

PROJECT 05

Introduction to collaborative coding with GitHub

1. Install Git	2
2. Do the Tutorial (Optional, but recommended for anyone new to GitHub)	2
3. Setup your team's repo via GitHub Organizations	2
A. Create New Organization	2
B. Create a new repo within your organization	3
C. Set up reviewing and merging rules	8
4. Setup your project repo on your local computer	9
A. Clone the repository	9
C. Create a new branch	9
D. Edit a file, commit a change, and push the branch	9
5. Create a Pull Request	10
6. Review and Merge Pull Requests	11
7. Sync your organization's repo with your local computer	13
A. Merge new changes into a branch you're already working on	13
B. Create a brand new branch from master	13
8. Enable GitHub Pages	14
9. Create a branch to submit [What to Turn In]	15
SUMMARY. Rules of the Suggested Workflow	15

1. Install Git

Install git on your computer (if you haven't already):

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

2. Do the Tutorial (Optional, but recommended for anyone new to GitHub)

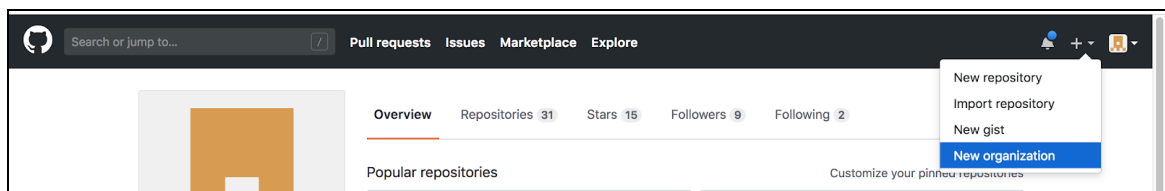
1. Register for GitHub if you haven't already
2. Watch [this overview video](#)
3. Enroll in [the GitHub Tutorial](#) (PDF version [here](#))
4. Refer to the [Git Cheat Sheet](#) if you have questions about specific commands

3. Setup your team's repo via GitHub Organizations

A. Create New Organization

One member of your team needs to create a GitHub organization (name it anything you want). In this example, we named ours EECS 330, but you should name yours something that reflects your team. To do this:

1. Log into GitHub
2. In the upper right-hand corner, click the plus button, and select "New organization" from the dropdown menu.



3. Once you've created the organization, add all of your team members to the organization. Make everyone an owner (to avoid any power differentials across the team).

Sign up your team

Step 1: Set up the organization

Step 2: Invite members

Step 3: Organization details

Create an organization account

Organization name *

Billing email *

We'll send receipts to this inbox.

Choose your plan

Team For Open Source

Team

Enterprise


Organization accounts allow your team to plan, build, review, and ship software — all while tracking bugs and discussing ideas.

Please also add your TA to your repository (see the list of GitHub handles below):

Section	Name	Handle
61	Shu	leaf715
62	Vishal	weeshal
64	Abizar	abizzaar
65	Anna	anna-deng
66	Richard	Rshang97
68	Cori	cpitiger
69	Sanfeng	wangsanfeng1998
70	David	MagicWilliams
72	Gabriel	gcanco
MSAI	Armaan	ArmaanDhingra

B. Create a new repo within your organization

Within your organization, one person needs to create a repository that *reflects the name of your project* (though you can always rename it later). Don't name it project5! This is just an example.

 eeecs-330

Repositories 0

People 2


Teams 0


Projects 0

Settings

This organization has no repositories.
[Create a new repository](#)

People 2 >


 christym

 vanwars
Sarah

[Invite someone](#)

Owner

Repository name *


 eeecs-330


 /

project5 ✓

Great repository names are short and memorable. Need inspiration? How about **bookish-giggle**.

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None** ⓘ

[Create repository](#)

Give access to the people you work with
You should give access to the collaborators and teams you need to work with.
[Add teams and collaborators](#)

Quick setup — if you've done this kind of thing before
[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/eecs-330/project5.git>
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

