



# Using GitHub for Collaboration

Workshop Taught by Jami Jackson Mulgrave



# Outline

- Creating an Account on GitHub
- Install Git
- The GitHub Flow
- Contributing to a Project in GitHub
- Maintaining a Project in GitHub
- Team Collaboration Tools
- Documenting your Projects on GitHub
- Being Social



# Create a GitHub Account

- GitHub <https://github.com/> offers free accounts to work on public and open source projects, as well as paid accounts that offer unlimited private repositories.
- NCSU GitHub <https://github.ncsu.edu/> is self-contained and self-managed by NC State University. It gives users the opportunity to create an unlimited number of repositories – both public and private.
- NCSU GitHub is specifically designed to not compete with Github.com for public offerings. “Public” inside of NCSU GitHub is public to the institution, not the world.
- If a given project is truly open source, then it may be more appropriate to host it at GitHub.com with a personal account.
- Any faculty, staff or student of NC State University is allowed a free account with NCSU GitHub.



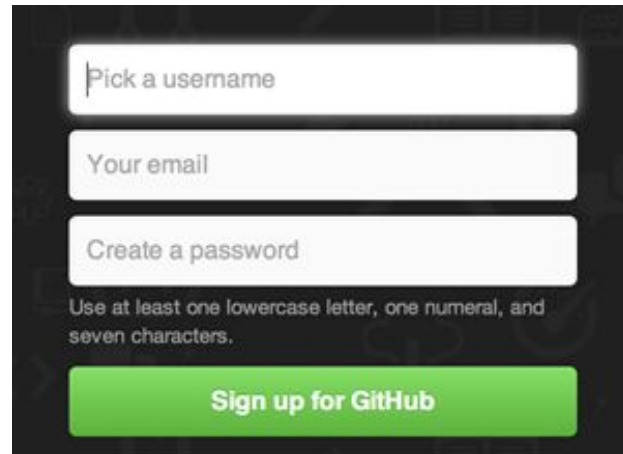
# Create a GitHub Account

- For NCSU GitHub account:
  - If you are a student or faculty member, you already have access to GitHub Enterprise. Simply login at <https://github.ncsu.edu/> with your Unity username and password. Your GitHub Enterprise Account will be created instantly after login.
  - “Account Setup and Configuration” is in my GitHub account <https://github.ncsu.edu/injacks3/GitHub-Workshops>



# Create a GitHub Account

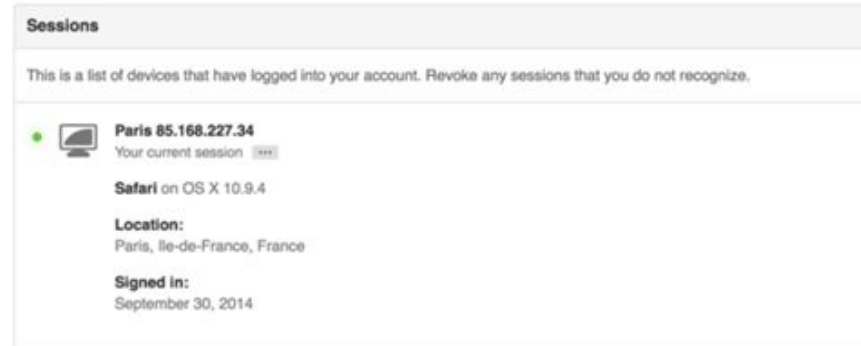
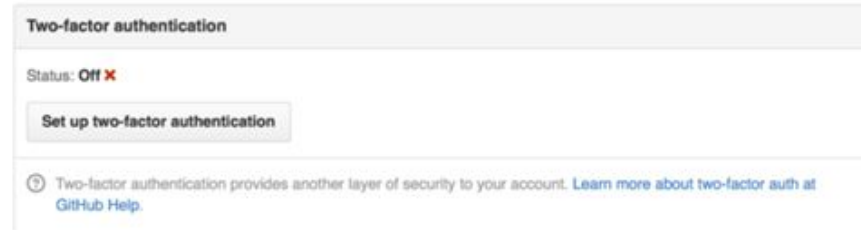
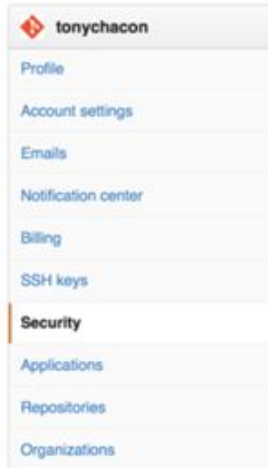
- For Public GitHub.com account:
  - Simply visit <https://github.com>, choose a username that isn't already taken, provide an email address and a password, and click the big green “Sign up for GitHub” button.



A screenshot of the GitHub sign-up form. It features three input fields: 'Pick a username', 'Your email', and 'Create a password'. Below the password field is a note: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom is a large green button labeled 'Sign up for GitHub'.

# Create a GitHub Account

- Add an Avatar to your profile
- Add your email addresses
- Set up Two-factor Authentication



# Housekeeping



- **Sample project for the exercises:**
  - You can use your own projects or download a file from my GitHub account to use as a sample project here: <https://github.ncsu.edu/injacks3/GitHub-Workshops>
  - This account also has all of the exercises, additional resources, and I will post the slides.



# Housekeeping: Install Git

- Make sure Git is installed
  - See “Installing Git” on my GitHub account  
<https://github.ncsu.edu/jnjacks3/GitHub-Workshops>
  - For Mac
    - On Mavericks (10.9+) you can try to run `git` from the Terminal or download at the Git website at <http://git-scm.com/download/mac> or install GitHub for Mac at <http://mac.github.com>
  - For Windows
    - Download at <http://git-scm.com/download/win> or install GitHub for Windows at <http://windows.github.com>





# Command Line

- Git for Windows includes the Git Bash command line.
- The default command line interface for Mac OS X is Terminal, which uses Bash. To open it, go to your Utilities folder. There you should see the icon for Terminal.
- See “Command Line Cheat Sheet” in my GitHub account for a quick reference.



# First Time Git Setup

- Now that Git is setup, we will customize your Git environment. You should have to do these things only once on any given computer; they'll stick around between upgrades. You can also change them at any time by running through the commands again.
- Git comes with a tool called `git config` that lets you get and set configuration variables that control all aspects of how Git looks and operates.



# First Time Git Setup

- You should first set your user name and email address.
  - This is important because every Git commit uses this information, and it's baked into the commits you start creating.

```
$ git config --global user.name "John Doe"
```

```
$ git config --global user.email johndoe@example.com
```

- You need to do this only once if you pass the --global option, because then Git will always use that information for anything you do on that system.
- If you want to override this with a different name or email address for specific projects, you can run the command without the --global option when you're in that project.



