

GITHUB

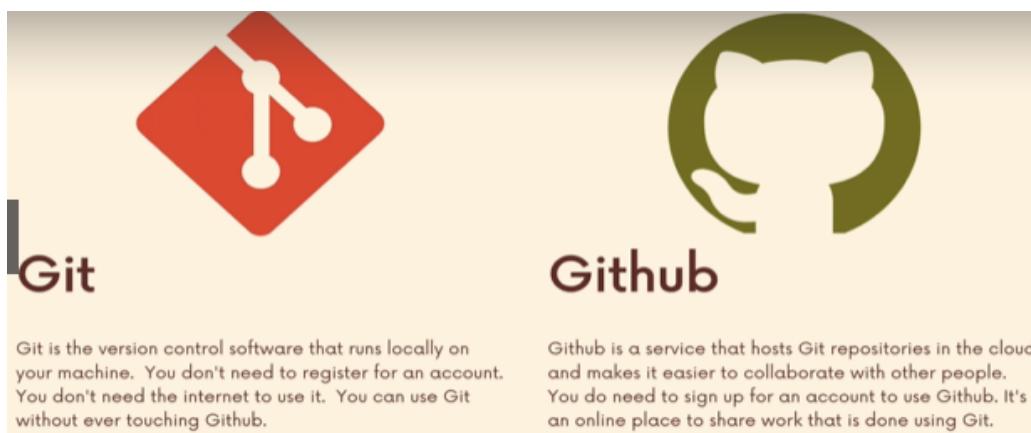
What Is Github?

Github is a hosting platform for git repositories. You can put your own Git repos on Github and access them from anywhere and share them with people around the world.

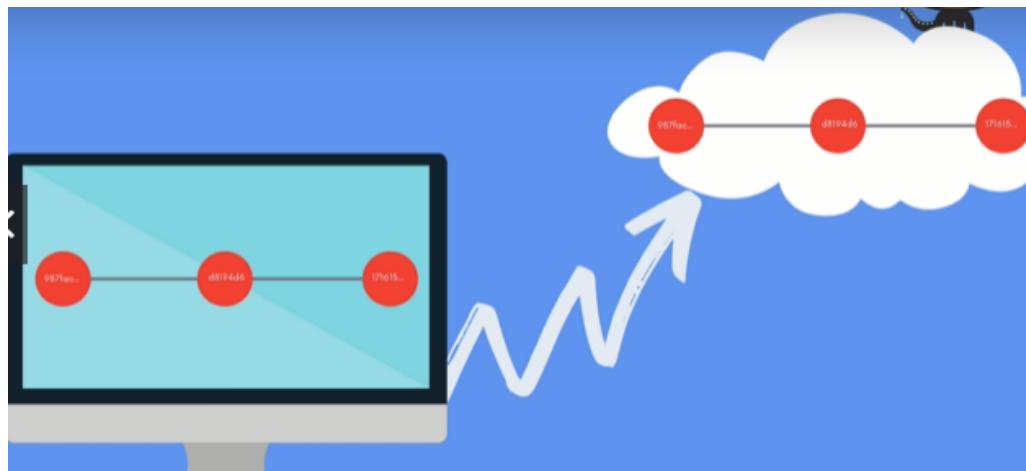
Beyond hosting repos, Github also provides additional collaboration features that are not native to Git (but are super useful). Basically, Github helps people share and collaborate on repos.



Its difficult to answer that question, cuz over the year it has add bunch of features.

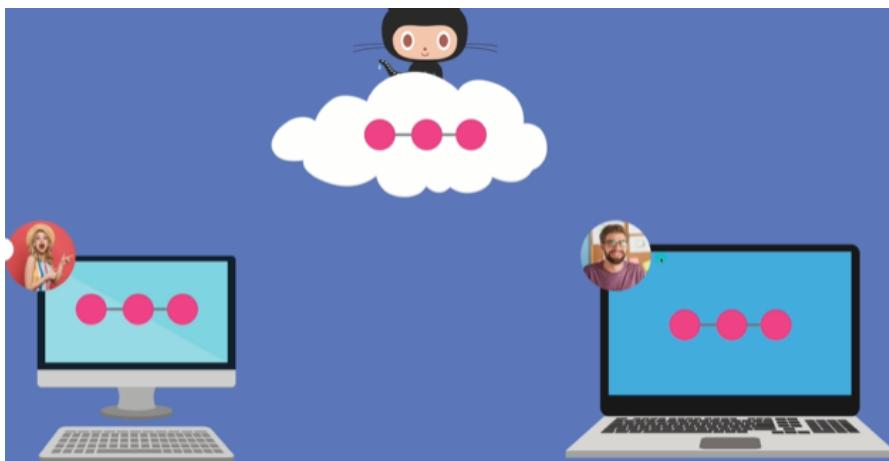


The diagram compares Git and Github. On the left, under 'Git', there is a red diamond icon with a white branching line and dots, labeled 'Git'. Below it, a text box states: 'Git is the version control software that runs locally on your machine. You don't need to register for an account. You don't need the internet to use it. You can use Git without ever touching Github.' On the right, under 'Github', there is a green circle icon with a white cat head, labeled 'Github'. Below it, a text box states: 'Github is a service that hosts Git repositories in the cloud and makes it easier to collaborate with other people. You do need to sign up for an account to use Github. It's an online place to share work that is done using Git.'



1.backup.

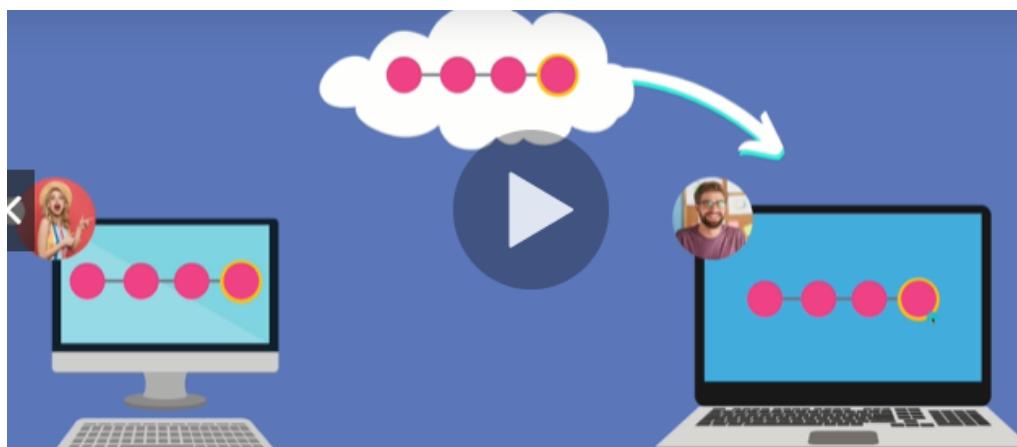
2. collaborate



in left diagram, zelda is an engineer, and lincoln is graphic designer.

Using github they can all clone and get access to this repositories in their machine and they can make contributions and put it on github and pull their contributions.

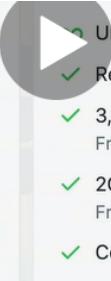
Or commits.



It's Free!

GitHub offers its basic services for free! While GitHub does offer paid Team and Enterprise tiers, the basic Free tier allows for unlimited public and private repos, unlimited collaborators, and more!

Free	Team	Enterprise	GitHub One
Basics for teams and developers	Advanced collaboration and support for teams	Security, compliance, and flexible deployment for enterprises	All of our best tools, support, and services
<ul style="list-style-type: none"> ○ Unlimited public/private repositories ○ Unlimited collaborators ✓ 2,000 Actions minutes/month Free for public repositories ✓ 500MB of GitHub Packages storage Free for public repositories ✓ Community Support 	<ul style="list-style-type: none"> ○ Unlimited public/private repositories ✓ Required reviewers ✓ 3,000 Actions minutes/month Free for public repositories ✓ 20B of GitHub Packages storage Free for public repositories ✓ Code owners 	<ul style="list-style-type: none"> ↳ Everything included in Team ✓ SAML single sign-on ✓ 50,000 Actions minutes/month Free for public repositories ✓ 50GB of GitHub Packages storage Free for public repositories ✓ Advanced auditing 	<ul style="list-style-type: none"> ↳ Everything included in Enterprise ✓ Community-powered security ✓ Actionable metrics ✓ 24/7 support ✓ Continuous learning
\$0 /month	\$4 per user/month	\$21 per user/month	Learn more

<ul style="list-style-type: none"> ∞ Unlimited public/private repositories ∞ Unlimited collaborators ✓ 2,000 Actions minutes/month Free for public repositories ✓ 500MB of GitHub Packages storage Free for public repositories ✓ Community Support 	 <ul style="list-style-type: none"> ∞ Unlimited public/private repositories ✓ Required reviewers ✓ 3,000 Actions minutes/month Free for public repositories ✓ 2GB of GitHub Packages storage Free for public repositories ✓ Code owners 	<ul style="list-style-type: none"> ◀ Everything included in Team ✓ SAML single sign-on ✓ 50,000 Actions minutes/month Free for public repositories ✓ 50GB of GitHub Packages storage Free for public repositories ✓ Advanced auditing
---	--	---

Social network for git repos, you can chat, write comments, approve , disapprove, suggest changes to projects , merge code in , reject code , call attention to certain lines of code.

Interactive developer enviornment : codespaces. Additional features like automatically scanning your repositories and looking for common vulnerbalities.

And letting you know ahead of time.

Why You should Use Github

For collaboration, backup, resume, trying to meet potential mentors.

Open Source Projects:

Open Source Projects

Today Github is THE home of open source projects on the Internet. Projects ranging from React to Swift are hosted on Github.

If you plan on contributing to open source projects, you'll need to get comfortable working with Github.



is where the codebase is available. Not just to typically (for viewing and using) but also contributions. To Fix bugs and to Discuss with greater community of ppl.

Basically the users of the product. Github is the home of Open source projects over the internet today.

Open source contribution is a great prove that you know what you doing.

For e.g: React, Tensorflow.

Exposure

Your Github profile showcases your own projects and contributions to others' projects. It can act as a sort of resumé that many employers will consult in the hiring process. Additionally, you can gain some clout on the platform for creating or contributing to popular projects.