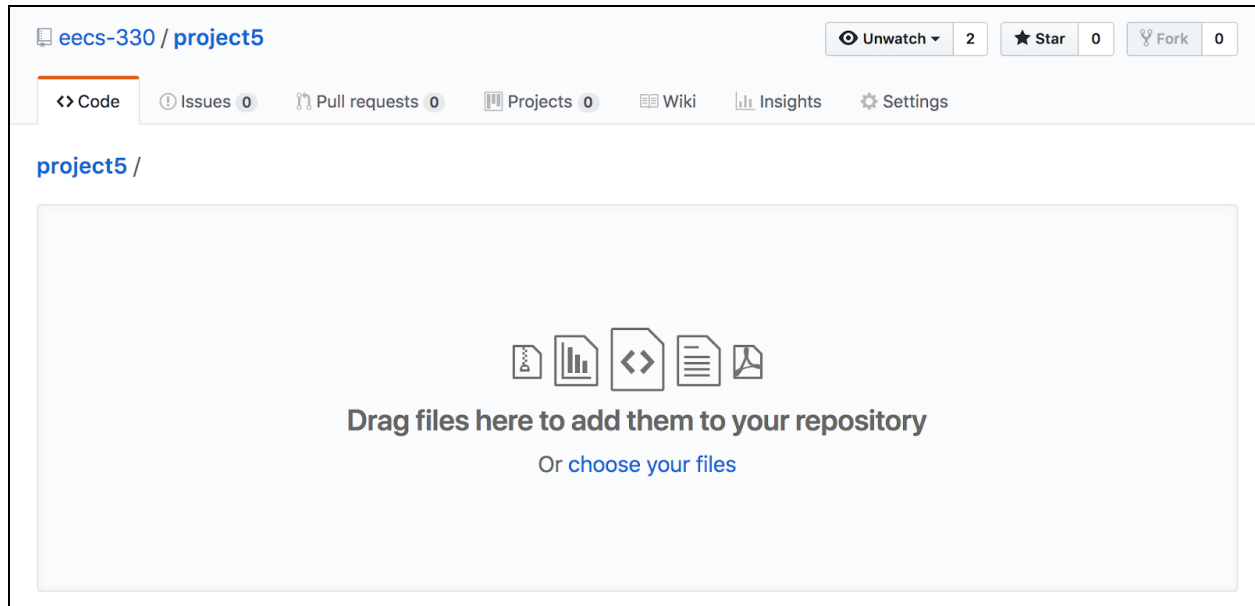
```
echo "# project5" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/eecs-330/project5.git
git push -u origin master
```

UPLOAD P5 SAMPLE FILES
BY CLICKING "UPLOADING
AN EXISTING FILE" LINK

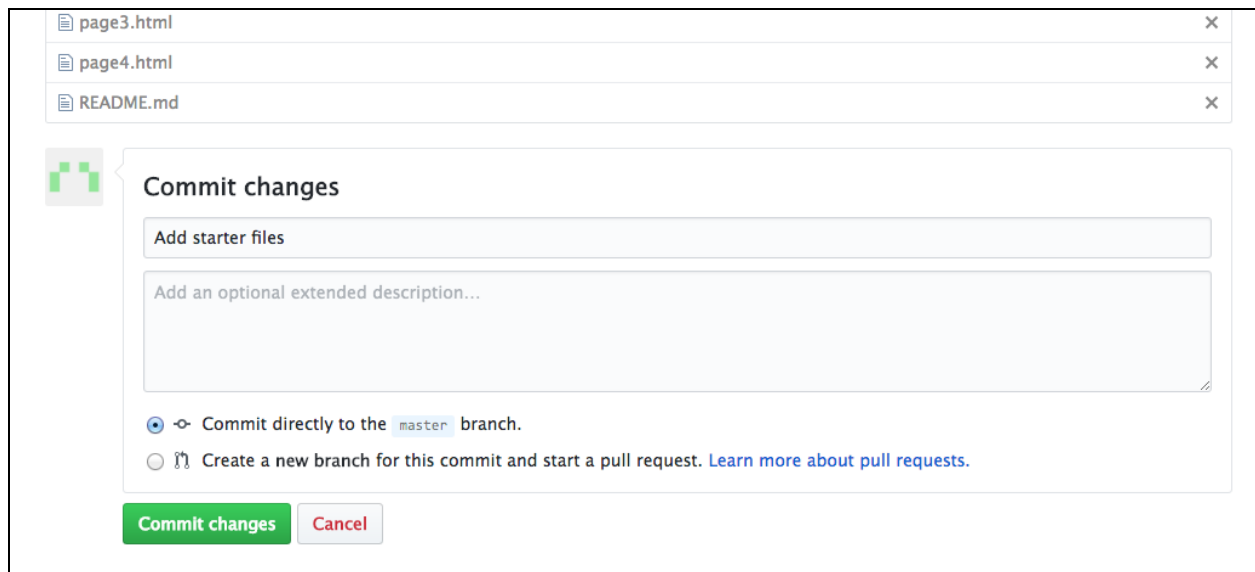
Unzip all of the files from [project 5 files.zip](#) and drag them into the "dropzone." When you're done, don't forget to scroll down and press the green commit button. The following files should be included:

```
.
├── css
│   ├── extras.css
│   └── w3.css
├── images
│   ├── avatar_hat.jpg
│   ├── map.jpg
│   ├── p1.jpg
│   ├── p2.jpg
│   ├── p3.jpg
│   ├── p4.jpg
│   ├── p5.jpg
│   ├── p6.jpg
│   ├── p7.jpg
│   ├── p8.jpg
│   ├── parallax1.jpg
│   ├── parallax2.jpg
│   └── parallax3.jpg
├── index.html
├── page2.html
├── page3.html
└── page4.html
```