# The GitHub Flow

- GitHub is designed around a particular collaboration workflow, centered on **Pull Requests**. This flow works whether you're collaborating with a tightly-knit team in a single shared repository, or a globally-distributed company or network of strangers contributing to a project through dozens of forks.
- Here's how it generally works:
  1. Fork the project
  2. Create a branch from master.
  3. Make some commits to improve the project.
  4. Push this branch to your GitHub project.
  5. Open a Pull Request on GitHub.
  6. Discuss, and optionally continue committing.
  7. The project owner merges or closes the Pull Request.



# Contributing to a Project in GitHub - Forking

- If you want to contribute to an existing project, and you don't have write access to the project, you can “fork” the project. When you “fork” a project, **GitHub will make a copy of the project from the original repository that is entirely yours**; it lives in your namespace, and you can push to your [GitHub.com](https://github.com) account.
- This way, project owners don't have to worry about adding users as collaborators and giving them write access.
- People can fork a project, push their changes to [GitHub.com](https://github.com), and contribute their changes back to the original repository by creating what's called a **Pull Request**, which we'll cover next.
- This opens up a discussion thread with a review, and the owner and the contributor can then communicate about the changes until the owner is happy with them, at which point the owner can merge the changes into the original repository.

# Contributing to a Project in GitHub - Exercise 1: Forking

- To fork a project, visit the project page, like <https://github.ncsu.edu/injacks3/GitHub-Workshops>, and click the “Fork” button at the top-right of the page.
- After a few seconds, you’ll be taken to your new project page, with your own writeable copy of the code on [GitHub.com](https://github.com).





# Contributing to a Project in GitHub - Exercise 2: Clone Your Fork

- To clone your fork to your local computer
  - If you're using GitHub for Desktop application, navigate over to the bottom of the right hand sidebar and click **Clone in Desktop**.
  - Or if you have Git installed and configured, decide where you want the repository saved and use the **git clone** command and include the URL link to the repository.

```
$ git clone https://github.ncsu.edu/jnjacks3/GitHub-Workshops  
$ cd GitHub-Workshops
```

- When you **git clone** to a remote repository, you'll be asked for your GitHub username and password.
- You can use a **credential helper** to tell Git to remember your GitHub username and password.



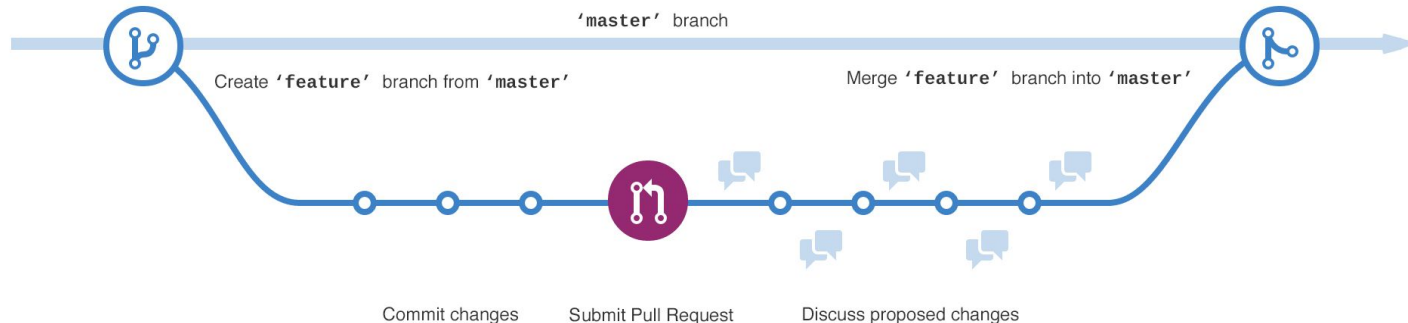
# Contributing to a Project in GitHub - Branching

- **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.



# Contributing to a Project in GitHub - Branching

- This diagram shows:
  - The **master** branch
  - A new branch called **feature**
  - The journey that **feature** takes before it's merged into **master**





## Contributing to a Project in GitHub - Exercise 3: Create a Branch on GitHub

- To create a new branch
  1. Go to the repository “**GitHub-Workshops**”.
  2. Click the drop down at the top of the file list that says **branch: master**.
  3. Type a branch name, **develop**, into the new branch textbox.
  4. Select the blue **Create branch** box or hit “**Enter**” on your keyboard.
- Now you have two branches, **master** and **develop**.



## Exercise 4: Create a Branch on Git

- You can also create a branch on your local computer and push the branch to [GitHub.com](https://github.com) using the `git branch` and `git push` commands.

```
$ git branch develop2  
$ git push origin develop2
```

- Now we can go to the repository on [GitHub.com](https://github.com) and you will see the new branch there.
- You can also add the **develop** branch to your local computer from [GitHub.com](https://github.com) by using the `git fetch` (or `git pull`) command

```
$ git pull
```



## Note: Git Pull vs Git Fetch

- When you use `git pull`, Git tries to automatically do your work for you. `Git pull` automatically merges the commits without letting you review them first. If you don't closely manage your branches, you may run into frequent conflicts.
- When you `git fetch`, Git gathers any commits from the target branch that do not exist in your current branch and stores them in your local repository. However, it does not merge them with your current branch. This is particularly useful if you need to keep your repository up to date, but are working on something that might break if you update your files.
- You would first `git fetch`, look at the differences using `git diff`, and then integrate the commits into your master branch, using `git rebase`.
- Read more about `git fetch`, `git diff`, `git pull`, `git merge`, and `git rebase` at <https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell>

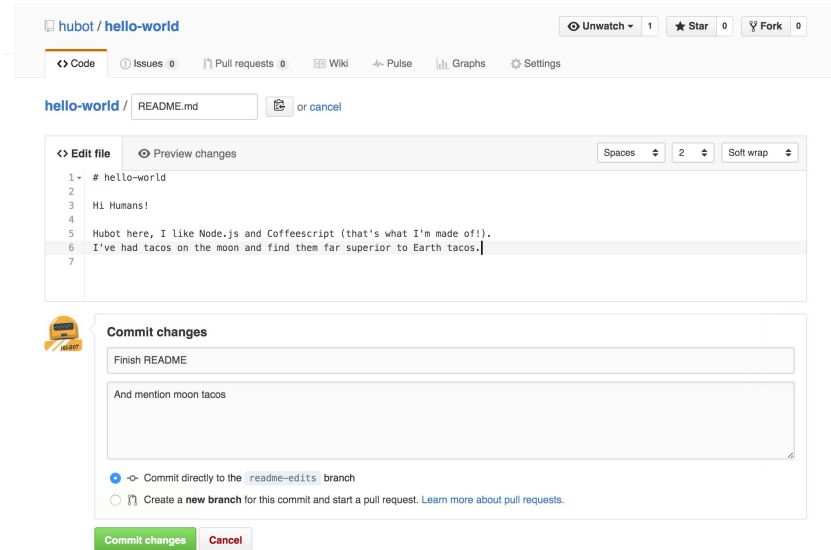


# Contributing to a Project in GitHub - Commit and Push Changes

- You can either make changes directly in GitHub or you can make changes in Git and push to 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.