[Overview](#) [Repositories 43](#) [Projects](#) [Packages](#)

[Type: All](#) [Language: All](#)

Moonstruck
A minimalist, design-focused website for an ongoing concert series used by 300+ concert-goers - JavaScript, HTML5, SCSS, CSS3
● SCSS ★ 5 Updated 13 days ago 

Harvester
A website to harvest local produce built with Node.js, ReactJS, GraphQL, Apollo Client, and MongoDB
● JavaScript ★ 6 Updated 27 days ago

algorithms
● JavaScript ★ 2 Updated on Dec 12, 2020

Adventurebnb
Airbnb clone using ReactJS, Redux, and Ruby on Rails
● Ruby ★ 6 ⌚ 5 Updated on Jul 30, 2020

rails-ani
● Ruby Updated on Jun 25, 2020

[master](#) [1 branch](#) [0 tags](#) [Go to file](#) [Code](#)

 Carly-Schaaf Use https	dd5e99ee 13 days ago	 43 commits	
 .idea	Update for next show	9 months ago	
 .sass-cache	fix media queries	2 years ago	
 images	create custom favicon	2 years ago	
 javascript	Update content	13 days ago	
 styles	Edit promise cb	10 months ago	
 .DS_Store	create custom favicon	2 years ago	
 .gitignore	Update for next show	9 months ago	
 .htaccess	Update everything	2 years ago	
 README.md	Update README.md	15 months ago	
 faqs.html	organize files	2 years ago	
 index.html	Use https	13 days ago	

About
A minimalist, design-focused website for an ongoing concert series used by 300+ concert-goers - JavaScript, HTML5, SCSS, CSS3
[Readme](#)

Releases
No releases published

Packages
No packages published

Languages

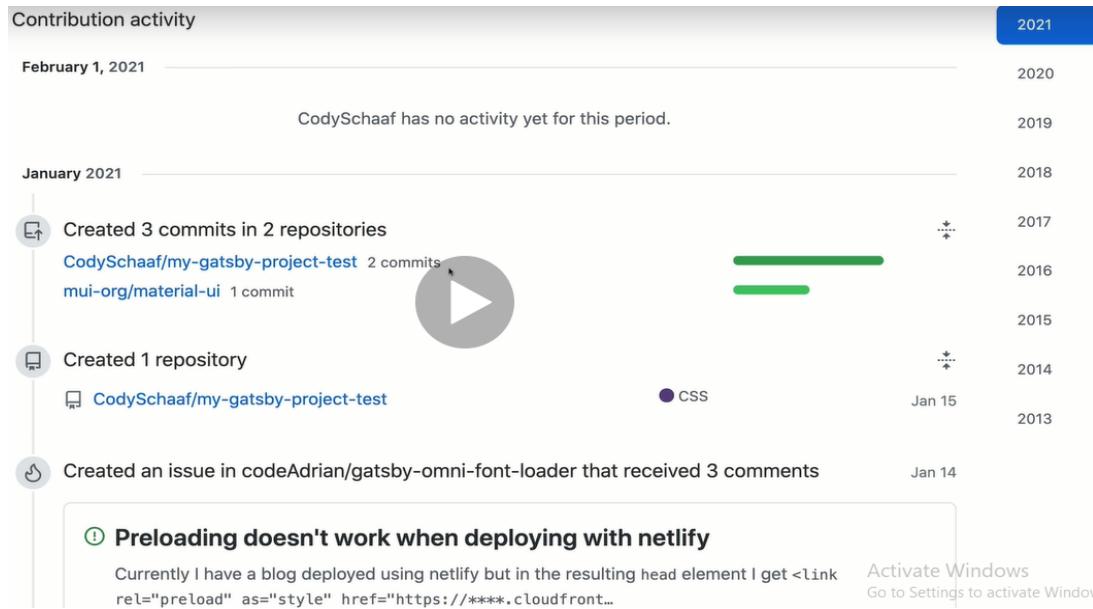

- SCSS 34.6%
- HTML 33.6%
- JavaScript 25.9%
- CSS 5.9%

README.md

Moonstruck
[Live Link](#)

Moonstruck is a minimalist, design-focused website I designed and built for an ongoing concert series. It has been used by 300+ concert-goers.





ⓘ Preloading doesn't work when deploying with netlify

Currently I have a blog deployed using netlify but in the resulting head element I get <link rel="preload" as="style" href="https://****.cloudfront...

Activate Windows
Go to Settings to activate Window

dependabot-preview Bump next from 10.0.5 to 10.0.6 (#24714) cc4a632 6 hours ago 15,931 commits

.circleci [test] Test declaration files in TS nightly (#24391) 19 days ago

.codesandbox [core] Skip downloading browser binaries in codesandbox/ci (#24628) 6 days ago

.dependabot [core] Fix 'next' using stale pages (#24635) 5 days ago

.github Release v5.0.0-alpha.24 (#24614) 6 days ago

.yarn/releases [core] Use common yarn version (#21779) 7 months ago

benchmark Release v5.0.0-alpha.24 (#24614) 6 days ago

docs [docs] Fix indent 11 hours ago

examples [examples] Update examples to use StyledEngineProvider (#24489) 3 days ago

framer [Chip] Migrate to emotion (#24649) 3 days ago

modules/waterfall [core] Misc modules/* cleanup (#22983) 4 months ago

packages [Link] Fix CSS prefix property casing with emotion (#24701) yesterday

scripts [docs] Add API documentation for *DatePicker components (#24655) 3 days ago

test [test] Conformance to handle wrapped elements (#24679) 2 days ago

.browserslistrc [core] Add support for iOS Safari 12 (#23068) 4 months ago

.codecov.yml [test] Increase the Codecov threshold (#14796) 2 years ago

.editorconfig [docs] Change http to https part 2 (#16171) 2 years ago

.eslintignore [core] Fix formatting (#23567) 3 months ago

.eslintrc.js [core] Fix 'next' using stale pages (#24635) 5 days ago

.gitattributes [core] Force LF for text files (#23932) 2 months ago

.gitignore [core] All packages are published from /build (#23886) 2 months ago

.mochrc.js [core] Batch small changes (#24599) 6 days ago

.stylelintrc.js [docs] Add API of picker components (#24497) 7 days ago

About

Material-UI is a simple and customizable component library to build faster, beautiful, and more accessible React applications. Follow your own design system, or start with Material Design.

[material-ui.com/](#)

react javascript design-systems
typescript material-design
react-components hacktoberfest

[Readme](#)

[MIT License](#)

Releases 302

v4.11.3 (Latest)
8 days ago

+ 301 releases

Sponsor this project

[opencollective.com/material-ui](#)
[tidelift.com/funding/github/npm/@...](#)

Used by 438k

+ 438,437

Stay Up To Date

Being active on GitHub is the best way to stay up to date with the projects and tools you rely on. Learn about upcoming changes and the decisions/debate behind them.

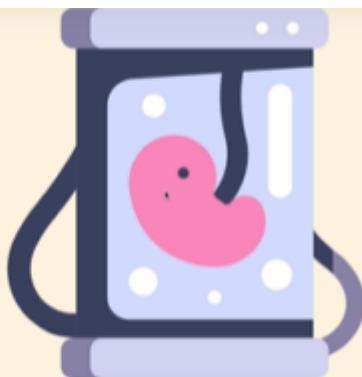


Cloning Github Repos With Git Clone

Cloning

So far we've created our own Git repositories from scratch, but often we want to get a local copy of an existing repository instead.

To do this, we can clone a remote repository hosted on Github or similar websites. All we need is a URL that we can tell Git to clone for use.



git clone

To clone a repo, simply run `git clone <url>`.

Git will retrieve all the files associated with the repository and will copy them to your local machine.

In addition, Git initializes a new repository on your machine, giving you access to the full Git history of the cloned project.



➤ `git clone <url>`

Make sure you are not inside
of a repo when you clone!

Go to file Code ▾

Clone ⓘ

HTTPS GitHub CLI

`https://github.com/gabrielecirulli/`

Use Git or checkout with SVN using the `web` URL.

Open with GitHub Desktop

Download ZIP

Contributing 6 years ago

7 years ago

```
Cloning > ls
Cloning > git clone https://github.com/gabrielecirulli/2048.git
Cloning into '2048'...
remote: Enumerating objects: 1315, done.

 README.md      index.html
 2048 > git status
On branch master
Your branch is up to date wit

nothing to commit, working tr
 2048 > git log
```

Hiring Without Whiteboards



A list of companies (or teams) that don't do "whiteboard" interviews. "Whiteboards" is used as a metaphor, and is a symbol for the kinds of CS trivia questions that are associated with bad interview practices. Whiteboards are not bad – CS trivia questions are. Using sites like HackerRank/LeetCode probably fall into a similar category.

The companies and teams listed here use interview techniques and questions that resemble day-to-day work. For example, pairing on a real world problem or a paid/unpaid take home exercise. Read (and contribute to) our [recommendations](#) for ways to conduct better interviews.

tl;dr

Cloning Non-Github Repos

Permissions?

Anyone can clone a repository from Github, provided the repo is public. You do not need to be an owner or collaborator to clone the repo locally to your machine. You just need the URL from Github.

Pushing up your own changes to the Github repo...that's another story entirely! You need permission to do that!



git clone works with any type of repositories.

Github Setup: SSH Config

SSH Keys

You need to be authenticated on Github to do certain operations, like pushing up code from your local machine. Your terminal will prompt you every single time for your Github email and password, unless...

You generate and configure an SSH key! Once configured, you can connect to Github without having to supply your username/password.



Creating Our First Github Repo!

Option 1: Existing Repo

If you already have an existing repo locally that you want to get on Github...

