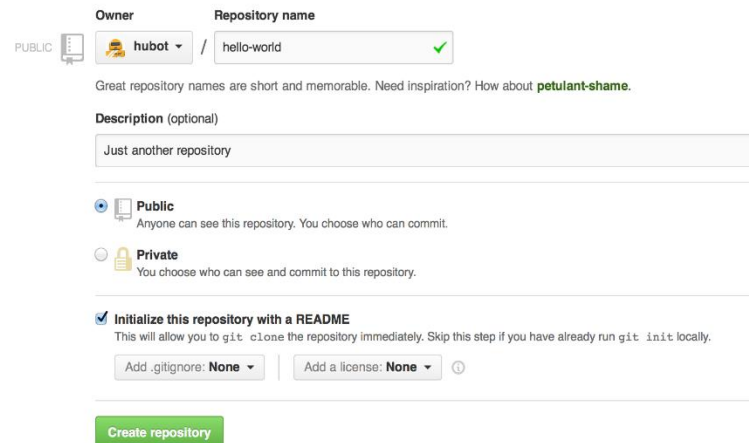
To Create a New Repository

In the upper right corner, next to your avatar or identicon, click + and then select **New repository**.

Name your repository *hello-world*.

Write a short description.

Select **Initialize this repository with a README**.



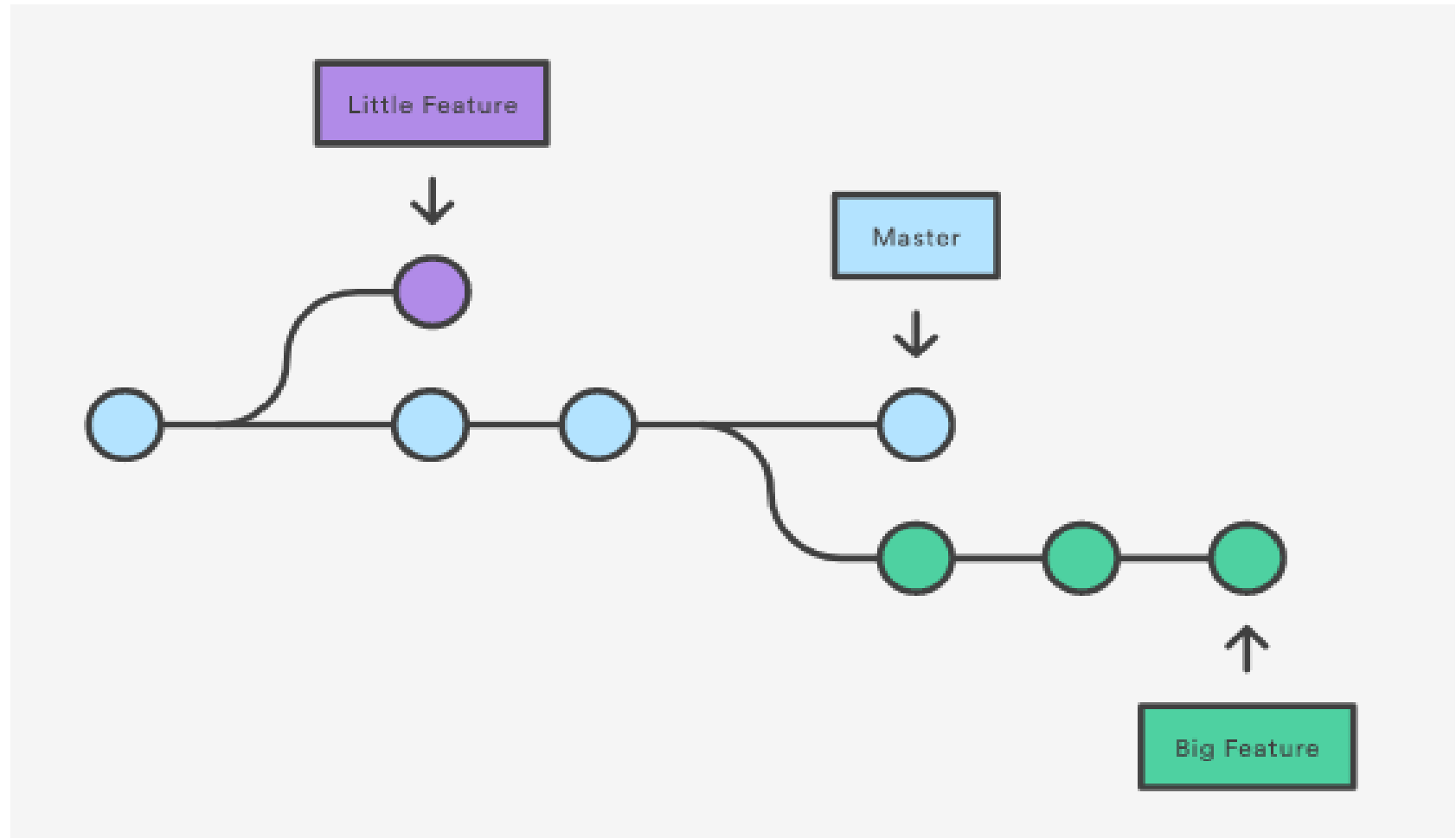
The screenshot shows the GitHub 'Create new repository' form. At the top, there are two tabs: 'Owner' and 'Repository name'. Under 'Owner', there is a 'PUBLIC' button and a dropdown menu showing 'hubot'. Under 'Repository name', there is a text input field containing 'hello-world' with a green checkmark to its right. Below this, a message says: 'Great repository names are short and memorable. Need inspiration? How about [petulant-shame](#).' The 'Description (optional)' section has a text input field containing 'Just another repository'. Below the description, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option has a subtext: 'Anyone can see this repository. You choose who can commit.' The 'Private' option has a subtext: 'You choose who can see and commit to this repository.' Below the visibility options, there is a checked checkbox for 'Initialize this repository with a README'. A subtext for this checkbox says: 'This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.' At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None'. A green 'Create repository' button is at the very bottom.