# Contributing to a Project in GitHub - Exercise 5: Commit Changes on GitHub

1. In the **develop** branch, 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
5. Click **Commit changes** button.



The screenshot shows the GitHub web interface for the repository 'hubot / hello-world'. At the top, there are navigation links for Code, Issues, Pull requests, Wiki, Pulse, Graphs, and Settings. Below these, the file 'hello-world / README.md' is selected. The file content is displayed in a code editor with line numbers 1 through 7. The text in the file is: 1 # hello-world, 2 Hi Humans!, 3, 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 code editor, there is a 'Commit changes' dialog box. It has a text input field for the commit message, which currently contains 'Finish README' and 'And mention moon tacos'. At the bottom of the dialog, there are two radio buttons: 'Commit directly to the readme-edits branch' (which is selected) and 'Create a new branch for this commit and start a pull request'. At the very bottom of the dialog are two buttons: 'Commit changes' and 'Cancel'.



# Contributing to a Project in GitHub - Push changes

- For this example, we are already in the [GitHub.com](https://github.com) browser, so once we commit the changes, we don't need to push them to [GitHub.com](https://github.com) because the changes are already in the the remote server.
- To bring the changes to your local computer,

```
$ git checkout develop
```

```
$ git pull
```



## Contributing to a Project in GitHub - Exercise 6: Commit and Push Changes to GitHub

- What if we make the updates in a branch on our local computer and then push them to GitHub?
- On your local computer, you can work in the **develop** or **develop2** branch by typing `git checkout develop` or `git checkout develop2`.
- Here are you in the **develop** branch on your local computer and you make the edits to the README.md file that you want separate from the master branch.

```
$ git checkout develop
$ git add README.md
$ git commit -m 'Update readme develop'
$ git push origin develop
```

- Now these updates are in the **develop** branch on GitHub.





# Contributing to a Project in GitHub - Creating a Pull Request

- Now that you have made 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 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.
- By using GitHub's **@mention system** in your pull request message, you can ask for feedback from specific people or teams.




# Creating a Pull Request


- You can create a **pull request** on GitHub for 2 reasons:
  - To merge your **develop** branch into your **master** branch, after you've made the changes.
  - To use your **develop** branch to collaborate by suggesting changes to the project owner's original repository in which you are comparing your **develop** branch to their **master** branch.
- See <https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging> to learn more about merging branches on Git.


## Exercise 7: Creating a Pull Request to Merge Branches


- In the repository “GitHub-Workshops” on [GitHub.com](https://github.com), you’ll see a banner indicating that you’ve recently pushed a new branch, and that you can submit this branch “**upstream**,” to the project owner or click
- Click on **Create Pull Request** or **Compare and Pull Request** for the develop branch.

Your recently pushed branches:

 **new-file** (less than a minute ago)



 **Compare & pull request**

 branch: **master** ▾

**Spoon-Knife** / 


This branch is 0 commits ahead and 0 commits behind master

Adding a brand new file

 Pull Request  Compare

## Exercise 7: Creating a Pull Request to Merge Branches

- To merge your develop branch into your master branch on GitHub, change the base fork to your repository (not the one you forked). GitHub tells you if you can merge the branches.
- Finally, click on **Create pull request** and then **merge pull request** and then **confirm merge**. Now it tells you that the develop branch is merged to the master branch and that you can delete the develop branch now. So delete the develop branch.
- Now go back to your project page and you will see the updates you made and that the branch is deleted.
- Update your local computer using  
`$ git checkout master`  
`$ git pull`



The screenshot shows the GitHub 'Create pull request' interface. At the top, it says 'Create CONTRIBUTING.md'. Below this are tabs for 'Write' and 'Preview'. The 'Write' tab is active, showing a text area with the placeholder text 'Let's add a contributing file so we can work better, together!'. To the right of the text area, there are links for 'Parsed as Markdown' and 'Edit in fullscreen'. Below the text area, there is a note: 'Attach images by dragging & dropping, selecting them, or pasting from the clipboard.' On the right side of the interface, there is a green icon of two branches merging, followed by the text '✓ Able to merge. These branches can be automatically merged.' At the bottom right, there is a green button labeled 'Create pull request'.



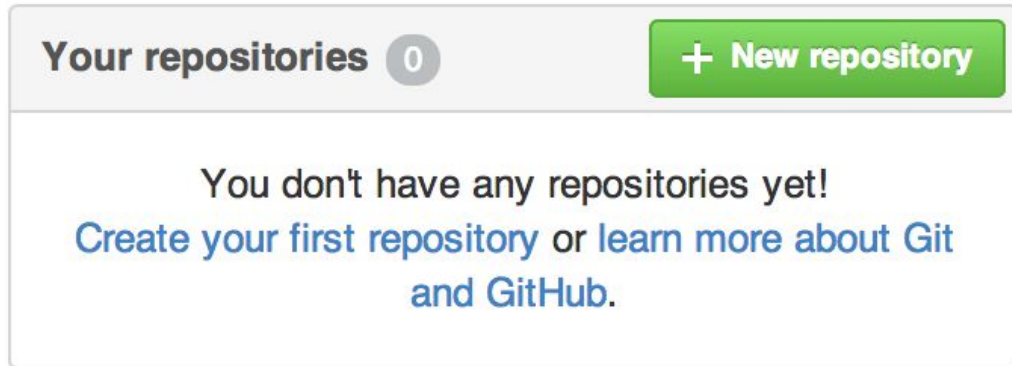
## Exercise 8: Creating a Pull Request to Collaborate

- Make a new file in the **develop2** branch on GitHub.
- Click on **Compare and Pull Request** for the **develop2** branch. It sends you to a discussion page, where you can enter a title and description. It's important to provide useful information and a rationale for why you're making this **Pull Request** in the first place.
- Change the base fork to the repository you forked.
- Click on **Create pull request** and then owner of the project you forked will get a notification that someone is suggesting a change and a link to a page that has all of the information.
- If the owner approves the changes, they can **merge the pull request** with their original repository.



## Maintaining a Project in GitHub - Exercise 9: Creating a New Repository

- Now that we're comfortable contributing to a project, let's look at the other side: creating, maintaining and administering your own project.
- Click the “**New repository**” button on the right-hand side of the dashboard, or from the + button in the top toolbar next to your username as seen in the “**New repository**” dropdown.





# Maintaining a Project in GitHub - Creating a New Repository

- Provide a project name and select “Initialize this repository with a README”. Now you have a new repository on GitHub, named `<user>/<project_name>`.
- Now that your project is hosted on GitHub, you can give the URL to anyone you want to share your project with.
- Every project on GitHub is accessible over HTTPS as `https://github.com/<user>/<project_name>` and over SSH as `git@github.com:<user>/<project_name>`.
- Git can fetch from and push to the HTTPS and SSH URLs, but they are access-controlled based on the credentials of the user connecting to them.