- Create a new repo on Github
- Connect your local repo (add a remote)
- Push up your changes to Github

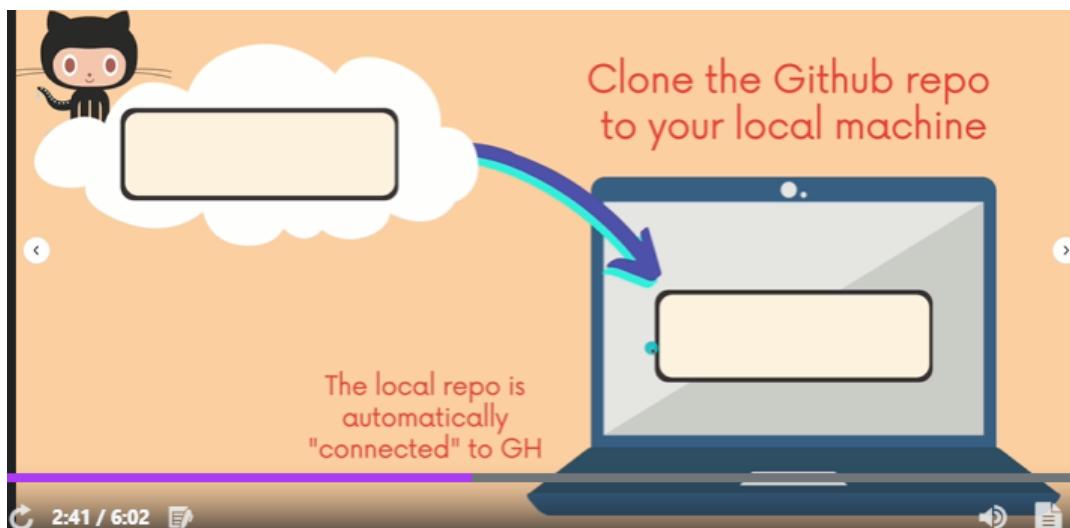
Option 2: Start From Scratch

If you haven't begun work on your local repo, you can...

- Create a brand new repo on Github
- Clone it down to your machine
- Do some work locally
- Push up your changes to Github



If i do that way then i dont have to manually connect my local repository to my github repository. If i clone this repo from github its automatically connected to that github url.



Quick setup — if you've done this kind of thing before

 Set up in Desktop or [HTTPS](#) [SSH](#) https://github.com>IDK9911/to_learn_git.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 "# to_learn_git" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com>IDK9911/to_learn_git.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com>IDK9911/to_learn_git.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

A Crash Course on Git Remotes

Remote

Before we can push anything up to Github, we need to tell Git about our remote repository on Github. We need to setup a "destination" to push up to.

In Git, we refer to these "destinations" as remotes. Each remote is simply a URL where a hosted repository lives.

Viewing Remotes

To view any existing remotes for your repository, we can run `git remote` or `git remote -v` (verbose, for more info)

```
● ● ●  
❯ git remote -v
```

This just displays a list of remotes. If you haven't added any remotes yet, you won't see anything!

When I cloned a directory from GitHub, I get this along with it.

```
2048 > git remote                               master  
origin  
2048 > git remote -v                           master  
origin https://github.com/gabrialecirulli/2048.git (fetch)  
origin https://github.com/gabrialecirulli/2048.git (push)  
2048 > █                                     master
```

By setting this URL (remote), it gives us the ability to tell if there is any new code on the URL, is there any new updates.

Or if I was a collaborator, hey Git I would like to push up my 10 commits.

Adding A New Remote

A remote is really two things: a URL and a label.

To add a new remote, we need to provide both to Git.

```
● ● ●  
❯ git remote add <name> <url>
```

Adding A New Remote

```
● ● ●  
❯ git remote add origin  
https://github.com/blah/repo.git
```

Okay Git, anytime I use the name "origin", I'm referring to this particular GitHub repo URL.

Origin?

Origin is a conventional Git remote name, but it is not at all special. It's just a name for a URL.

When we clone a Github repo, the default remote name setup for us is called origin. You can change it. Most people leave it.

Other commands

They are not commonly used, but there are commands to rename and delete remotes if needed.

```
git remote rename <old> <new>
```

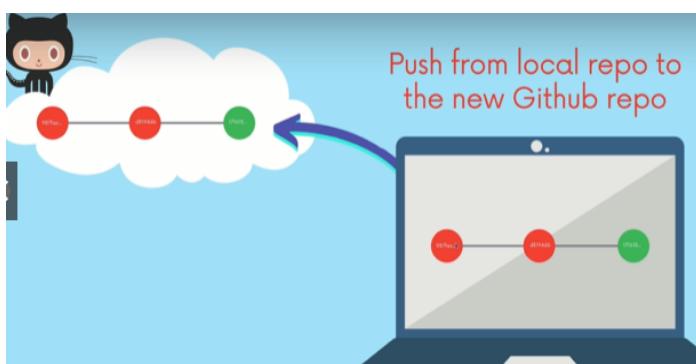
```
git remote remove <name>
```

Introducing Git Push

Option 1: Existing Repo

If you already have an existing repo locally that you want to get on Github...

- Create a new repo on Github
- Connect your local repo (add a remote)
- Push up your changes to Github



Pushing

Now that we have a remote set up, let's push some work up to Github! To do this, we need to use the `git push` command.

We need to specify the remote we want to push up to AND the specific local branch we want to push up to that remote.

```
> git push <remote> <branch>
```

In a repository, we often have 7 or 8 branches, so when we push we may not want to push them all on github.

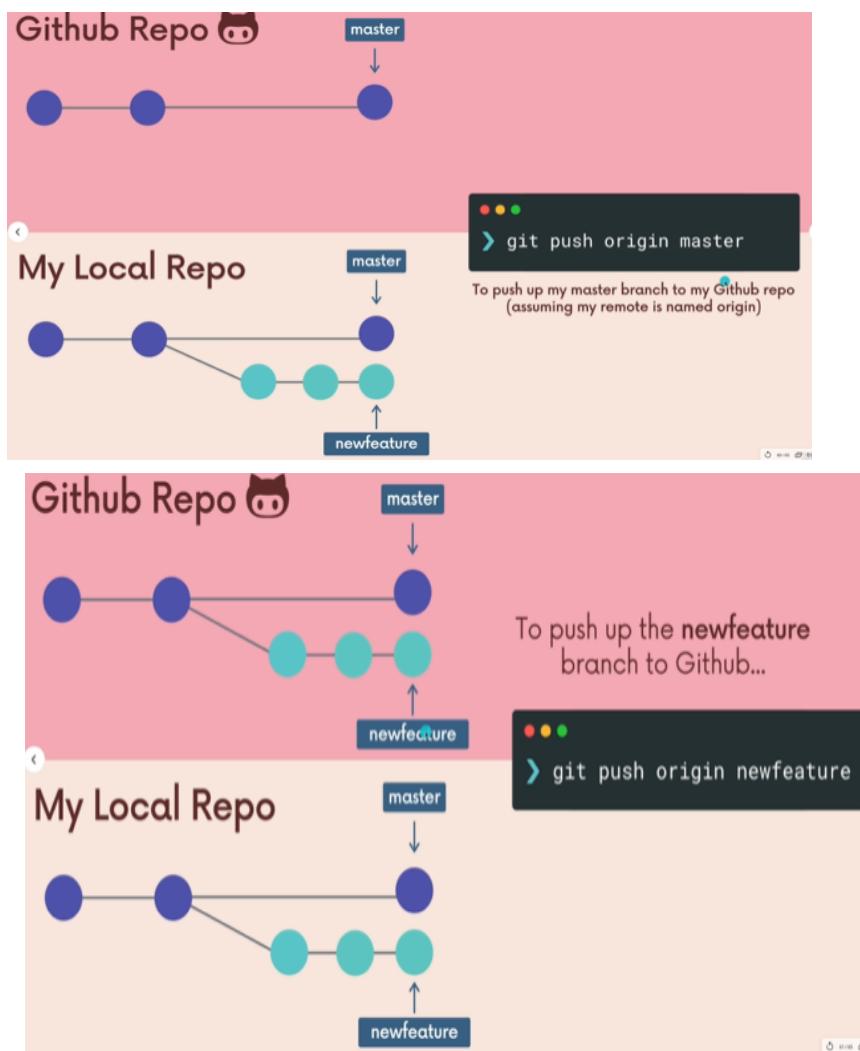
Often it is master or main branch.

An Example

git push origin master tells git to push up the master branch to our origin remote.

git push origin master

```
MyFirstNovel > git switch -c empty
Switched to a new branch 'empty'
MyFirstNovel > ls
MoodBoard      chapter2.txt    characters.txt
chapter1.txt   chapter3.txt    outline.txt
MyFirstNovel > rm -rf MoodBoard
MyFirstNovel > rm chapter1.txt chapter2.txt chapter3.txt
MyFirstNovel > ks
zsh: command not found: ks
MyFirstNovel > ls
characters.txt outline.txt
no changes added to commit (use "git add" and/or "git commit")
MyFirstNovel > git add .
MyFirstNovel > git commit -m "delete most things"
[empty 0ba2122] delete most things
 8 files changed, 584 deletions(-)
 delete mode 100644 MoodBoard/bar.jpeg
 delete mode 100644 MoodBoard/billboard.jpeg
 delete mode 100644 MoodBoard/mural.jpg
 delete mode 100644 MoodBoard/rolls.jpg
 delete mode 100644 MoodBoard/woman.jpg
 delete mode 100644 chapter1.txt
 delete mode 100644 chapter2.txt
 delete mode 100644 chapter3.txt
```

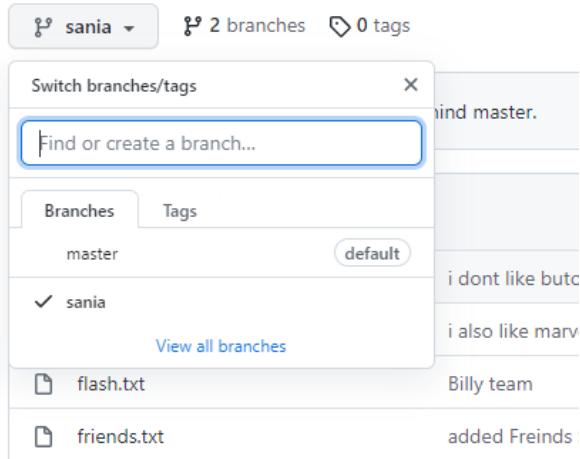


```
sahilsyed@DESKTOP-7VRCIK9:/mnt/c/Users/Dell/Desktop/Learning Git/kaju$ git push origin sania
Username for 'https://github.com': IDK9911
Password for 'https://IDK9911@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'sania' on GitHub by visiting:
remote:     https://github.com/IDK9911/to_learn_git/pull/new/sania
remote:
To https://github.com/IDK9911/to_learn_git.git
 * [new branch]      sania -> sania
```

sania had recent pushes 1 minute ago

[Compare & pull request](#)

branch master
branches 2
tags 0
Go to file
Add file
Code



```
(use "git add <file>..." to include in what will be committed)
chapter4.txt
```

```
nothing added to commit but untracked files present (use "git add <file>..." to include in what will be committed)
MyFirstNovel > git add chapter4.txt
MyFirstNovel > git commit -m "begin chapter 4"
[master 02c0f23] begin chapter 4
 1 file changed, 0 insertions(+), 0 deletions(-)
```

When we commit, we don't just push that commit we push the entire branch.