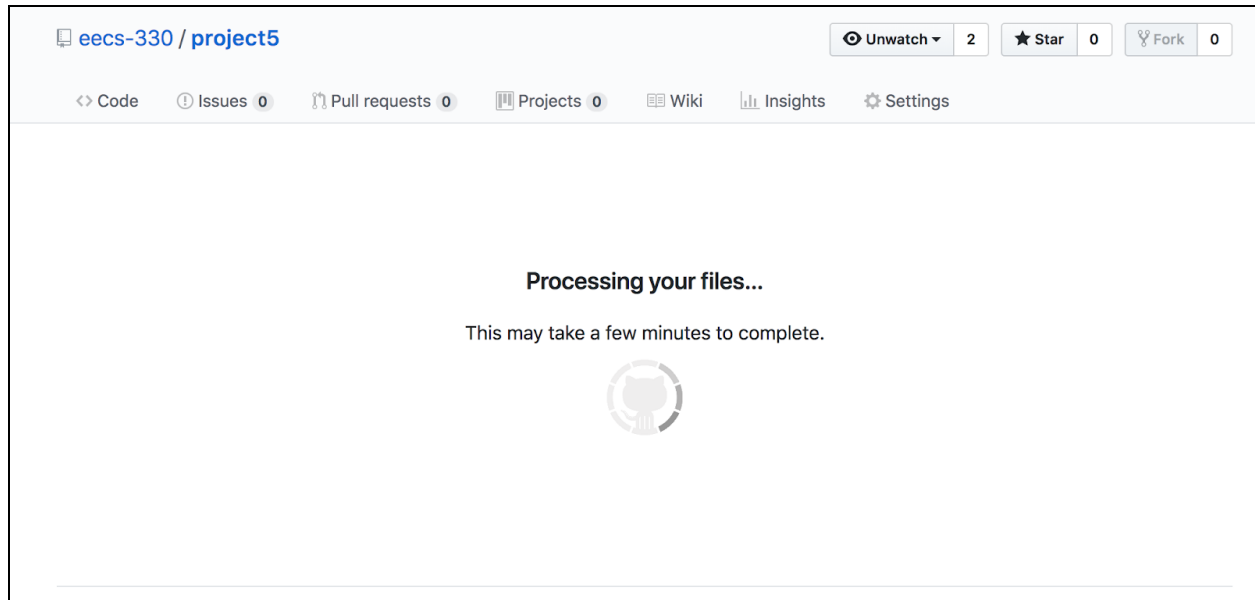
NOTE: You are welcome to use your own project files here (instead of the sample files we have provided). The purpose of project 5 is that everyone on your team gets practice editing, committing, merging, and reviewing via git/GitHub. The actual files used are not important



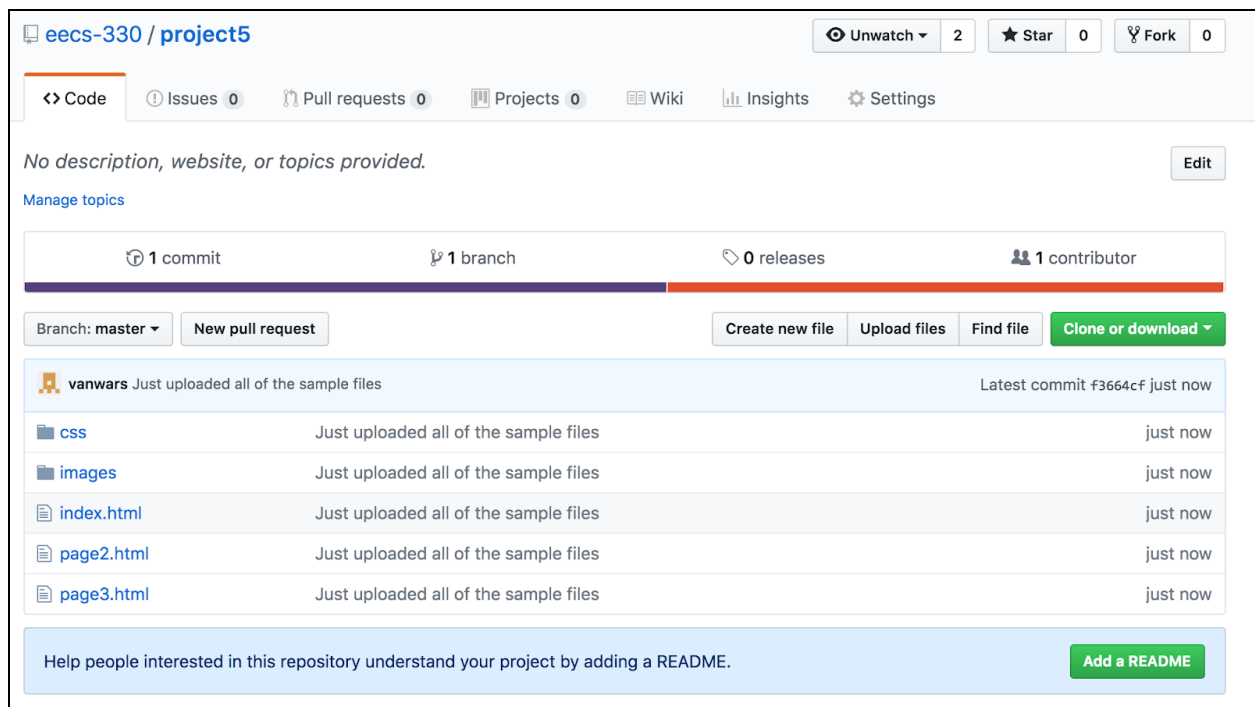
After the files upload, fill out the text box at the bottom of the screen to commit the changes to the master branch of your repository:



Then GitHub will process the files:



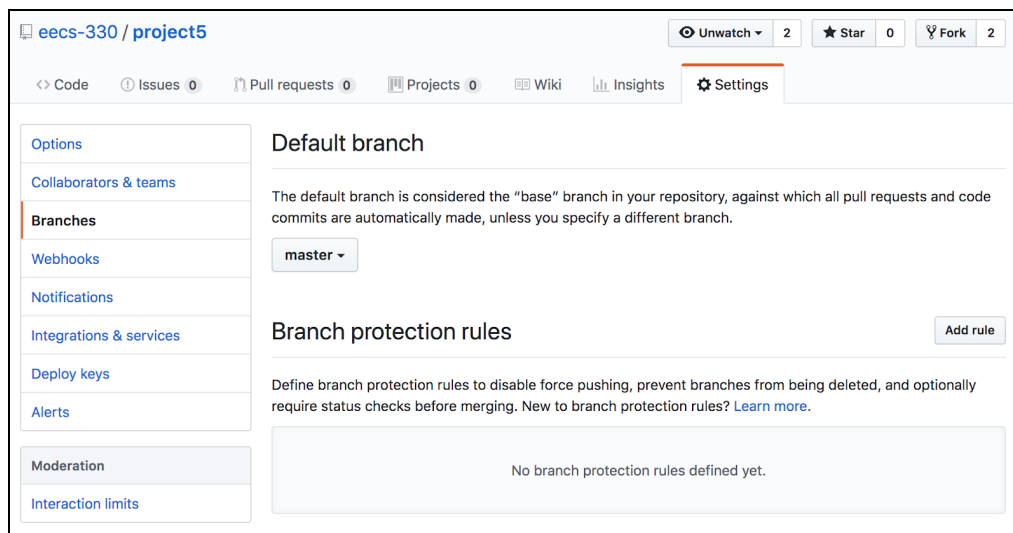
When GitHub is done processing your files, you should see them all listed within the organization's copy of the repository:



C. Set up reviewing and merging rules

A key part of the collaborative software development process is review and revision. Typically users are not allowed to approve their own *pull requests* (PRs) — requests to commit code to the main repository — because review is essential to code quality and team morale. To facilitate reviews, GitHub has some constraints that you can set up to ensure code review and code quality.

Click the settings tab for your repository (towards the upper right-hand side of the screen), and then click the branches tab on the left-hand navigation menu. You are going to add a ‘branch protection rule’ to your master, requiring that at least one person review the pull request (other than the author). Click the “Add rule” button:



Then, type in “master” in the textbox that says “apply rule to” and check the box next to “Require pull request reviews before merging.” Make sure that 1 review is selected. Don’t forget to click the green “Create” button at the bottom of the page.

Options

Collaborators & teams

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Alerts

Moderation

Interaction limits

Branch protection rule

[Apply Rule to master branch](#)

Apply rule to

master

Rule settings

Protect matching branches [Require 1 approving review](#)

Disables force-pushes to all matching branches and prevents them from being deleted.

☒ **Require pull request reviews before merging**

When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

Required approving reviews: 1

☐ **Dismiss stale pull request approvals when new commits are pushed**

New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☐ **Require review from Code Owners**

Require an approved review in pull requests including files with a designated code owner.

☐ **Restrict who can dismiss pull request reviews**

4. Setup your project repo on your local computer

A. Clone the repository

On your own computer, clone the organization's repo on your local machine using the command line.

```
$ git clone https://github.com/name-of-organization/name-of-repo.git
$ cd name-of-repo
```

C. Create a new branch

After you clone your repository, each teammate will need to create a branch that has a unique name (since you'll all be sharing the same repository)

```
$ git branch p5           # creates a new branch (name it something meaningful)
$ git checkout p5         # switches the active branch from master to p5
$ git branch              # allows you to verify that p5 is now the active branch
  master                  # note: asterisk next to p5 indicates that p5 is active
* p5-vanwars
```