# Maintaining a Project in GitHub - Exercise 10: Adding Collaborators

- If you're working with other people who you want to give commit access to, you need to add them as “**Collaborators**”.
- Doing so will give them “**push**” access, which means they have both read and write access to the project.
- Click the “**Settings**” link at the bottom of the right-hand sidebar.

🔔 Issues

7

🔗 Pull Requests

3

📖 Wiki

📈 Pulse

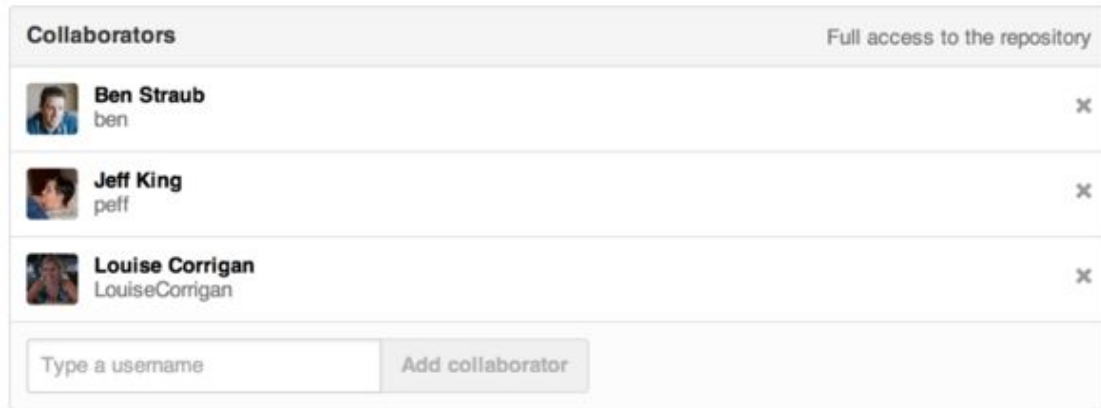
📊 Graphs

⚙️ Settings



# Maintaining a Project in GitHub - Exercise 10: Adding Collaborators

- Then select “**Collaborators**” from the menu on the left-hand side. Then, just type a username into the box, and click “**Add collaborator.**” If you need to revoke access, just click the “**X**” on the right-hand side of their row. You can add each other in this workshop to try it out.





# Maintaining a Project in GitHub - Managing Pull Requests

- Pull Requests can either come from a branch of someone else's fork of your repository or they can come from another branch in the same repository.
- The only difference is that the ones in a fork are often from people where you can't push to their branch and they can't push to yours, whereas with internal Pull Requests generally both parties have push access to the branch.
- Internal Pull Requests would be useful to make sure the master branch has only finished and approved work when working in a team on the same repository.



# Maintaining a Project in GitHub - Managing Pull Requests

- Say that someone comes along and makes a change to your code and sends you a **Pull Request**. You should get an email notifying you about the new **Pull Request**.
- You can have a conversation with the person who opened the **Pull Request**. You can comment on specific lines of code, comment on whole commits or comment on the entire Pull Request itself.
- Every time someone else comments on the **Pull Request** you will continue to get email notifications so you know there is activity happening.
- They will each have a link to the **Pull Request** where the activity is happening and you can also directly respond to the email to comment on the **Pull Request** thread.

# At Home Exercise: Managing Pull Requests

- Once the code is in a place you like and want to merge it in, you can hit the “Merge pull request” button on the GitHub site.
- If you decide you don’t want to merge it, you can also just close the Pull Request and the person who opened it will be notified.



**This pull request can be automatically merged.**

You can also merge branches on the [command line](#).



**Merge pull request**

## Merging via command line

If you do not want to use the merge button or an automatic merge cannot be performed, you can perform a manual merge on the command line.

**HTTP** **Git** **Patch** <https://github.com/schacon/fade.git>



**Step 1:** From your project repository, check out a new branch and test the changes.

```
git checkout -b schacon-patch-1 master
git pull https://github.com/schacon/fade.git patch-1
```



**Step 2:** Merge the changes and update on GitHub.

```
git checkout master
git merge --no-ff schacon-patch-1
git push origin master
```



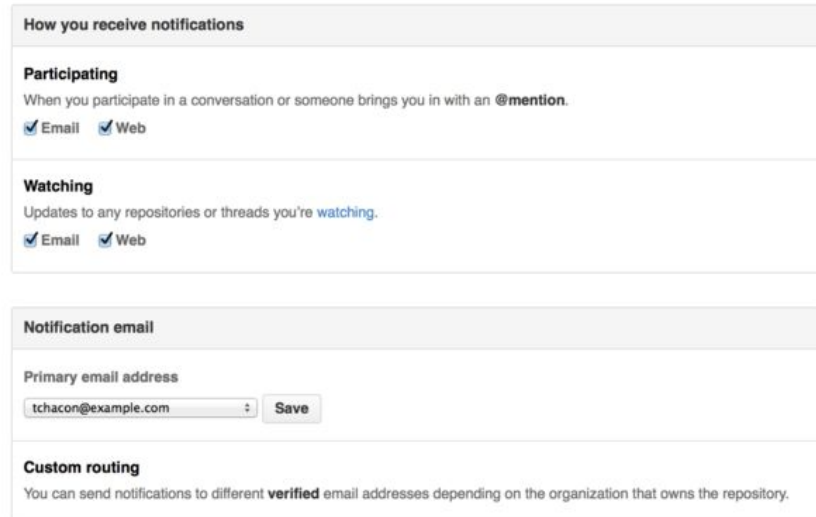
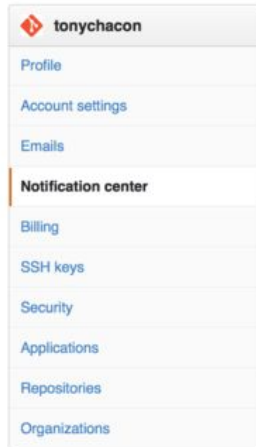


# Maintaining a Project in GitHub - Mentions

- In any comment you can start typing a **@ character** and it will **begin to autocomplete** with the names and usernames of people who are **collaborators** or **contributors** in the project.
- Once you post a comment with a user mention, that user will be notified. This can be a really effective way of pulling people into conversations.
- Very often in **Pull Requests** on GitHub people will pull in other people on their teams or in their company to review an **Issue** or **Pull Request**.
- If someone gets mentioned on a **Pull Request** or **Issue**, they will be “**subscribed**” to it and will continue getting notifications any time some activity occurs on it.
- If you no longer wish to receive notifications, there is an “**Unsubscribe**” button on the page you can click to stop receiving updates on it.