Touring a git repository

This branch is 1 commit ahead, 1 commit behind master.

Colt delete most things

characters.txt add headings to all files

outline.txt add headings to all files

A screenshot of a GitHub repository tour. It shows a message 'This branch is 1 commit ahead, 1 commit behind master.' above a list of files. The first file, 'characters.txt', has a note 'add headings to all files'. The second file, 'outline.txt', also has a note 'add headings to all files'. There is a play button icon next to the commit message.

It means this branch has 1 commit that master doesn't have, and is missing one commit that master doesn't have.

The screenshot shows a GitHub repository interface. At the top, there's a list of files with their commit history:

File	Author	Date
Boys.txt	Billy team	yesterday
Loki.txt	i also like marvel loki series	yesterday
flash.txt	Billy team	yesterday
friends.txt	wb now	6 hours ago

Below this, a dropdown menu shows the current branch: **master**. To the right, there are three commit cards:

- 7b104f9 6 hours ago 18 commits** (highlighted with a red box)
- e362ab1** (with a tooltip: "Browse the repository at this point in the history")
- ff051eb**
- 035dc54**
- b80fe96**

On the left, a sidebar shows commit history by date:

- o Commits on Aug 26, 2021
 - wb now** (IDK9911 committed 6 hours ago)
 - still on test** (IDK9911 committed 6 hours ago)
 - 1st test** (IDK9911 committed 6 hours ago)
- o Commits on Aug 25, 2021

At the bottom, a detailed view of the **friends.txt** file shows a diff between lines 1 and 4:

Line	Original	Modified
1	Chandler Bing	Chandler Bing
2	Joey Tribbiani	Joey Tribbiani
3	Monica Geller	Monica Geller
4	+ Ross Geller	(highlighted in green)

Note u can edit the file by clicking on the pencil icon. and then if u r on the main branch.
u might not be able to do it.
Cuz of branch protection rules.
so u can create feature branch.

0 comments on commit e362ab1

Write

Preview

Aa ^



Leave a comment

Commenting is really important. It's important for discussing changes as you start working with other people.

A closer Look At Git push

An Example

git push origin master tells git to push up the master branch to our origin remote.

git push origin master

```
Pushing > git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 231 bytes | 231.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Colt/pushme.git
 * [new branch]      master -> master
```

Push In Detail

While we often want to push a local branch up to a remote branch of the same name, we don't have to!

To push our local pancake branch up to a remote branch called waffle we could do:
git push origin pancake:waffle

git push <remote>
<local-branch>:<remote-branch>

git push origin pancake:waffle

Take the local branch pancake and push it to the origin branch waffle.

Pushing > git push origin cats

cats

Enumerating objects: 2, done.

Pushing > git push origin cats:master

```
Pushing > git push origin cats:master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Colt/pushme.git
  f508e11..f3d84fa  cats -> master
Pushing > [REDACTED]
```

cats

cats

What does “git push -u” mean?

The -u option

The -u option allows us to set the upstream of the branch we're pushing. You can think of this as a link connecting our local branch to a branch on Github.

```
...> git push -u origin master
```

Running `git push -u origin master` sets the upstream of the local master branch so that it tracks the master branch on the origin repo.

What this means...

Once we've set the upstream for a branch, we can use the `git push` shorthand which will push our current branch to the upstream.

```
...> git push -u origin master
...> git push
```



Next time we push...

This tells git on my computer, I want to push up the master branch to origin and I want you to remember it.

```
Pushing > git switch -c dogs                               master
Switched to a new branch 'dogs'
Pushing > █                                            dogs
```

```
Pushing > git push origin dogs                           dogs
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'dogs' on GitHub by visiting:
remote:     https://github.com/Colt/pushme/pull/new/dogs
remote:
To github.com:Colt/pushme.git
 * [new branch]      dogs -> dogs
Pushing > touch dogs.txt                                dogs
Pushing > git add dogs.txt                             dogs
Pushing > git commit -m "create dogs"                  dogs
[dogs 7f0d5e2] create dogs
 1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 dogs.txt
Pushing > git push                                     dogs
```

```
-----  
Pushing > git push
fatal: The current branch dogs has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin dogs
```

```
Pushing > git push -u origin dogs
```

```
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Colt/pushme.git
 f508e11..7f0d5e2  dogs -> dogs
Branch 'dogs' set up to track remote branch 'dogs' fr
Pushing > touch moredogs.txt
Pushing > git add moredogs.txt
Pushing > git commit -m "create more dogs"
[dogs 3910353] create more dogs
 1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 moredogs.txt
Pushing > git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
```

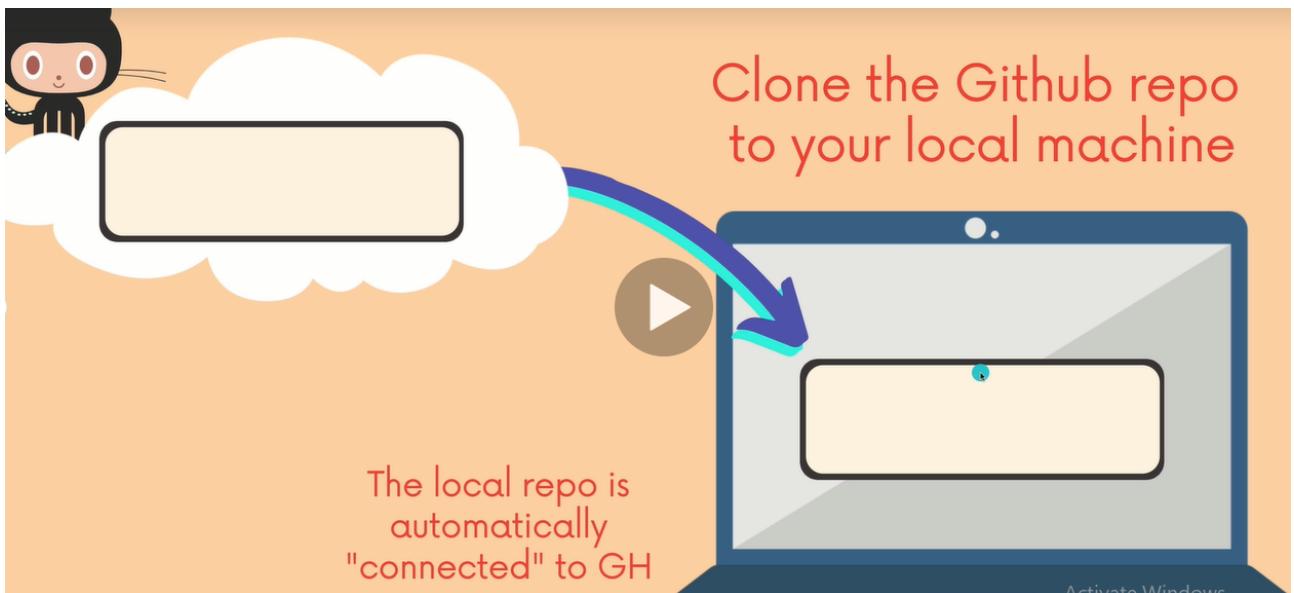
```
Pushing > git push -u origin dogs:cats
```

If I ran the above, not only it would make dog branch at cats on github. But it will remember this too.

git push -u origin cats.

So be careful.

Another Github Workflow: Cloning first.



Automatically do the remote for us.

Main & Master : Github Default branches

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

if i uncheck Readme, it doesn't say anything about Main, cuz there is no default branch.

This will set `main` as the default branch. Change the default name in your [settings](#).

We can change that in our settings that it always default to master.

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'main' (selected), '1 branch', '0 tags', 'Go to file', 'Add file', and 'Code'. A commit card for 'Colt Initial commit' is shown, with a timestamp of '27fcef now' and '1 commit'. Below the commit, a file named 'README.md' is listed with the content 'colors'.

```
git remote add origin https://github.com/Colt/asdasd.git
git branch -M main
git push -u origin main < it tells to rename the
branch we currently on
with main
```

```
chickens-demo > git branch
chickens-demo > git branch -M main
chickens-demo >
```

master
master
main

```
chickens-demo > git push origin main
Total 0 (delta 0), reused 0 (delta 0)
```



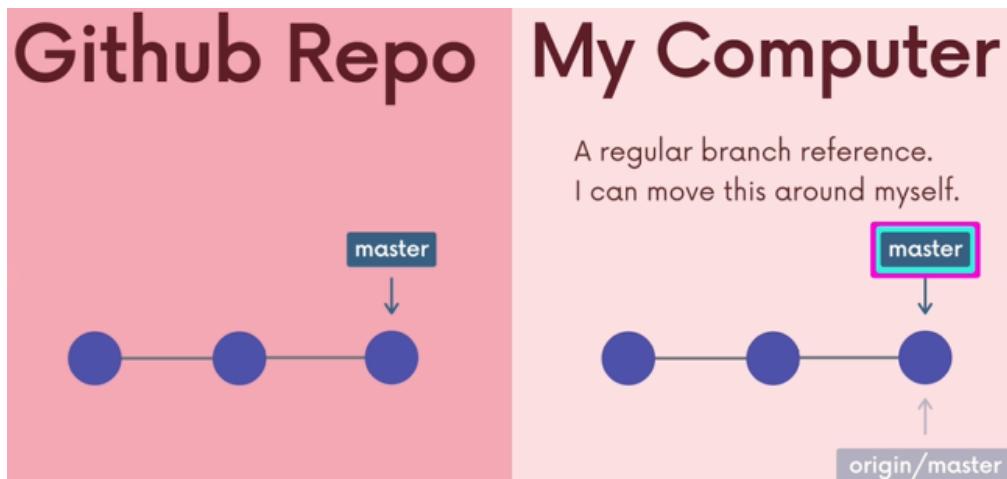
You will Still see master is still consider the default branch. Bcoz we push master first.

A screenshot of the GitHub repository settings page. At the top, there are tabs for 'Code', 'Wiki', 'Security', 'Insights', and 'Settings' (selected). On the left, a sidebar has tabs for 'Options', 'Manage access', 'Security & analysis', and 'Branches' (selected). The main content area is titled 'Default branch' and contains the text: 'The default branch is considered the "base" branch for pull requests and other operations made, unless you specify a different branch.' Below this text is a button labeled 'master'.