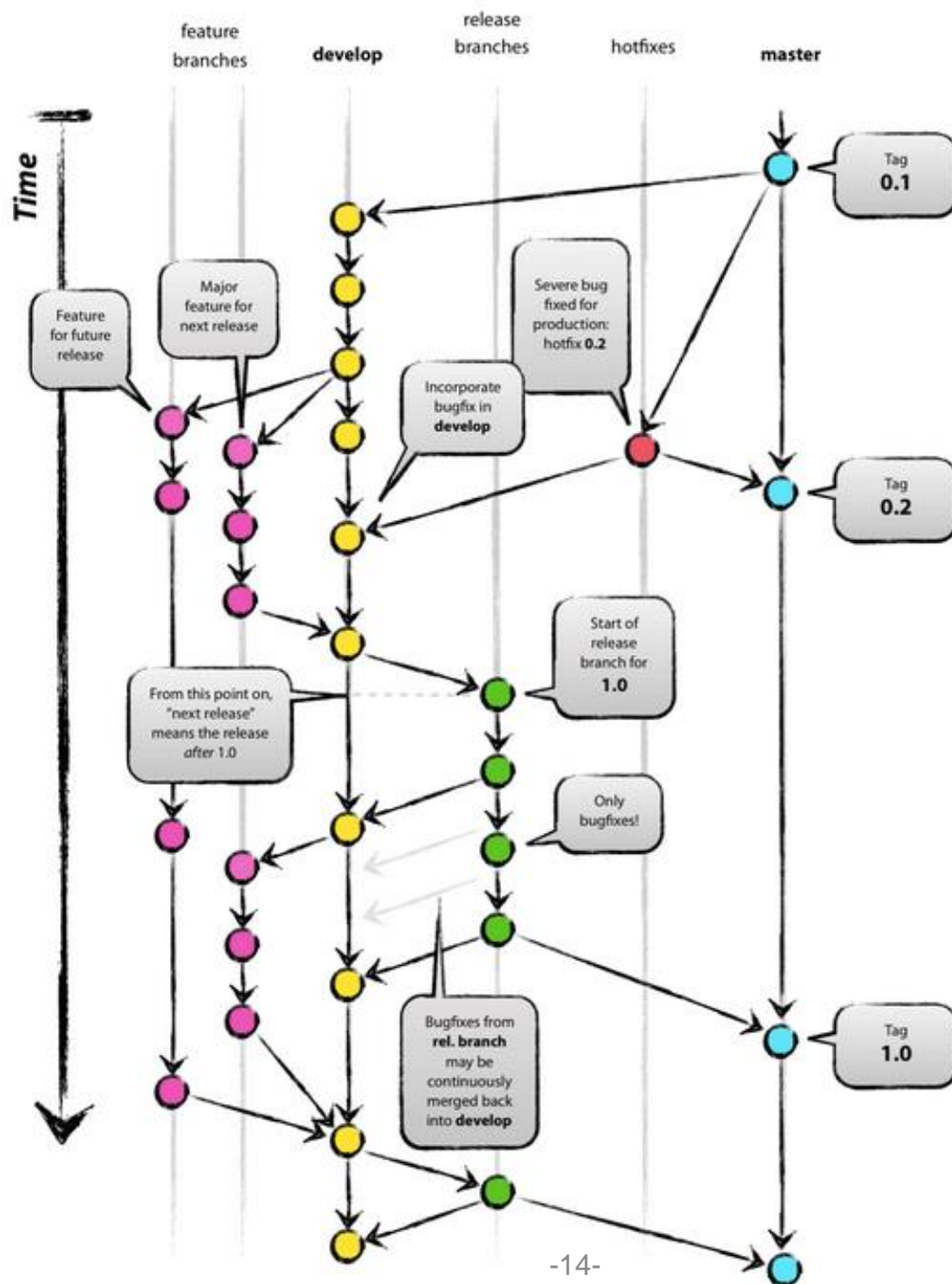
Step 2. Create a Branch

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.





Step 2. Create a Branch

This diagram shows:

The master branch

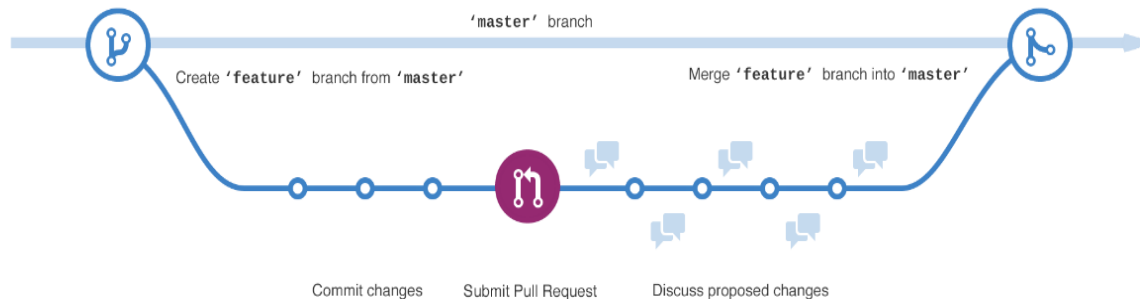
A new branch called feature (because we're doing 'feature work' on this branch)

The journey that feature takes before it's merged into master

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

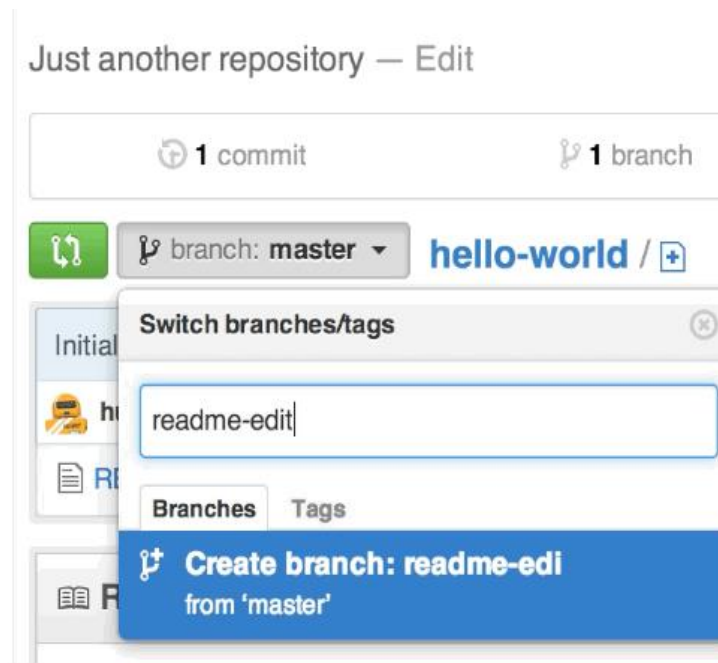
story.txt

story-joe-edit.txt



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



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.

Step 3. Make and Commit Changes

Click the README.md file.

Click the pencil icon in the upper right corner of the file view to edit.

In the editor, write a bit about yourself.

Write a commit message that describes your changes.

Click **Commit changes** button.

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*.



hubot / hello-world

Unwatch 1Star 0Fork 0

<> Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

Settings

hello-world / README.md or cancel

<> Edit file

Preview changes

Spaces 2Soft wrap

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



Commit changes

Finish README

And mention moon tacos

☒ 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 changesCancel

Step 4. 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.