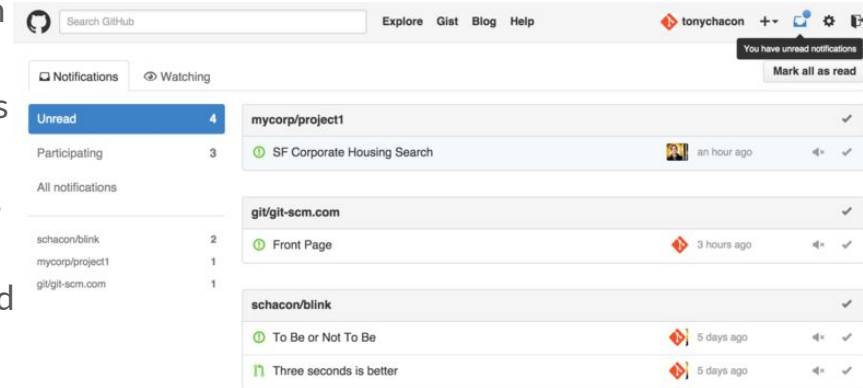
# Maintaining a Project in GitHub - Notifications

- **Notifications** are a specific way that GitHub tries to get in touch with you when events happen.
- Here are a few different ways you can configure them. If you go to the “**Notification center**” tab from the settings page, you can see some of the options you have.
- You can choose to get notifications over “**Email**” and over “**Web**”.



# Maintaining a Project in GitHub - Notifications

- Web notifications only exist on GitHub and you can only check them on GitHub.
- You will see a **small blue dot** over your notifications icon at the top of your screen.
- If you click on that, you will see a list of all the items you have been notified about, grouped by project.
- You can **filter** by clicking on its name in the left hand sidebar. You can also **acknowledge** the notification by clicking the checkmark icon, or acknowledge all of the notifications in a project by clicking the checkmark at the top of the group.
- There is also a **mute button** that you can click to not receive any further notifications on that item.





# Maintaining a Project in GitHub - Notifications

- Email notifications are the other way you can handle notifications through GitHub.
- If you have this turned on, you will get emails for each notification. The emails will also be threaded properly, which is nice if you're using a threading email client.
- There is also a fair amount of metadata embedded in the headers of the emails that GitHub sends you, which can be really helpful for setting up custom filters and rules.
- If you have both email and web notifications enabled and you read the email version of the notification, the web version will be marked as read as well if you have images allowed in your mail client.





# Maintaining a Project in GitHub - README File

- The **README** file in your repository is a file which can be of nearly any format that GitHub recognizes as prose. For example, it could be **README**, **README.md**, **README.asciidoc**, etc.
- If GitHub sees a **README** file in your source, it will render it on the landing page of the project.



# Maintaining a Project in GitHub - README File

- Many teams use the **README** file to hold all the relevant project information for someone who might be new to the repository or project. This generally includes things like:
  - What the project is for
  - How to configure and install it
  - An example of how to use it or get it running
  - The license that the project is offered under
  - How to contribute to it
- Since GitHub will render this file, you can embed images or links in it for added ease of understanding.

# Maintaining a Project in GitHub - Contributing File

- If you have a file named **CONTRIBUTING** with any file extension, GitHub will show a note to review the contributing guidelines when anyone starts opening a Pull Request.
- The idea here is that you can specify specific things you want or don't want in a Pull Request sent to your project.


Please review the [guidelines for contributing](#) to this repository.

Title

Write Preview

Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.



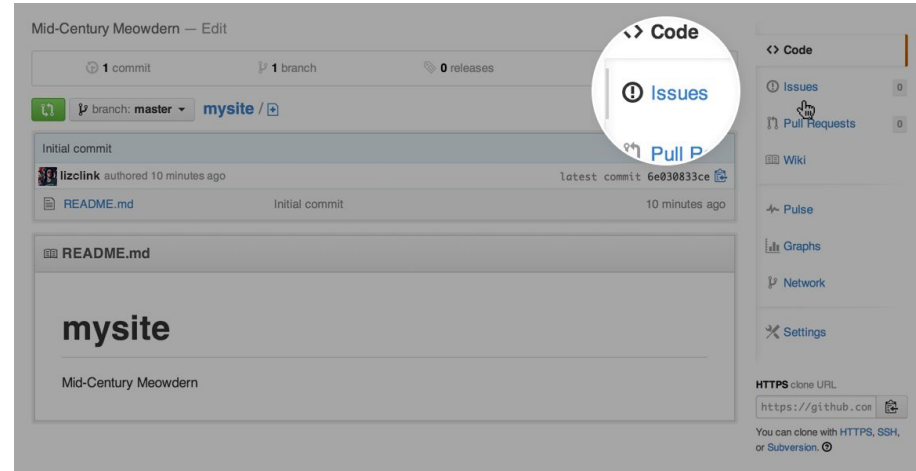
**We can't automatically merge these branches.**  
Don't worry, you can still create the pull request.

[Create pull request](#)

MDX Parsed as Markdown Edit in fullscreen

# Team Collaboration Tools - Issues

- In GitHub, it is helpful to use **Issues** in the following ways:
  - Track Bugs: Things that are obviously broken and need fixes
  - Features List: New ideas to implement
  - To do list: A checklist of items to complete
- They're kind of like email—except they can be shared and discussed with the rest of your team.



# Team Collaboration Tools - Issues

- A typical issue on GitHub looks a bit like this:

The screenshot displays a GitHub issue page. At the top, the issue title is "The no-conflict mode should be the default behaviour #12395". To the right of the title are "Edit" and "New issue" buttons. Below the title, a green "Open" button is followed by the text "thewebdreamer opened this issue 3 days ago · 10 comments". The main content area shows three comments. The first comment, by thewebdreamer, asks why a Bootstrap client needs to implement no-conflict mode. The second comment, by cvrebert, states that no-conflict is the norm for jQuery plugins. The third comment, by thewebdreamer, responds that it is true but asks if there could be a clash with other jQuery plugins. On the right side, there are sections for "Labels" (with a "js" label), "Milestone" (set to "No milestone"), "Assignee" (set to "No one assigned"), and "Notifications" (with a "Subscribe" button). At the bottom right, it says "3 participants" and shows avatars for thewebdreamer, cvrebert, and another user.

◀ The no-conflict mode should be the default behaviour #12395

**Edit** **New issue**

**Open** thewebdreamer opened this issue 3 days ago · 10 comments

**thewebdreamer** commented 3 days ago

The no-conflict mode should be the default behaviour. Why would a Bootstrap client need to implement this?

**cvrebert** commented 3 days ago

I believe no-conflict-is-not-the-default is the norm for jQuery plugins?

**thewebdreamer** commented 3 days ago

It is true that it is the norm for jQuery plugins.

Couldn't there be a clash with other jQuery plugins with the current implementation of Bootstrap though?