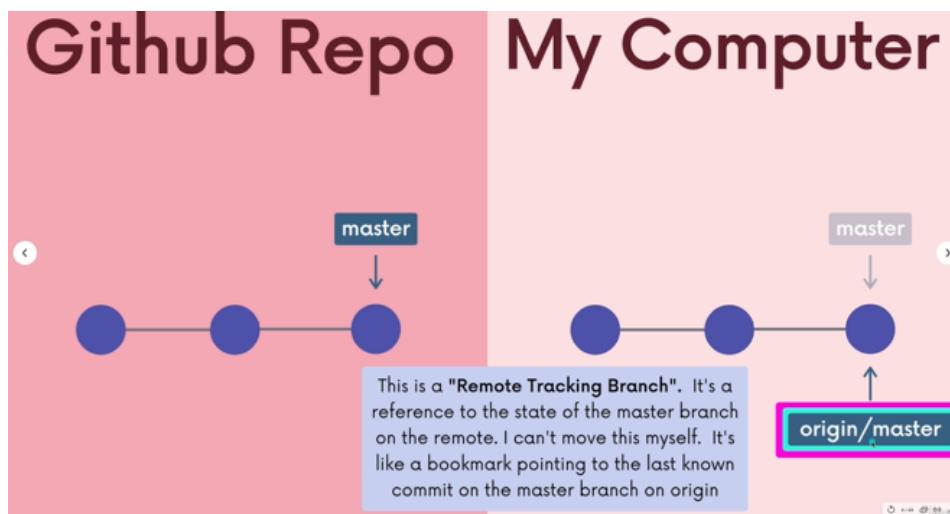
Fetching and Pulling

We are talking about remote repositories, we are talking about git fetch and git pull. They both are talking about getting changes down from the remote.

Remoting tracking branches : WTF are they



Whatever the github repo as its defualt repository, we gonna have that, in this e.g its master, we gonna start with that as our default.



master and origin/master may start form the same starting point, but they might diverge as we do more work aka commits in our local repository.

Remote Tracking Branches

"At the time you last communicated with this remote repository, here is where x branch was pointing"

They follow this pattern <remote>/<branch>.

- `origin/master` references the state of the master branch on the remote repo named origin.
- `upstream/logoRedesign` references the state of the logoRedesign branch on the remote named upstream (a common remote name)



Think of remote tracking branch as the last time you communicated with this remote repository. Here is where, the master branch, pointing at the github repository.

Remote Branches

Run `git branch -r` to view the remote branches our local repository knows about.

```
...  
git branch -r  
origin/master
```

Quick setup — if you've done this kind of thing before

Set up in Desktop or [HTTPS](#) [SSH](#) <https://github.com/Colt/animals.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

The screenshot shows the GitHub interface for committing a new file. On the left, there's a preview of the 'pets.txt' file content:

```
1 Rusty
2 Blue
```

On the right, the 'Commit new file' dialog is open, showing fields for the commit message ('Create pets.txt') and an optional description. At the bottom are 'Commit new file' and 'Cancel' buttons.

main ▾

1 branch

0 tags



Colt Create pets.txt

pets.txt

Create pets.txt

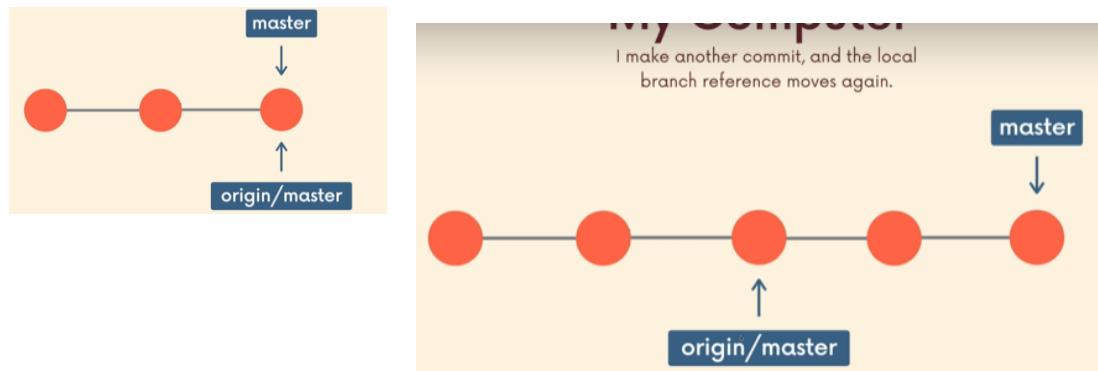
```
animals > git remote -v
origin  https://github.com/Colt/animals.git (fetch)
origin  https://github.com/Colt/animals.git (push)
```

```
animals > git remote -v
origin  https://github.com/Colt/animals.git (fetch) main
origin  https://github.com/Colt/animals.git (push)
```

above, git setup this remote tracking branch called origin at Main.

```
animals > git branch -r
origin/HEAD -> origin/main
origin/main
(END)
```

Checking out remote tracking branch



```
animals > git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

animals > git add pets.txt
animals > git commit -m "add George the Frog"
```

hmm...what did this project look like when I first cloned this repo?

You can checkout these remote branch pointers

```

> git checkout origin/master
Note: switching to 'origin/master'.
You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this blah blah blah

```

Detached HEAD! Don't panic. It's fine.

```

animals > git checkout origin/main
animals > git push origin main

```

Commits on Feb 1, 2021

- add George the Frog
- add childhood cats
- Create pets.txt

```

animals > git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
animals >

```

Working With Remote Branches

Suppose I Just Cloned This Github Repo

Compare & pull request

Compare & pull request

Compare & pull request

main 4 branches 0 tags

Switch branches/tags

Find or create a branch...

Branches Tags

✓ main default

fantasy

food

movies

Create cactus.txt 7fdeb88 6 minutes ago 2 commits

Create rose.txt 7 minutes ago

6 minutes ago

erstand your project by adding a README.

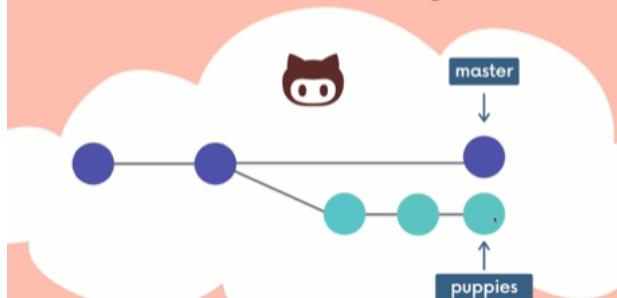
Add a README

```
FetchingPulling > git status
fatal: not a git repository (or any of the
FetchingPulling > git clone https://github.com/colt-holzclaw/remote-branches-demo.git
Cloning into 'remote-branches-demo'...
```

```
remote-branches-demo > git branch
```

```
  * main
  (END)
```

Suppose I Just Cloned This Github Repo



Remote Branches

Once you've cloned a repository, we have all the data and Git history for the project at that moment in time. However, that does not mean it's all in my workspace!

The github repo has a branch called **puppies**, but when I run `git branch` I don't see it on my machine! All I see is the **master** branch. What's going on?

```
● ● ●
> git branch
*master
```

```
remote-branches-demo > git branch -r  
origin/HEAD -> origin/main  
origin/fantasy  
origin/food  
origin/main  
origin/movies  
(END)
```

We see everyone of the branches, it has a remote tracking reference. It can tell me where each one his. Just like a bookmark.

Below, but we don't see the branches.

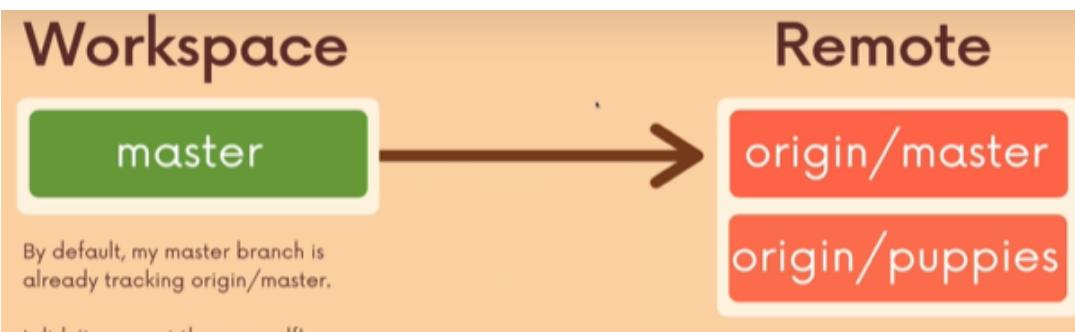
```
* main  
(END)
```

Remote Branches

Run `git branch -r` to view the remote branches our local repository knows about.

```
> git branch -r  
origin/master  
origin/puppies
```

Whenever we clone,



I want to work on the puppies branch locally!

I could `checkout origin/puppies`, but that puts me in detached HEAD.

I want my own local branch called `puppies`, and I want it to be connected to `origin/puppies`, just like my local `master` branch is connected to `origin/master`.



```
remote-branches-demo > git checkout origin/food
Note: switching to 'origin/food'.

You are in 'detached HEAD' state. You can look around,
remote-branches-demo > ls
banana.txt  popsicle.txt          HEAD
```

I want to work on the puppies branch locally!

I could checkout origin/puppies, but that puts me in detached HEAD.

I want my own local branch called puppies, and I want it to be connected to origin/puppies, just like my local master branch is connected to origin/master.



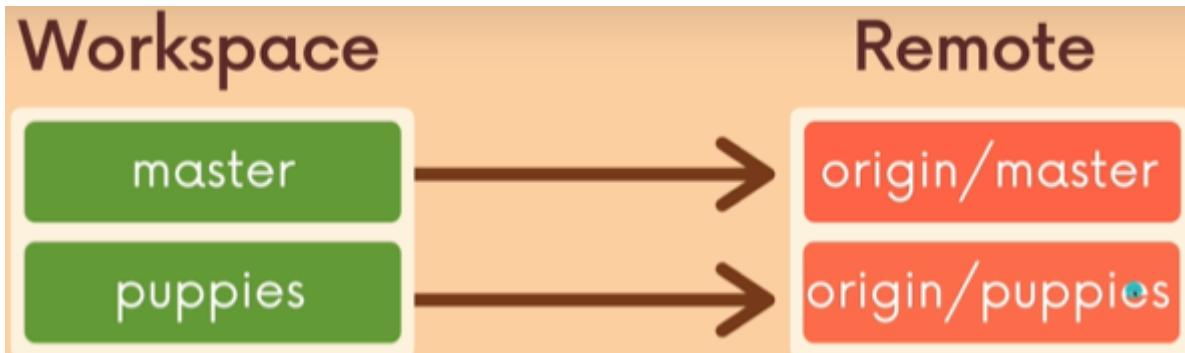
It's super easy!