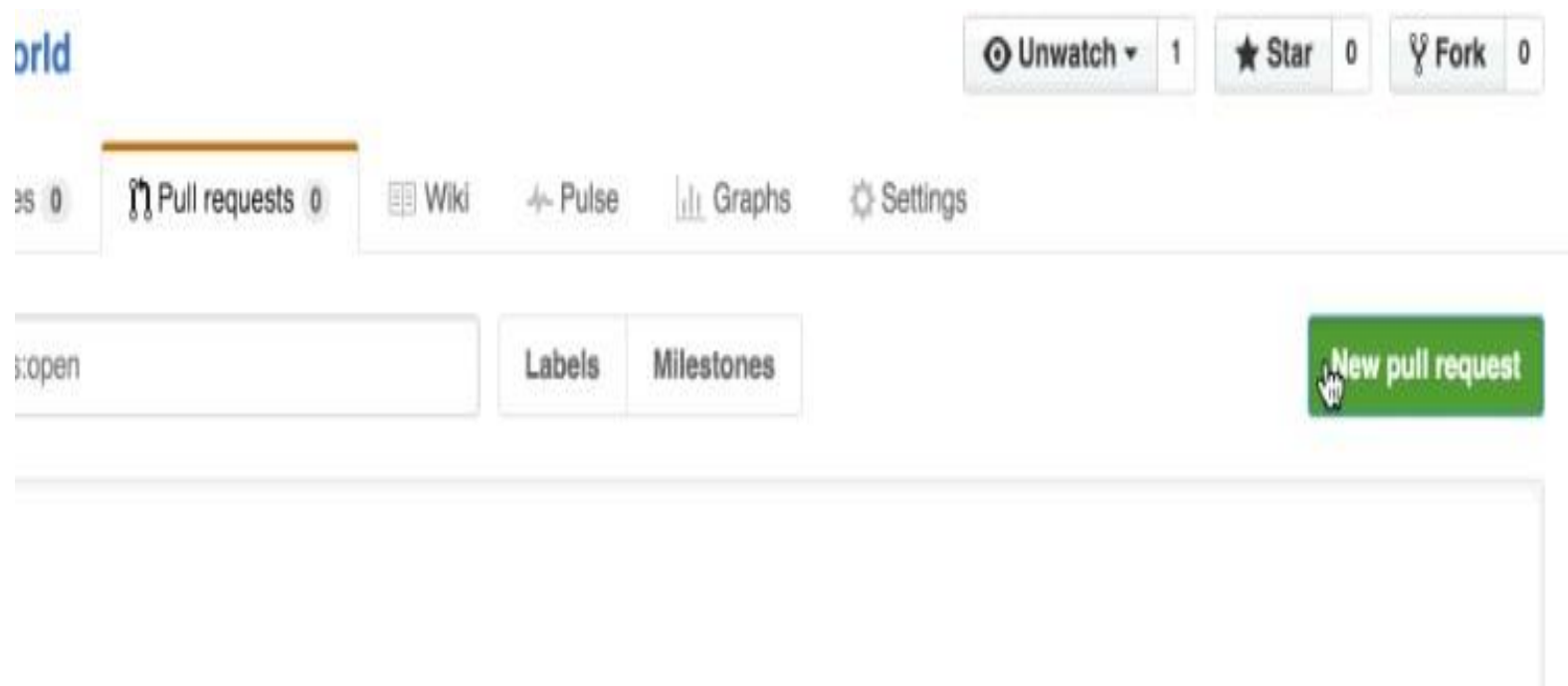
By using GitHub's [@mention system](#) in your pull request message, you can ask for feedback from specific people or teams, whether they're down the hall or 10 time zones away.