**Labels**

js

**Milestone**

No milestone

**Assignee**

No one assigned

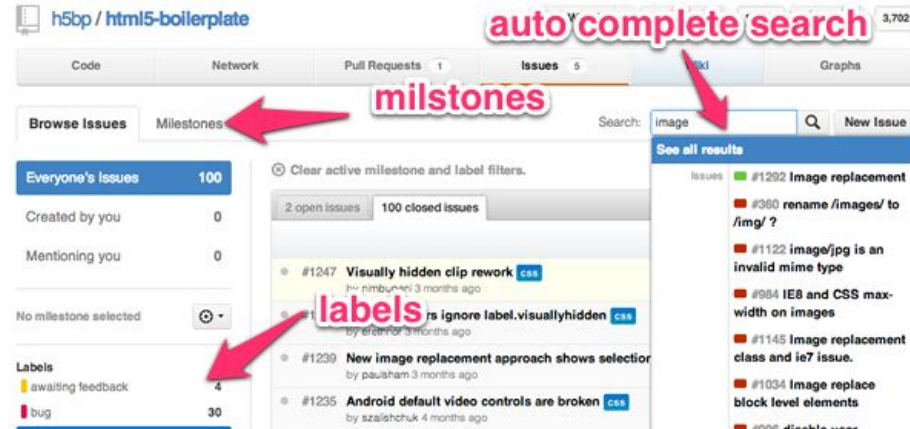
**Notifications**

**Subscribe**

3 participants

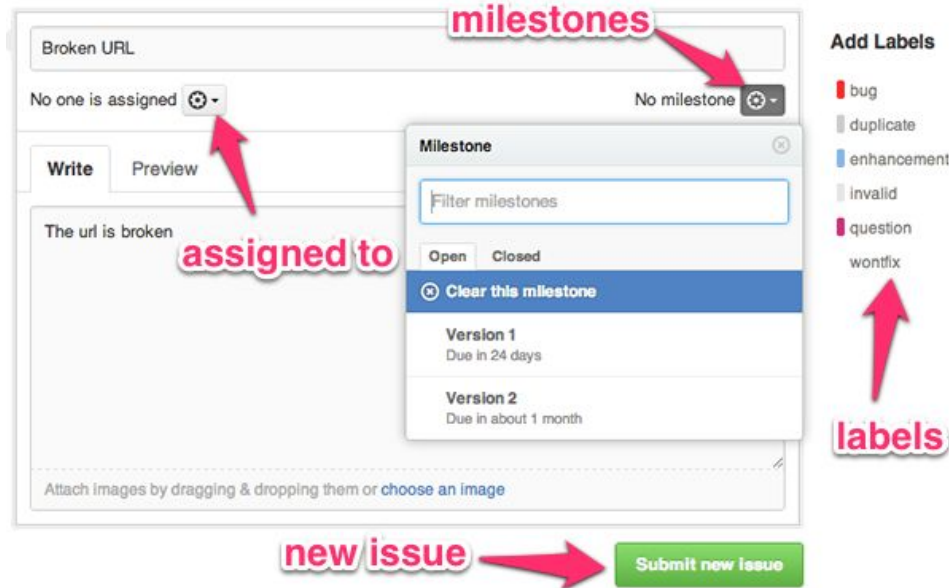
# Team Collaboration Tools - Issues

- A **title** and **description** describe what the issue is all about.
- Color-coded **labels** help you categorize and filter your issues.
- **Search:** Search issues using created/updated dates, labels, authors, comment counts, by repository owner, and more.



# Team Collaboration Tools - Issues

- One **assignee** is responsible for working on the issue at any given time.
- **Milestones** are groups of issues that correspond to a project, feature, or time period. This is useful for associating issues with specific features or project phases (e.g. Weekly Sprint 9/5-9/16 or Shipping 1.0).
- **Comments** allow anyone with access to the repository to provide feedback on the issue.



# Team Collaboration Tools - Issues

- Each repository has a section called **Pulse** — **Pulse** is a snapshot of everything that's happened in the repository in the past week (or day, or past 3 months, etc).
- It's a great way to catch up with repositories when you've been away and don't want the notifications.

August 01 2013 - August 08 2013

Period: 1 week ▾

## Overview



65 active pull requests



257 active issues

49

merged pull requests

16

proposed pull requests

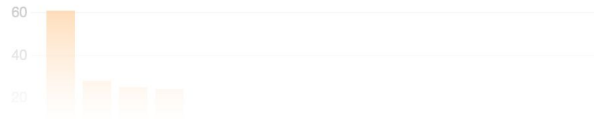
230

closed issues

27

new issues

54 authors have pushed 253 commits to all branches, excluding merges. On 3.0.0-wip, 64 files have changed and







# Team Collaboration Tools - Analytics

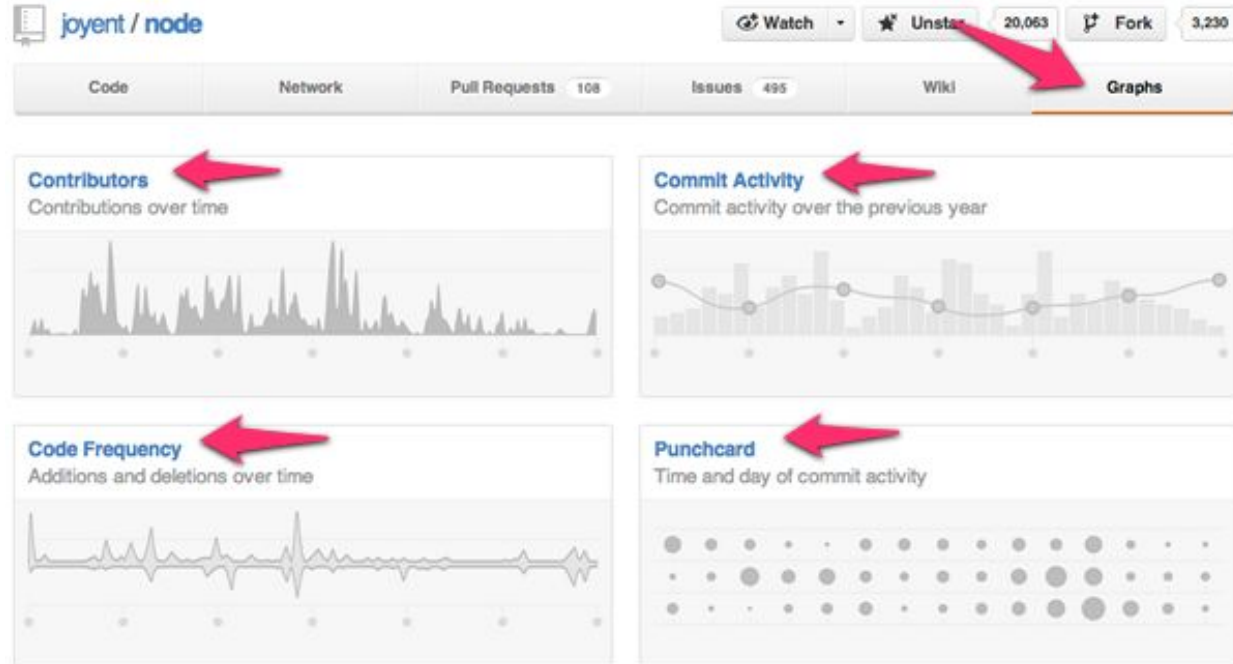
- There are two tools that give insight into a repository - **Graphs** and **Network**.
- **GitHub Graphs** provides an insight into the collaborators and commits behind each code repository.
- **GitHub Network** provides a visualization on each contributors and their commits across all forked repositories. These analytics and graphs become very powerful, especially when working in teams.