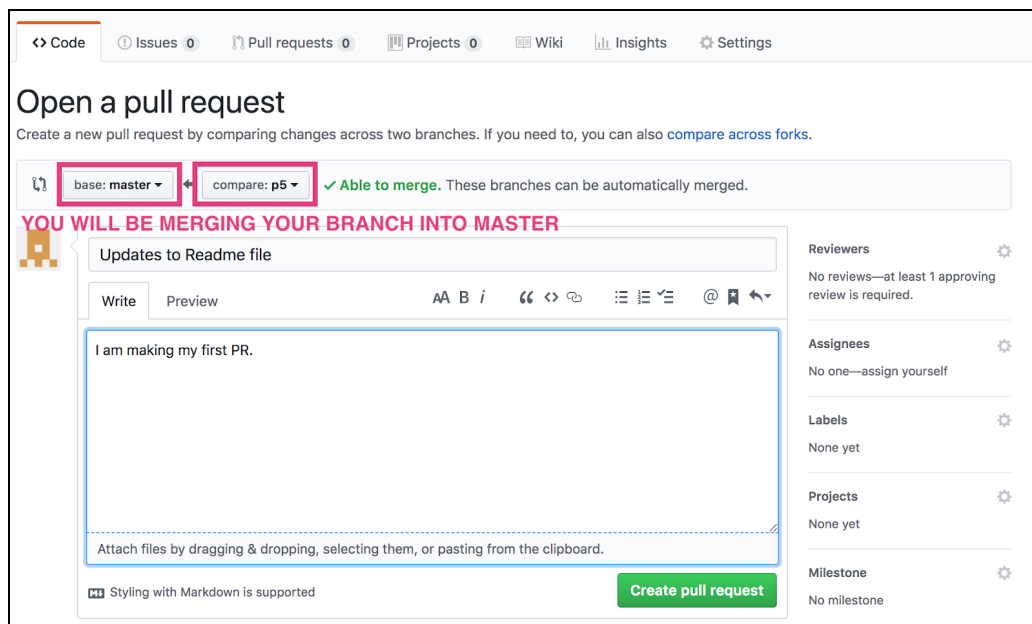
D. Edit a file, commit a change, and push the branch

1. Each teammate should make a small edit to one HTML file (index.html, page1.html, page2.html, page3.html). Coordinate with your teammates to decide who is changing which file.
2. Make edits to your respective HTML files. When you're done, commit the change and push it to GitHub:

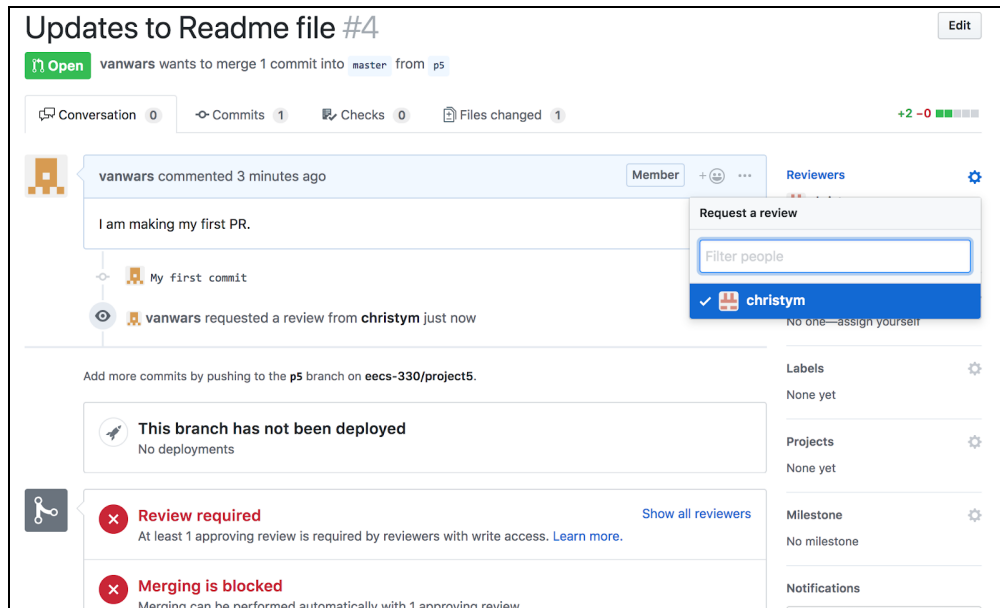
```
# shows which files changed:
$ git status
# committing all changes w/a message:
$ git commit -am 'Made edit to index.html'
# pushing changes + creating new remote branch called p5:
$ git push --set-upstream origin p5
```

5. Create a Pull Request

When you've pushed your new branch to GitHub, navigate to *your* organization's GitHub account in your web browser, and create a Pull Request (PR). A pull request is a way of initiating the formal review process for merging your code into a 'master' copy of the codebase. You will be asking to merge *your branch* with your organization's *master branch*. Give your pull request a name and a description and then click the green "Create pull request" button.