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



Open a Pull Request for Changes to the README

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

EXAMPLE COMPARISONS

 **readme-edits**

4 minutes ago



Open a Pull Request for Changes to the README

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

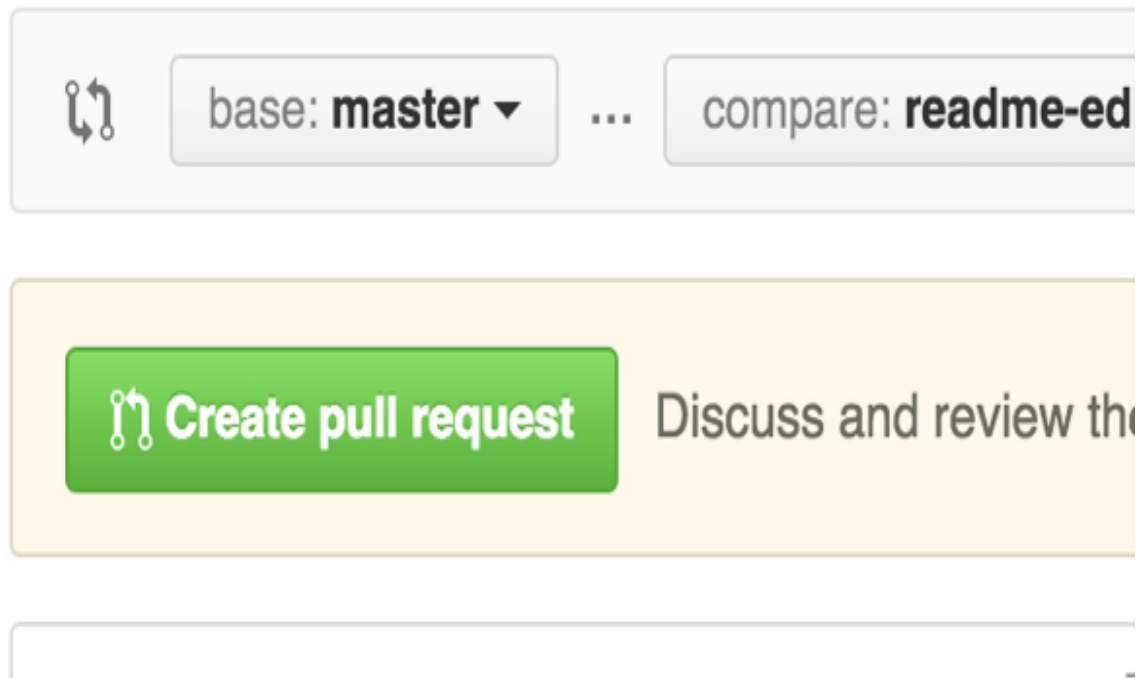
hubot Finish README ...

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.

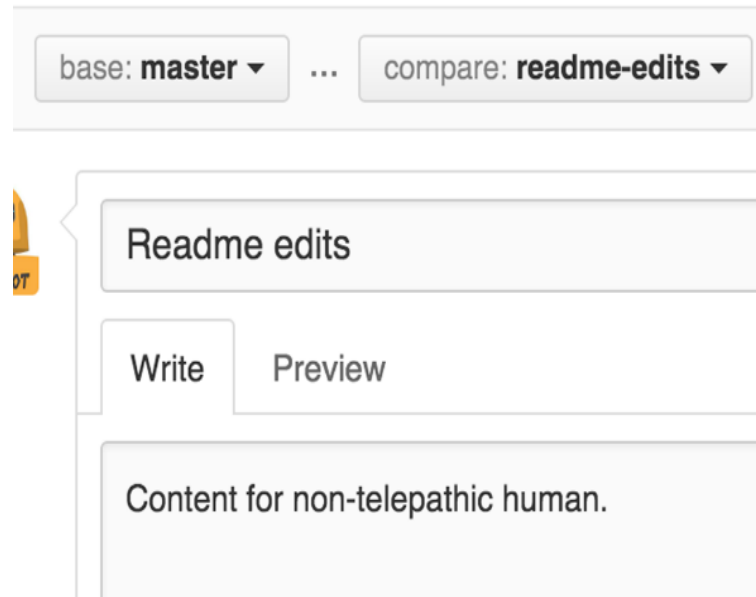
Open a Pull Request for Changes to the README

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



Open a Pull Request for Changes to the README

Give your pull request a title and write a brief description of your changes
When you're done with your message, click **Create pull request!**