# Team Collaboration Tools - Graphs

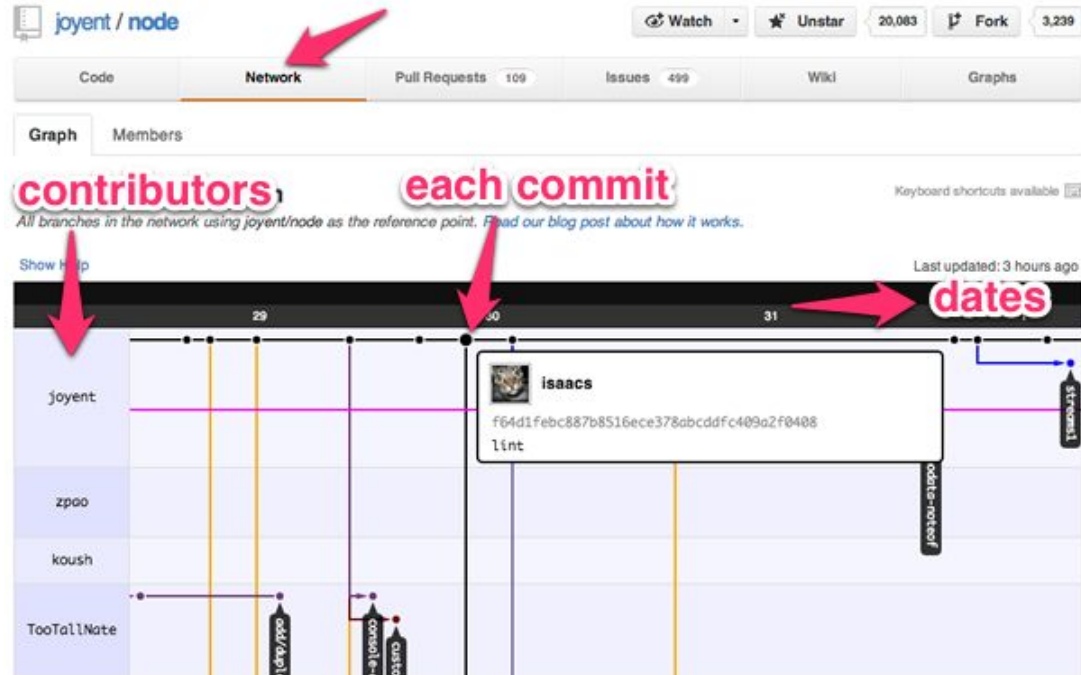
- Graphs provide detailed analytics such as:
  - **Contributors:** Who were the contributors? And how many lines of code did they add or delete?
  - **Commit Activity:** Which weeks did the commits take place in the past year?
  - **Code Frequency:** How many lines of code were committed throughout the entire lifecycle of the project?
  - **Punchcard:** During which times of the day do the commits usually take place?

# Team Collaboration Tools - Graphs



# Team Collaboration Tools - Network

- **GitHub Network** is a powerful tool that lets you see every contributor's commits and how they are related to one another.
- When we look at the visualizer in its entirety, we see every commit on every branch of every repository that belongs to a network.



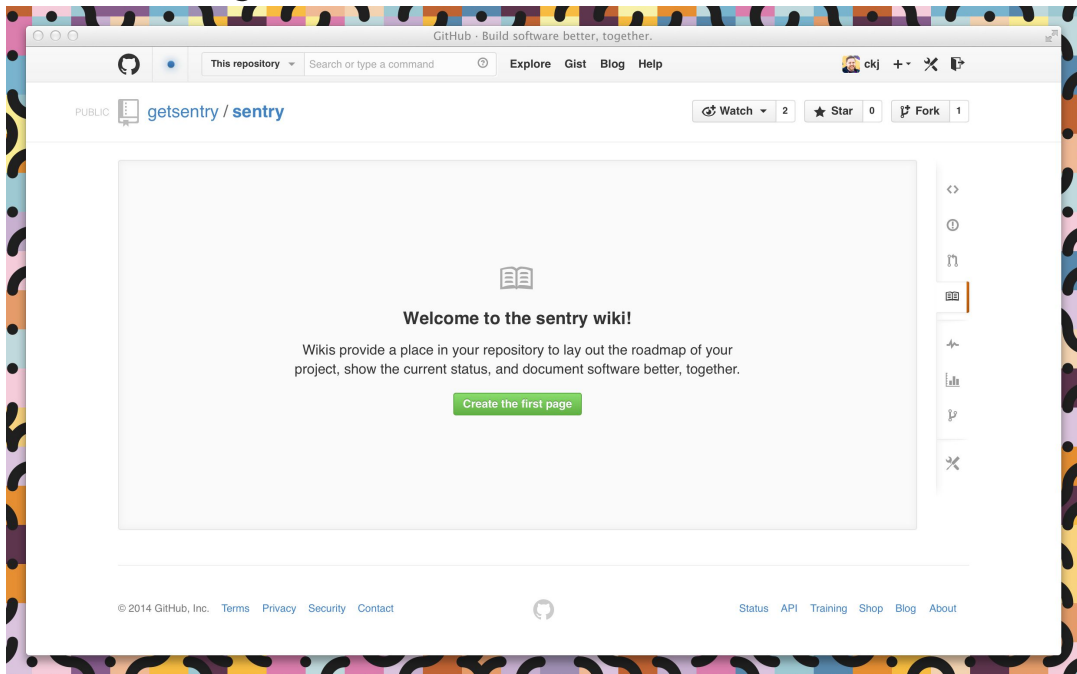


# Team Collaboration Tools - Project Management

- While GitHub Issues have project management capabilities with **Issues** and **Milestones**, some teams might prefer another tool because of other features or existing workflow.
- GitHub can link with two other popular project management tools - **Trello** and **Pivotal Tracker**.
- With GitHub, we can automate updating tasks with commits, issues and many other activities.
- This automation helps in not only saving time, but also increases accuracy in updates for any team.