Run `git switch <remote-branch-name>` to create a new local branch from the remote branch of the same name.

`git switch puppies` makes me a local puppies branch AND sets it up to track the remote branch origin/puppies.

```
... ...
> git switch puppies
```

So if we switch to a branch with that exact name, so if git detects that we are switching to a branch puppies which doesn't exist. It will check then remote branches in the repositories and will make the connection.



```
origin/HEAD -> origin/main
origin/fantasy
origin/food
origin/main
origin/movies
```

remote-branches-demo > git switch movies

Branch 'movies' set up to track remote branch 'movies' from 'origin'.
Switched to a new branch 'movies'

main |

NOTE!

the new command `git switch` makes this super easy to do!
It used to be slightly more complicated using `git checkout`

```
... ...
> git checkout --track origin/puppies
```

```
remote-branches-demo > git status
```

On branch fantasy

Your branch is ahead of 'origin/fantasy' by 1 commit.
(use "git push" to publish your local commits)

fantasy

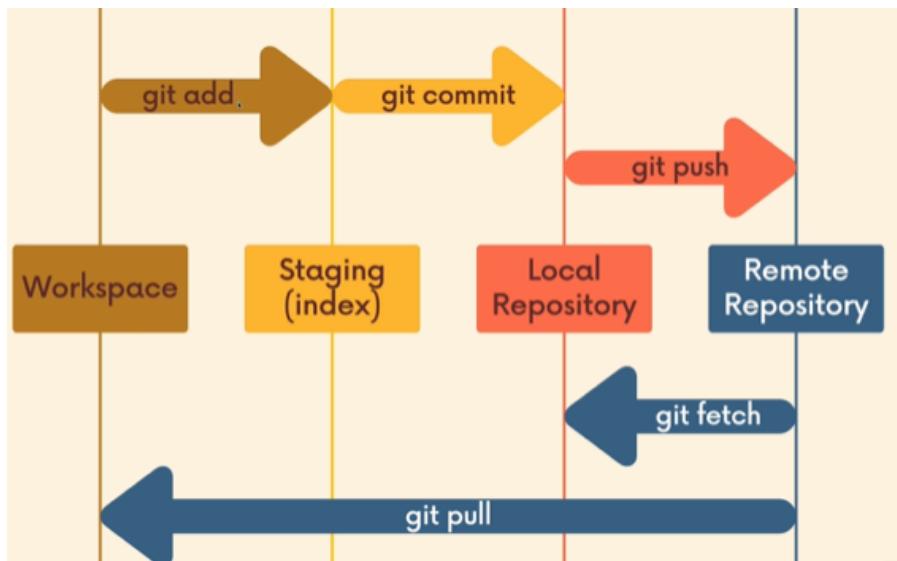
Git Fetch: The Basics

Github 

Uh oh! The remote repo has changed! A teammate has pushed up changes to the master branch, but my local repo doesn't know!

How do I get those changes???

Local 



Fetching

Fetching allows us to download changes from a remote repository, BUT those changes will not be automatically integrated into our working files.

It lets you see what others have been working on, without having to merge those changes into your local repo.

Think of it as "please go and get the latest information from Github, but don't screw up my working directory."

Git Fetch

The `git fetch <remote>` command fetches branches and history from a specific remote repository. It only updates remote tracking branches.

`git fetch origin` would fetch all changes from the origin remote repository.

▶ `git fetch <remote>`

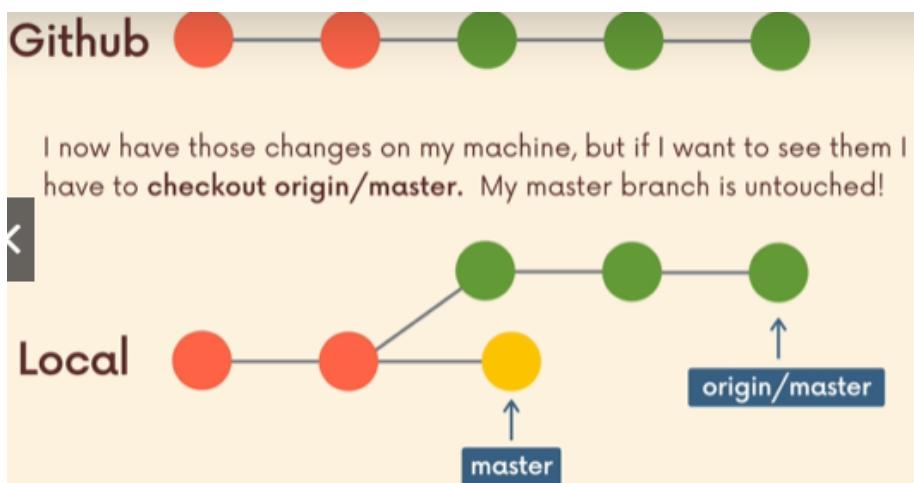
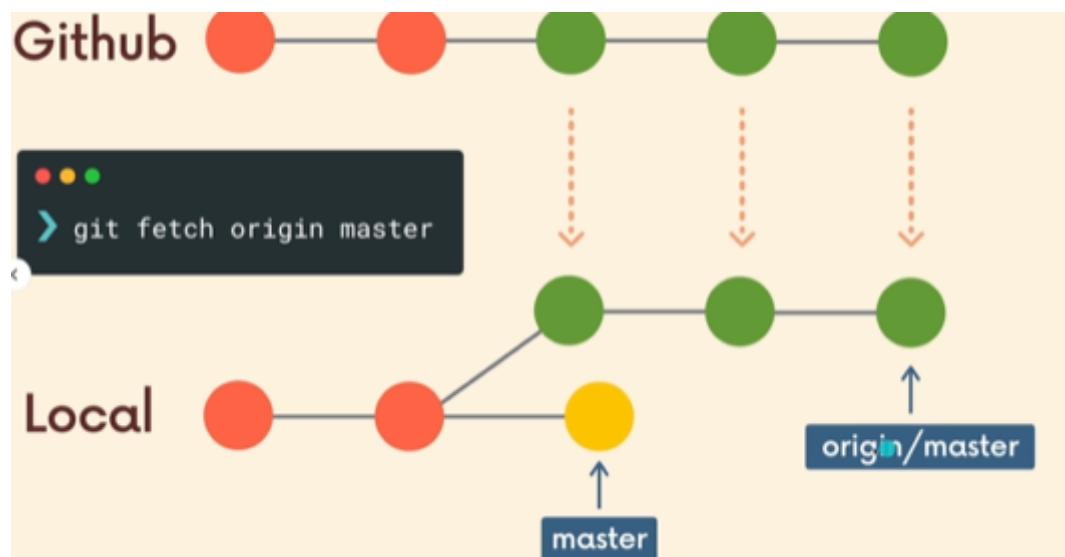
If not specified, `<remote>` defaults to `origin`

Git Fetch

We can also fetch a specific branch from a remote using `git fetch <remote> <branch>`

For example, `git fetch origin master` would retrieve the latest information from the master branch on the origin remote repository.

▶ `git fetch <remote> <branch>`



Demonstrating Git Fetch

Git Fetch

The `git fetch <remote>` command fetches branches and history from a specific remote repository. It only updates remote tracking branches.

`git fetch origin` would fetch all changes from the origin remote repository.

```
●●●
> git fetch <remote>
```

If not specified, `<remote>` defaults to `origin`

⌚ food had recent pushes 2 minutes ago

⌚ movies had recent pushes less than a minute ago

⌚ movies ▾ ⌚ 4 branches ⌚ 0 tags

```
remote-branches-demo > git status
On branch movies
Your branch is up to date with 'origin/movies'.

nothing to commit, working tree clean
```

movies

Git Fetch

The `git fetch <remote>` command fetches branches and history from a specific remote repository. It only updates remote tracking branches.

`git fetch origin` would fetch all changes from the origin remote repository.

```
●●●
> git fetch <remote>
```

If not specified, `<remote>` defaults to `origin`

Git Fetch

We can also fetch a specific branch from a remote using `git fetch <remote> <branch>`

For example, `git fetch origin master` would retrieve the latest information from the master branch on the origin remote repository.

```
●●●
> git fetch <remote> <branch>
```

```
remote-branches-demo > git fetch origin
remote-branches-demo >
```

movies
movies

```
remote-branches-demo > git status
On branch movies
Your branch is behind 'origin/movies' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

movies

```
nothing to commit, working tree clean
remote-branches-demo >
```

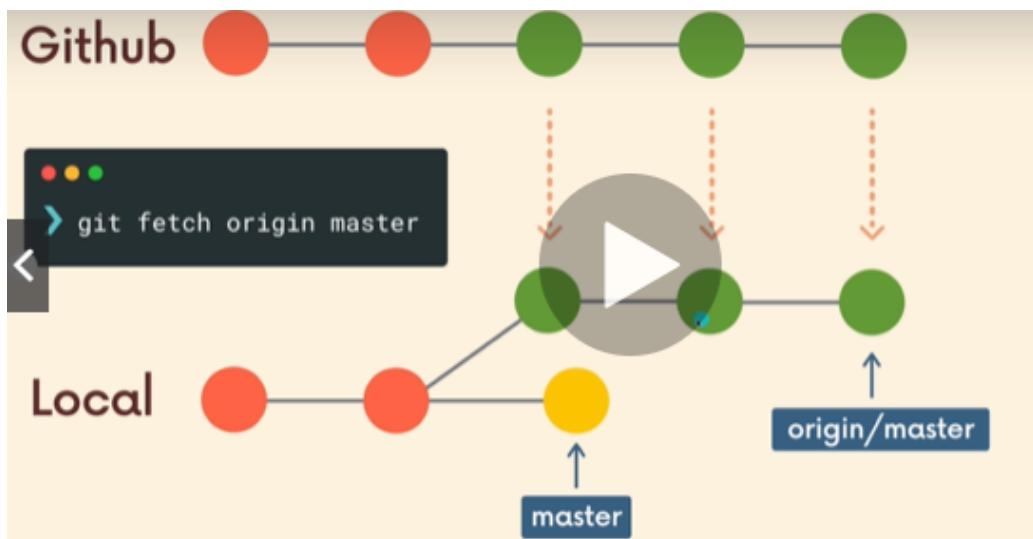
movies

```

remote-branches-demo > git status                               movies
On branch movies
Your branch is behind 'origin/movies' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
remote-branches-demo >                                     movies

```



```

remote-branches-demo > git log

```

```

commit 6c32e319078957ebbf9a0df486b1afbd9a3952e7 (HEAD -> movies)
Author: Colt Steele <5498438+Colt@users.noreply.github.com>
Date:   Mon Feb 1 14:05:17 2021 -0800

  Create bambi.txt

commit 335cbf5f7edb25a79cb666a93ee464a461089f4e
Author: Colt Steele <5498438+Colt@users.noreply.github.com>
Date:   Mon Feb 1 14:04:48 2021 -0800

  Create ghostbusters.txt

commit 2cfeb8a1982ad1f684a93b945ca95eacl901c8f0
Author: Colt Steele <5498438+Colt@users.noreply.github.com>
Date:   Mon Feb 1 14:04:32 2021 -0800

  Delete rose.txt

```

You don't see anything new.