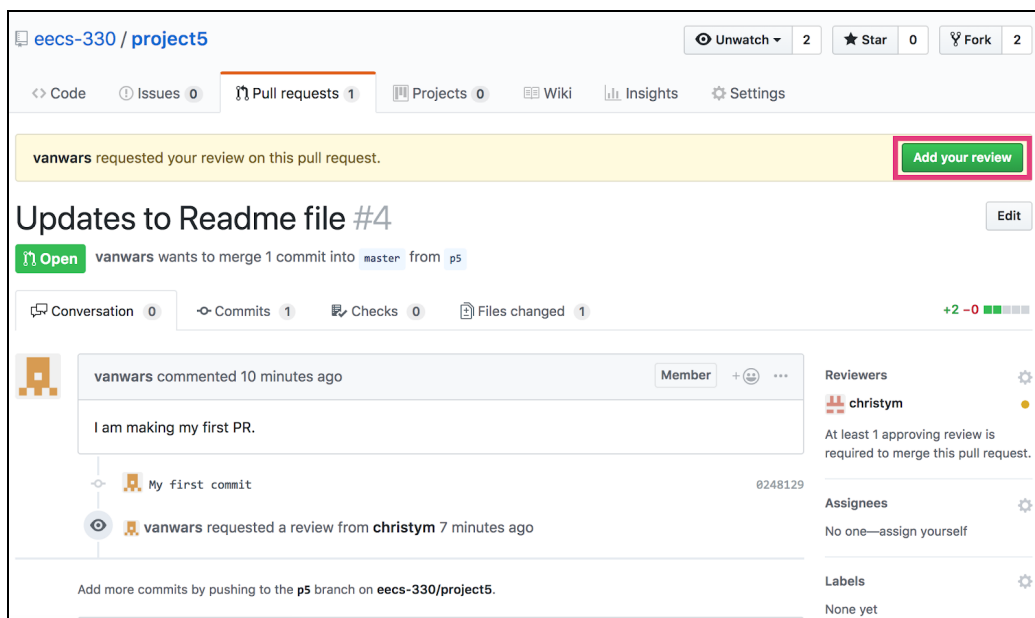


Once you have created the pull request, your screen should update such that you will see that merging is blocked because you need one reviewer to approve your PR. On the right-hand side menu, you will click the "Reviewers" link and assign a reviewer to review your PR. Note: you won't be able to review your own PR, only someone else's.