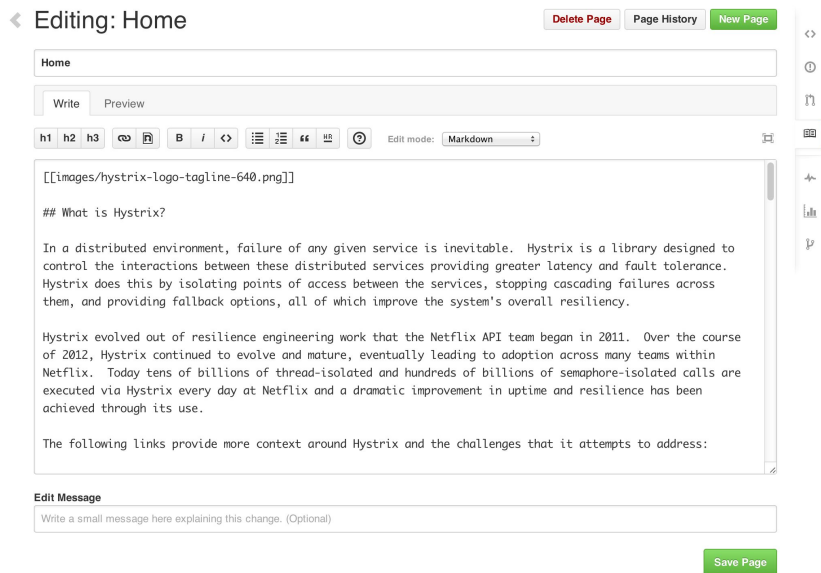
# Documenting your Projects on GitHub

- Every repository on GitHub comes with a wiki.
- After you've created a repository, you can set up the included wiki.
- Start your wiki by clicking the wiki button and creating your first page.



# Documenting your Projects on GitHub

- Wiki content is designed to be easily editable.
- You can add or change content on any wiki page by clicking the **Edit** button which opens up the wiki editor.
- Using the drop-down menu in the editor, you can select the format of your wiki, and then use wiki toolbar to create and include content on a page.
- Wikis also give you the option of including a custom footer where you can list things like contact details or license information for your project.



The screenshot shows the GitHub Wiki editor interface. At the top, there's a header bar with 'Editing: Home' on the left and three buttons: 'Delete Page' (red), 'Page History' (grey), and 'New Page' (green). Below this is a text input field containing 'Home'. Underneath the input field are two tabs: 'Write' (active) and 'Preview'. A rich text toolbar is visible with various icons for text formatting (bold, italic, link, unlink, list, indent, outdent, quote, code), alignment, and other functions. The 'Edit mode' dropdown is set to 'Markdown'. The main editing area contains the following text:

```
[[images/hystrix-logo-tagline-640.png]]

## What is Hystrix?

In a distributed environment, failure of any given service is inevitable. Hystrix is a library designed to control the interactions between these distributed services providing greater latency and fault tolerance. Hystrix does this by isolating points of access between the services, stopping cascading failures across them, and providing fallback options, all of which improve the system's overall resiliency.

Hystrix evolved out of resilience engineering work that the Netflix API team began in 2011. Over the course of 2012, Hystrix continued to evolve and mature, eventually leading to adoption across many teams within Netflix. Today tens of billions of thread-isolated and hundreds of billions of semaphore-isolated calls are executed via Hystrix every day at Netflix and a dramatic improvement in uptime and resilience has been achieved through its use.

The following links provide more context around Hystrix and the challenges that it attempts to address:
```

At the bottom, there's an 'Edit Message' section with a text input field containing the placeholder 'Write a small message here explaining this change. (Optional)'. A green 'Save Page' button is located at the bottom right corner.

# Documenting your Projects on GitHub

- GitHub keeps track of changes made to each page in your wiki.
- Below a page title, you can see who made the most recent edits, in addition to the number of commits made to the page.
- Clicking on this information will take you to the full page history where you can compare revisions or see a detailed list of edits over time.

◀ Editing: Home

Delete Page

Page History

New Page

Home

Write

Preview

h1

h2

h3

🔗

📁

B

i

<>

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

☰

[[images/hystrix-logo-tagline-640.png]]

## What is Hystrix?

In a distributed environment, failure of any given service is inevitable. Hystrix is a library designed to control the interactions between these distributed services providing greater latency and fault tolerance. Hystrix does this by isolating points of access between the services, stopping cascading failures across them, and providing fallback options, all of which improve the system's overall resiliency.

Hystrix evolved out of resilience engineering work that the Netflix API team began in 2011. Over the course of 2012, Hystrix continued to evolve and mature, eventually leading to adoption across many teams within Netflix. Today tens of billions of thread-isolated and hundreds of billions of semaphore-isolated calls are executed via Hystrix every day at Netflix and a dramatic improvement in uptime and resilience has been achieved through its use.

The following links provide more context around Hystrix and the challenges that it attempts to address:

Edit Message

Write a small message here explaining this change. (Optional)

Save Page



# Documenting your Projects on GitHub

- You can add additional pages to your wiki by selecting **New Page** in the upper right corner.
- By default, each page you create is included automatically in your wiki's sidebar and listed in alphabetical order.
- You can also add a custom sidebar to your wiki by clicking the **Add custom sidebar** link.
- Custom sidebar content can include text, images, and links.

A JavaScript library for manipulating documents based on data. **D3.js** helps you bring data to the browser using **HTML** and **CSS**. D3.js's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization techniques and a data-driven approach to DOM manipulation.

**D3.js**

Library  
Links

Overflow  
Group

(русскоязычная версия)

Support

▼ Pages (66)

Find a Page...

3.0

3.1

API Reference

API Reference (русскоязычная версия)

API参考

Arrays

Behaviors

Bundle Layout

Chord Layout

Cluster Layout

CN Home

Colors

Core

CSV

Drag Behavior

Show 51 more pages...



# Being Social

- With more and more people joining GitHub and adding projects every day, keeping up with all of them can be difficult.
- However, this can be fun and easy by following users or watching repositories.

## Be Social - Follow a Friend

- One of the great features on GitHub is the ability to see what other people are working on and who they are connecting with.
- Once you are on someone's profile, click the “**follow**” button.
- When you follow someone, you'll get notifications about their GitHub activity.
- Following friends helps you find new projects and new people that you may share interests with.
- You can see what your friends are interested in by looking at the **Explore** page  
<https://github.com/explore>





## Being Social - Watch a Project

- At some point you may want to stay up-to-date with a specific project. This is similar to following a person, except the focus is narrowed to only events on that project.
- When you watch a repository, you get notifications for any new pull requests and issues that are created, including those not mentioning you.
- You can choose to have notifications for this project sent via email or viewable on the web by configuring your settings.
- Once you are on the project page, you can click on the “watch” button at the top of the page.





## Be Social - Star a Repository

- Starring a repository allows you to keep track of projects that you find interesting, even if you aren't associated with the project.
- When you star a repository, you're actually performing two distinct actions:
  - Creating a bookmark for easier access
  - Showing appreciation to the repository maintainer for their work
- Notifications are not affected when a repository is starred. You won't receive any information about a repository you've starred, unless you're also watching the repository.



# Additional Resources

All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](https://creativecommons.org/licenses/by-nc-sa/3.0/)

## Free Tutorials

- Pro Git book: <https://git-scm.com/>
- GitHub Guides: <https://guides.github.com/activities/hello-world/>
- GitHub On Demand Training: <https://services.github.com/on-demand/>
- Command Line Tutorial: <https://www.davidbaumgold.com/tutorials/command-line/>
- Collaboration tools: <https://code.tutsplus.com/articles/team-collaboration-with-github--net-29876>

## Documentation

- git Documentation: <https://git-scm.com/documentation>
- GitHub help: <https://help.github.com/>