```
remote-branches-demo > git checkout origin/movies
```

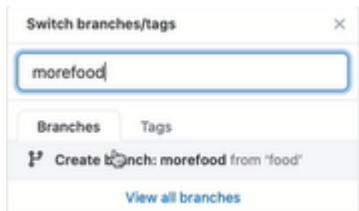
```
ghostbusters.txt  
tinkerbell.txt
```

```
commit d94a89ed48137475ecb89634c0aa31c44f4b2836 (HEAD, origin/movies)  
Author: Colt Steele <5498438+Colt@users.noreply.github.com>  
Date: Mon Feb 1 16:42:21 2021 -0800  
  
Create tinkerbell.txt  
  
commit 6c32e319078957ebbf9a0df486b1afbd9a3952e7 (movies)  
Author: Colt Steele <5498438+Colt@users.noreply.github.com>  
Date: Mon Feb 1 14:05:17 2021 -0800  
  
< Create bambi.txt  
  
commit 335cbf5f7edb25a79cb666a93ee464a461089f4e  
Author: Colt Steele <5498438+Colt@users.noreply.github.com>  
Date: Mon Feb 1 14:04:48 2021 -0800  
  
Create ghostbusters.txt
```

```
remote-branches-demo > git fetch origin food  
remote: Enumerating objects: 4, done.  
remote: Counting objects: 100% (4/4), done.  
< remote: Compressing objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), done.  
From https://github.com/Colt/remote-branches-demo  
 * branch food          -> FETCH_HEAD  
   ccda9ed..c4f3cb7  food    -> origin/food
```

```
remote-branches-demo > git checkout origin/food  
Note: switching to 'origin/food'.  
  
You are in 'detached HEAD' state. You can look around, r  
changes and commit them, and you can discard any commit  
you don't want.
```

Git fetch is not just about updating the particular branch that you are on and getting the latest. Also it might be that new branch was created.



```
III. Demonstrating Git Fetch  
origin/HEAD -> origin/main  
origin/fantasy  
origin/food  
origin/main  
origin/movies  
(END)
```

I don't see morefood.

```
remote-branches-demo > git fetch  
From https://github.com/Colt/remote-branches-demo  
 * [new branch]      morefood    -> origin/morefood
```

Git Pull

Pulling

git pull is another command we can use to retrieve changes from a remote repository. Unlike fetch, pull actually updates our HEAD branch with whatever changes are retrieved from the remote.

"go and download data from Github AND immediately update my local repo with those changes"

git pull = git fetch + git merge

update the remote tracking branch with the latest changes from the remote repository

update my current branch with whatever changes are on the remote tracking branch

git pull

To pull, we specify the particular remote and branch we want to pull using `git pull <remote> <branch>`. Just like with git merge, it matters WHERE we run this command from. Whatever branch we run it from is where the changes will be merged into.

`git pull origin master` would fetch the latest information from the origin's master branch and merge those changes into our current branch.

```
▶ git pull <remote> <branch>
```

pulls can result in merge conflicts!!

```
remote-branches-demo > git pull origin movies  
From https://github.com/Colt/remote-branches-demo  
 * branch      movies    -> FETCH_HEAD  
Updating 6c32e31..d94a89e  
Fast-forward  
 tinkerbell.txt | 37 ++++++  
 1 file changed, 37 insertions(+)  
< create mode 100644 tinkerbell.txt
```

movies



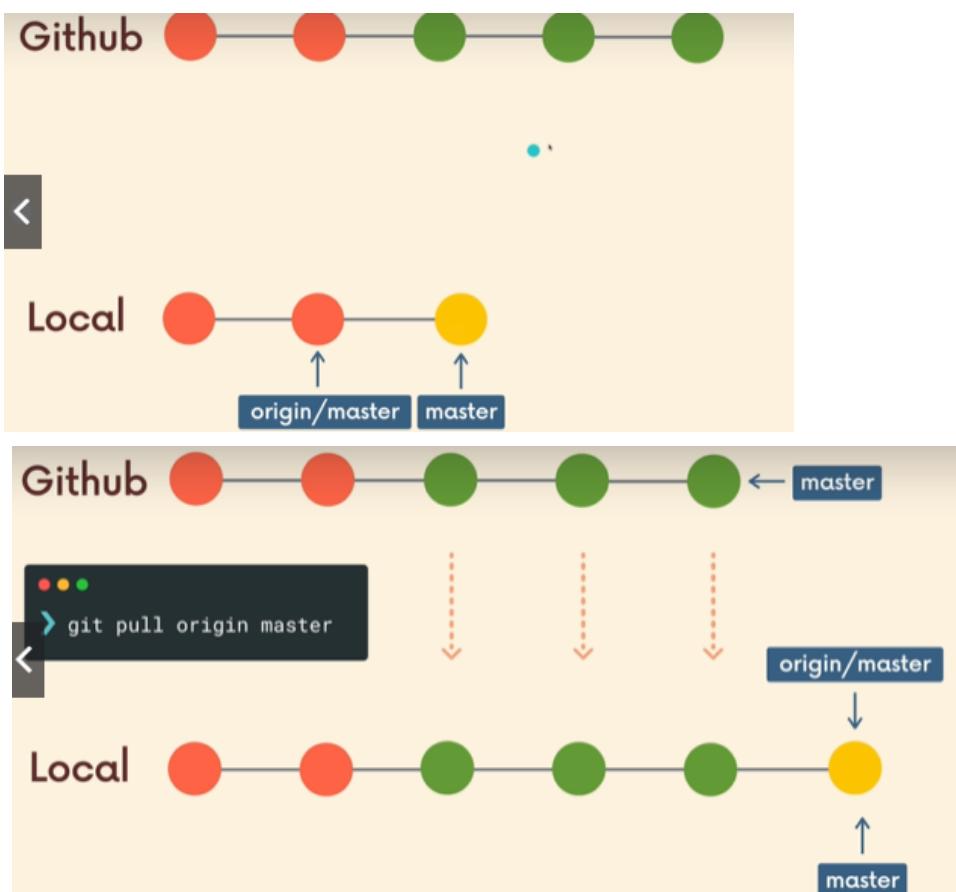
```
commit d94a89ed4af37475ecb89634c0aa31c44f4b2836 (HEAD -> movies, origin/movies)
Author: Colt Steele <5498438+Colt@users.noreply.github.com>
Date:   Mon Feb 1 16:42:21 2021 -0800

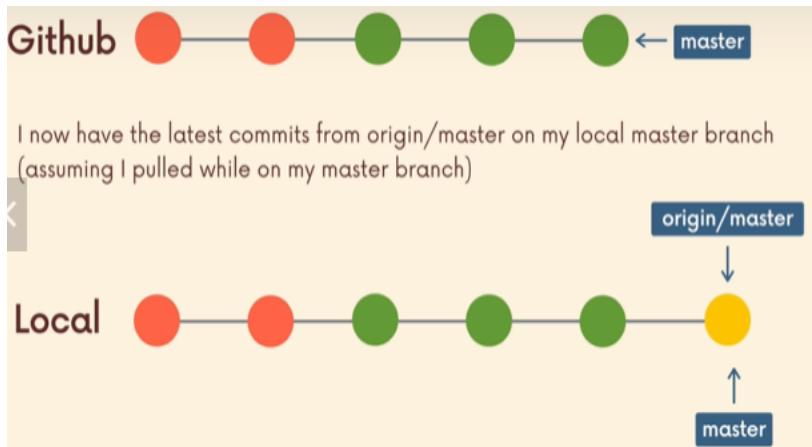
  Create tinkerbell.txt
```

Git Pull & Merge Conflicts

pulls can result in merge conflicts!!

Sometimes you may have some work locally, that is not on github and github might have some commits that you don't have locally. So when you pulled down there is gonna be conflicts.

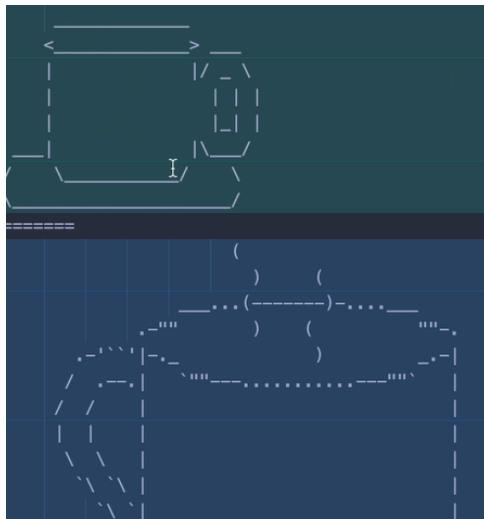




```
coffee.txt
nothing added to commit but untracked files present (use "git add" to track)
remote-branches-demo > git add coffee.txt
remote-branches-demo > git commit -m "add my coffee art"
```

Good practice, before you push anything on github, you might wanna pull down and see if theres any change.