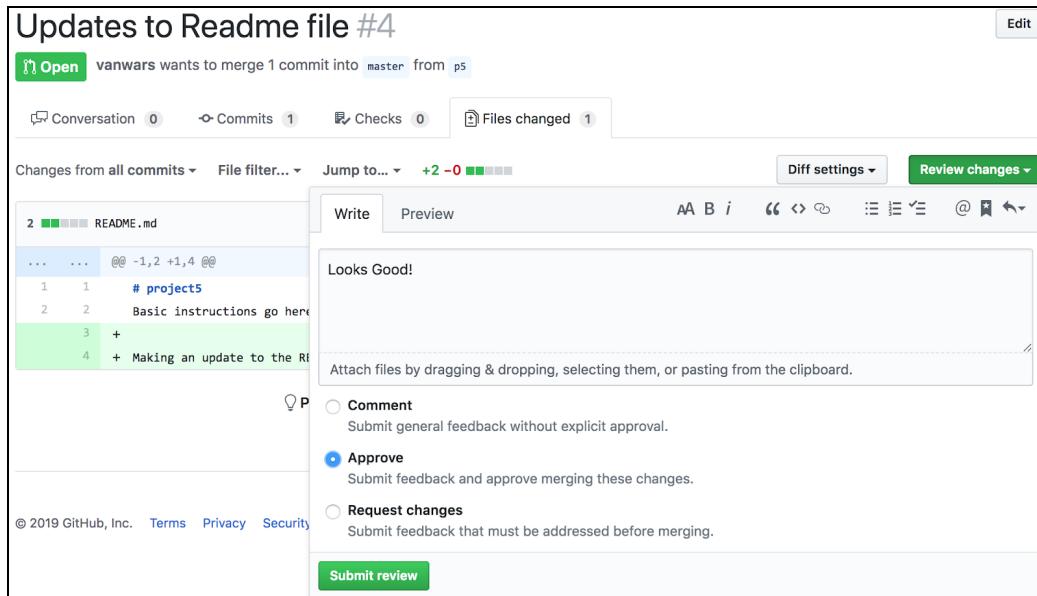


6. Review and Merge Pull Requests

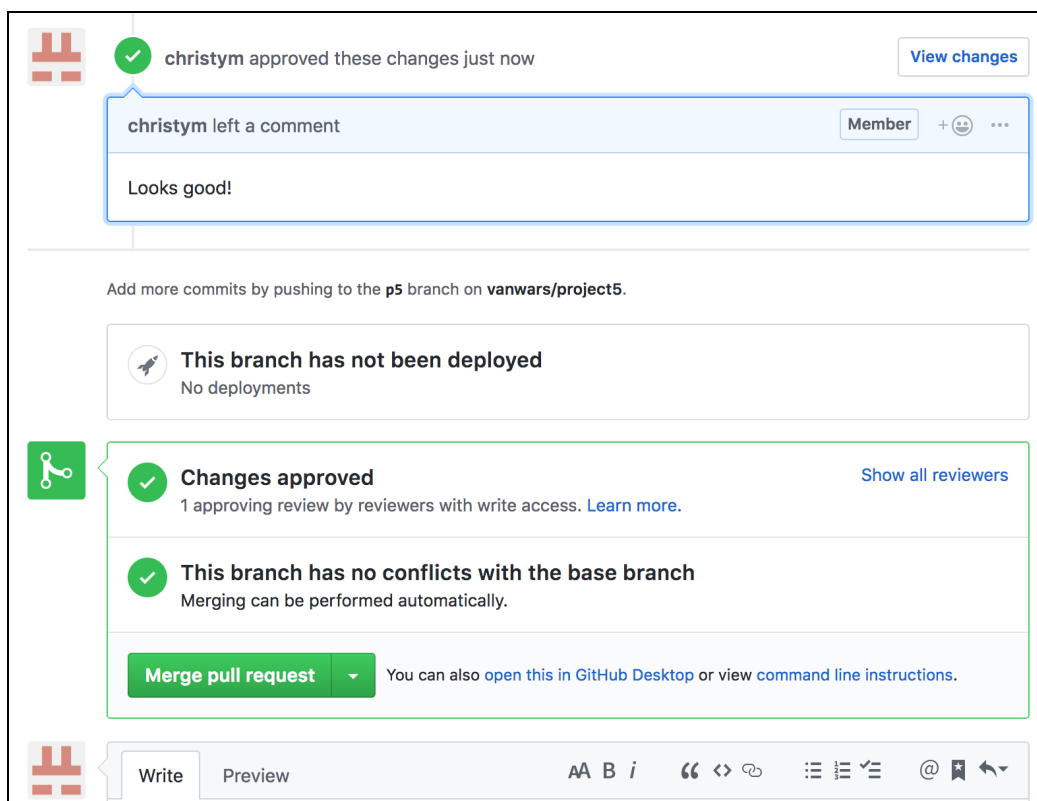
1. On your organization's GitHub page, navigate to the pull requests tab.
2. Click on a pull request and click the green "Add your review" button. Take a look at the code.



3. You may either approve it or request changes and gesture to particular lines of particular files. For now, just approve it and click the "Approve" radio button, and click the green "Submit review" button.



- The PR is now ready to be merged. Click the green “Merge pull request” button to merge your teammate’s change with your organization’s master branch.



5. Once your PR has been merged, you can delete your branch on Github (you can always make it again if needed).

7. Sync your organization's repo with your local computer

When a change has been made to your organization's repository, it's important that everyone on your team "pulls" that change into their local copy of the codebase (on their respective local computers). To do this, execute the following commands on the command line:

```
# switch branches back to master (IMPORTANT)
$ git checkout master
# fetch all of the changes from your organization's (i.e. upstream) master
$ git pull
```

After completing these new steps, you can either (a) merge these new changes into the branch you're currently working on locally, or (b) create a brand new branch from the newly updated master branch.

A. Merge new changes into a branch you're already working on

```
# switch from master branch back to the branch you were working on:
$ git checkout my-branch

# merge all of the most recent changes to the repo (now in master)
# into your branch:
$ git merge master
```

B. Create a brand new branch from master

Before you create a branch, first ensure that you are currently on master. This will ensure that your new branch starts out identical to master:

```
$ git checkout master
$ git branch
# create your new branch (replace <name-of-new-branch> with an actual name):
$ git branch <name-of-new-branch>
# check out your new branch and get to work:
$ git checkout <name-of-new-branch>
```

After creating and checking out your new branch, you're ready to get to work again. And remember, to push your new branch to GitHub, do this:

```
# replace <name-of-new-branch> with an actual name
$ git push --set-upstream origin <name-of-new-branch>
```

IMPORTANT: DO NOT MAKE CHANGES TO YOUR LOCAL COPY OF THE MASTER BRANCH.

The master branch is just for pulling down the latest changes from your organization's repo to your local machine.

8. Enable GitHub Pages

GitHub also allows you to host pages. To enable GitHub pages, go to your organization's repository and click on the settings tab. Then, scroll halfway down to the GitHub Pages section, and in the dropdown menu below "Source," select your master branch and then click the "Save" button.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

✓ Your site is published at <https://eecs-330.github.io/project5/>

Source

Your GitHub Pages site is currently being built from the master branch. [Learn more.](#)

master branch ▾

Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

Custom domain

Custom domains allow you to serve your site from a domain other than `eecs-330.github.io`. [Learn more.](#)

Save

This will generate a link to your GitHub site. A few caveats:

1. GitHub pages are a bit buggy. For whatever reason, for the first few hours after you enable github pages , you'll have to tack on a index.html after the root website so

that you don't get a HTTP404 (page not found) error. Eventually, GitHub pages will resolve the root URL.