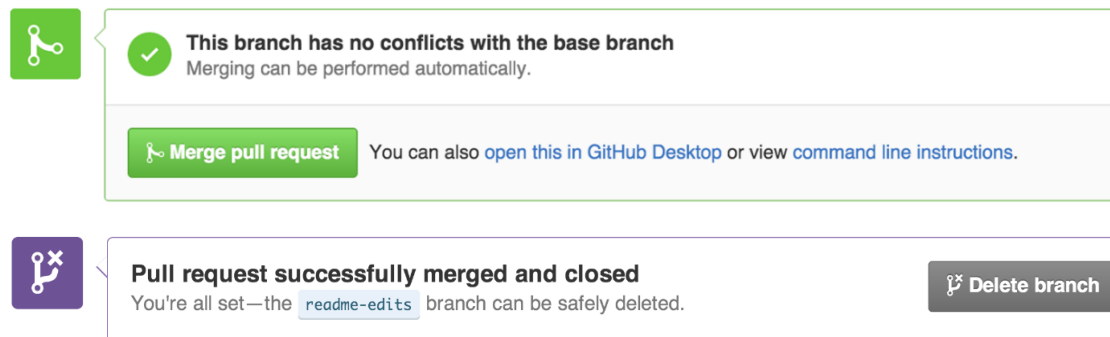


The screenshot shows the GitHub Pull Request interface. At the top, there are two dropdown menus: 'base: master' and 'compare: readme-edits'. Below these, there is a yellow icon of a notepad with a pencil. The main content area is divided into two tabs: 'Write' and 'Preview'. The 'Write' tab is active, showing a text input field with the placeholder text 'Content for non-telepathic human.'.

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.

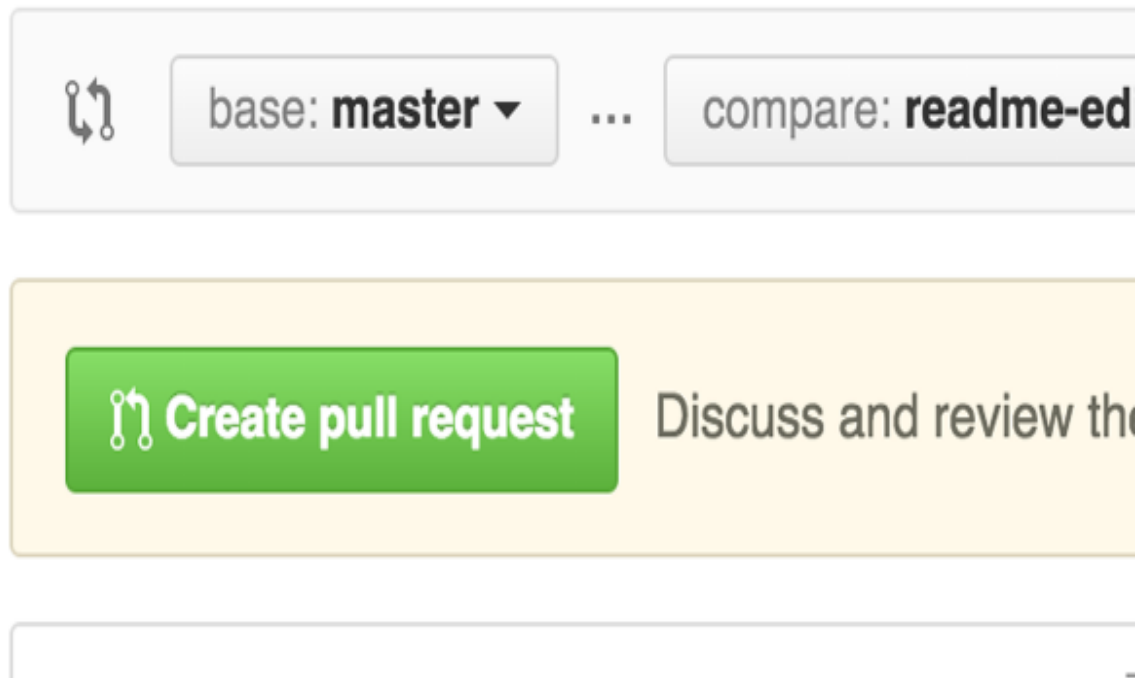
1. Click the green **Merge pull request** button to merge the changes into master.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.





Open a Pull Request for Changes to the README

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





Due in your respective week 12 workshop

What to expect:

- All team members need to be at the workshop
- Nominate a team member to share their screen and present the website
- You have 5-7 minutes to present
- Present your website/app

Done!

References:

<https://www.atlassian.com/git/tutorials>

[https://www.atlassian.com/continuous-delivery/continuous-integration#:~:text=Continuous%20integration%20\(CI\)%20is%20the,builds%20and%20tests%20then%20run.](https://www.atlassian.com/continuous-delivery/continuous-integration#:~:text=Continuous%20integration%20(CI)%20is%20the,builds%20and%20tests%20then%20run.)

<https://git-scm.com/>

[http://thehackernews.com/2015/03/github-hit-by-massive-ddos-attack from 27.html](http://thehackernews.com/2015/03/github-hit-by-massive-ddos-attack-from_27.html)