2. GitHub pages does not support backends, so if you want a more complex prototype that requires installing backend software, you might want to use something else (e.g. Heroku, Docker, etc.)

9. Create a branch to submit [What to Turn In]

When you're done with steps 1-7 and every person on your team has both (a) created a Pull Request with a code change, and (b) reviewed someone else's Pull Request and merged it, you are ready to submit. Please read the submission instructions carefully. Note that only one person on your team will need to do this:

1. On your local computer, checkout your master branch and rebase.

```
$ git checkout master
```

```
$ git pull
```

2. Next, create a submission branch. Call it **project_5**, and check it out:

```
$ git branch project_5
```

```
$ git checkout project_5
```

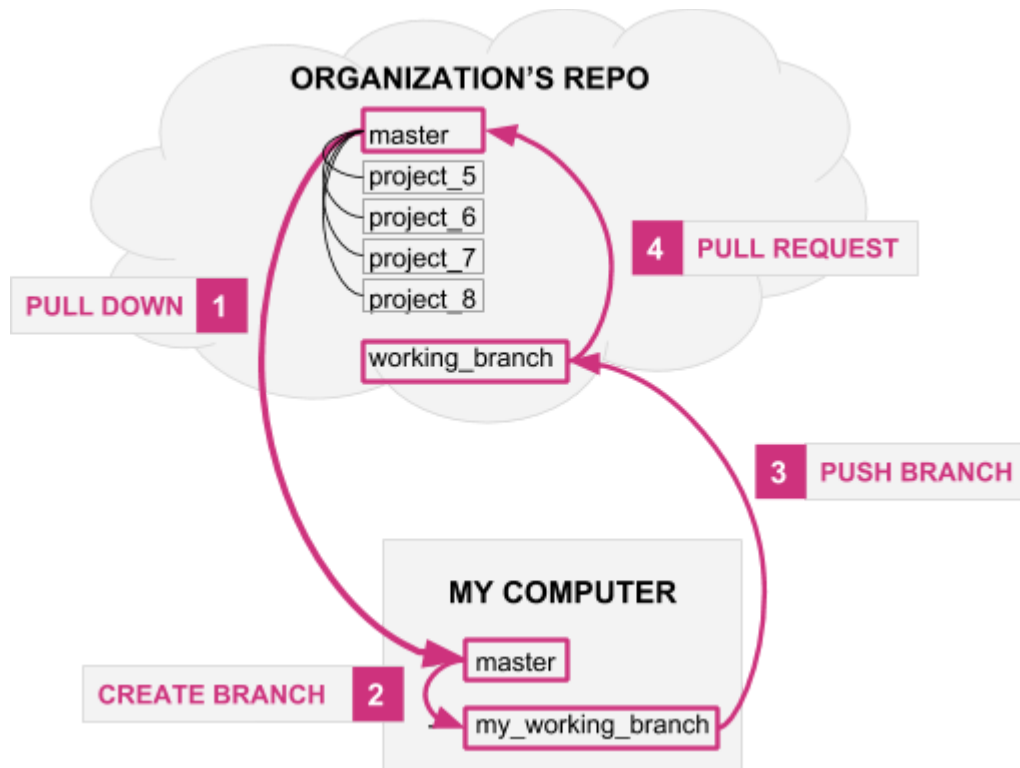
3. Push your project_5 branch to your repository (so that it gets created on the Organization's repo):

```
$ git push origin project_5
```

4. Go to your Organization's repo and verify that your project_5 branch got created. Then, submit a link to (1) your team's GitHub repository, and (2) your team's GitHub pages link to Canvas.

5. The project_5 branch serves as a snapshot in time for us to assess. Moreover, now that everything is set up and archived, feel free to any of the files in your master branch and begin working on your computer prototypes (which will not affect your project_5 branch).

SUMMARY. Rules of the Suggested Workflow



1. **Never make edits to the master branch.** Make any code additions, subtractions, etc. on a separate branch.
2. When you are making any edits to your team's repo, you will make them in **your fork of the repo** by:
 - a. Creating a new branch
 - b. Committing files to your new branch
 - c. Pushing them to Github, and
 - d. Making a PR
 - e. Asking someone on your team to review, approve, and merge your PR.
3. Once your PR has been merged, you will rebase on your local machine as follows:


```
# switch branches back to master (IMPORTANT)
$ git checkout master
# fetch all of the changes from your organization's (i.e. upstream)
master
$ git pull --rebase upstream master
# pushes your organization's version of the code onto *your repo (so
that
```



```
# your master and organization's master are in sync)
$ git push
```

4. Then, to create new features, you will create a new branch, copied from master:

```
# make sure you are currently on the master branch, and that
# you make all of your new branches from master
```

```
$ git branch
```

```
# create your new branch:
```

```
git branch <name-of-new-branch>
```

```
# check out your new branch and get to work:
```

```
git checkout <name-of-new-branch>
```

5. Each week, you will submit a branch of your code called (project_6, project_7, or project_8) that we will assess.