```
remote-branches-demo > git pull origin food
From https://github.com/Colt/remote-branches-demo
 * branch      food      -> FETCH_HEAD
CONFLICT (add/add): Merge conflict in coffee.txt
Auto-merging coffee.txt
Automatic merge failed; fix conflicts and then commit the result.
remote-branches-demo >
```



```
remote-branches-demo > git commit -m "fix merge conflicts"
[food 8690895] fix merge conflicts
remote-branches-demo > git status
On branch food
Your branch is ahead of 'origin/food' by 2 commits.
(use "git push" to publish your local commits)
```

```

remote-branches-demo > git status
On branch food
Your branch is ahead of 'origin/food' by 2 commits.
  (use "git push" to publish your local commits)

    113. Git Pull & Merge Conflicts
commit 86908959ace092e7ef34f84f6a8995b31729ed4c (HEAD -> food)
Merge: dece76e c4f3cb7
Author: Colt Steele <5498438+Colt@users.noreply.github.com>
Date:   Mon Feb 1 17:07:44 2021 -0800

  fix merge conflicts

commit dece76e13c9c99be9f5be158be216be1edbb656d
Author: Colt Steele <5498438+Colt@users.noreply.github.com>
Date:   Mon Feb 1 17:06:10 2021 -0800
<
  add my coffee art

```

The reason its ahead, the add my coffee art was my first commit(which was local).

The fix merge conflicts when we pull the github commits and merge them with our locally committed changes.

```

remote-branches-demo > git push origin food
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.

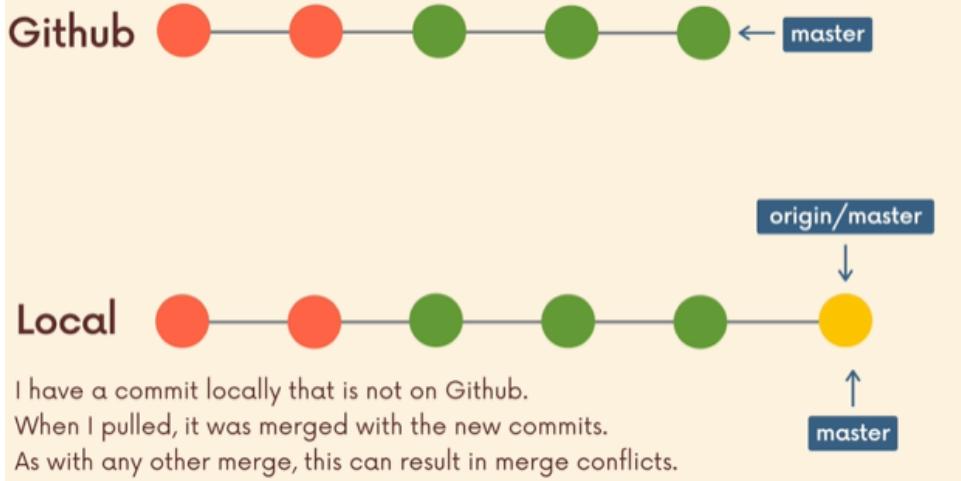
```

The screenshot shows a GitHub repository interface. At the top, there's a list of files:

- apple.txt**: Create apple.txt ↗
- banana.txt**: Create banana.txt
- coffee.txt**: fix merge conflicts
- popsicle.txt**: Create popsicle.txt

Below this, there's a commit history:

- fix merge conflicts** by **Colt** committed 1 minute ago
- add my coffee art** by **Colt** committed 3 minutes ago
- Create coffee.txt** by **Colt** committed 20 minutes ago

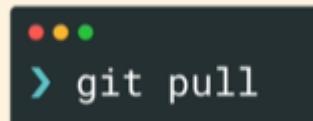


A shorter Syntax For Git Pull

An even easier syntax!

If we run `git pull` without specifying a particular remote or branch to pull from, git assumes the following:

- remote will default to origin
- branch will default to whatever tracking connection is configured for your current branch.

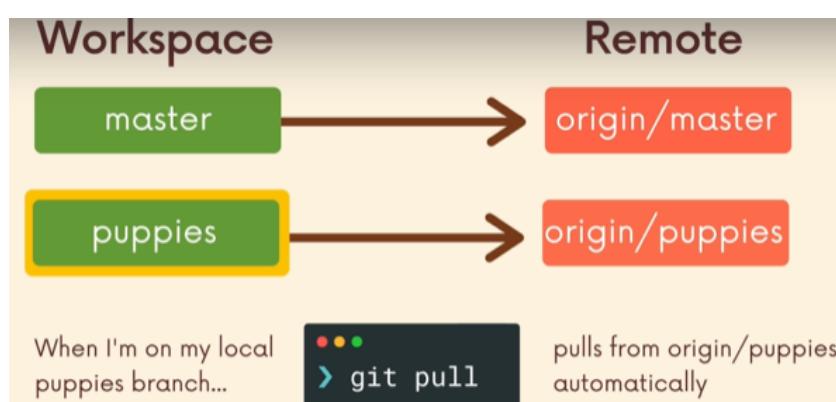


Note: this behavior can be configured, and tracking connections can be changed manually. Most people don't mess with that stuff

```
114. A Shorter Syntax For Git Pull:
fantasy
* food
  main
  movies
(END)

remote-branches-demo > git branch
remote-branches-demo > git pull origin food
```

We never setup the tracking connection.



```

remote-branches-demo > git branch
remote-branches-demo > git pull
Updating 8690895..ae2d02f
Fast-forward
  tea.txt | 20 ++++++
  1 file changed, 20 insertions(+)
  create mode 100644 tea.txt
remote-branches-demo > ls
apple.txt  banana.txt  coffee.txt  popsicle.txt tea.txt
remote-branches-demo > git log
remote-branches-demo > git branch
remote-branches-demo > git switch fantasy
Switched to branch 'fantasy'
Your branch is behind 'origin/fantasy' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

remote-branches-demo > git pull
Updating 463f23f..edd407b
Fast-forward
  gryphon.txt | 20 ++++++
  1 file changed, 20 insertions(+)
  create mode 100644 gryphon.txt

```

git fetch

- Gets changes from remote branch(es)
- Updates the remote-tracking branches with the new changes
- Does not merge changes onto your current HEAD branch
- Safe to do at anytime

git pull

- Gets changes from remote branch(es)
- Updates the current branch with the new changes, merging them in
- Can result in merge conflicts
- Not recommended if you have uncommitted changes!

Github Grab bags Odds and ends

Readme.md , Md stands for mark down.

Github Pages, a very nice tool from Github that allows us it helps us to easily host static web pages from our repositories.

Github Repo Visibility: Public vs Private

<h2>Public Repos</h2> <p>Public repos are accessible to everyone on the internet. Anyone can see the repo on Github</p>	<h2>Private</h2> <p>Private repos are only accessible to the owner and people who have been granted access.</p>
---	---

The screenshot shows the GitHub repository settings page. At the top, there are tabs for Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The Settings tab is active. Below the tabs, there are two buttons: Options and Settings. The main area is titled "Danger Zone". It contains several sections with destructive actions:

- Change repository visibility**: This repository is currently public. A "Change visibility" button is present.
- Transfer ownership**: Transfer this repository to another user or to an organization where you have the ability to create repositories. A "Transfer" button is present.
- Archive this repository**: Mark this repository as archived and read-only. An "Archive this repository" button is present.
- Delete this repository**: Once you delete a repository, there is no going back. Please be certain. A "Delete this repository" button is present, along with links to "Activate WI" and "Go to Settings".

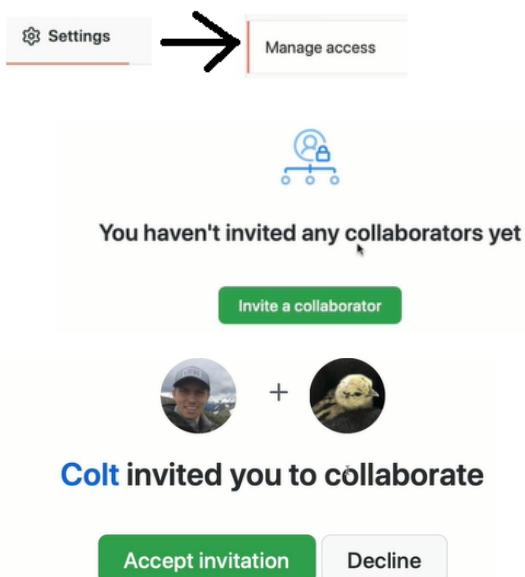
The screenshot shows a confirmation dialog for making a repository private. It features a large circular "X" button in the center. To the left, there is a warning message: "⚠ Warning: this is a potentially destructive action." Below the button, there are two radio button options:

- Make public**: This repository is currently public.
- Make private**: Hide this repository from the public.

Details about the private action are listed:

- You will permanently lose:
 - All stars and watchers of this repository.
 - All pages published from this repository.
- You can [upgrade your plan](#) to also avoid losing access to:
 - Codeowners functionality.
 - Any existing wikis.
 - Pulse, Contributors, Community, Traffic, Commits, Code Frequency and Network on the Insights page.
 - Draft PRs

Adding Github Collaborators



Owners of bookish-disco will be able to see:

- Your public profile information
- Certain activity within this repository
- Country of request origin
- Your access level for this repository
- Your IP address

Github Collaboration Demo

```
commit 4c595ct924262dc//8bc73/c21  
b4 (HEAD -> main)  
Author: Stevie Chicks <78455119+S  
users.noreply.github.com>  
Date:   Wed Feb 3 18:03:54 2021 -
```

add two artists

```
commit 47c68c41c118bb409d028641e3  
c4 (origin/main, origin/HEAD)  
Author: Colt Steele <5498438+Colt
```

stevie will push the changes to colt github. Since stevie is a collaborator.

```
bookish-disco > git pull origin main    main  
From github.com:Colt/bookish-disco  
 * branch            main      -> FETCH_HEAD  
Updating 47c68c4..4c595cf  
Fast-forward  
 artists.txt | 2 ++  
 1 file changed, 2 insertions(+)
```

What are READMEs?

Initialize this repository with:
Skip this step if you're importing an existing repository.

[Add a README file](#)

This is where you can write a long description for your project. [Learn more](#).

[Add .gitignore](#)

Choose which files not to track from a list of templates. [Learn more](#).

[Choose a license](#)

A license tells others what they can and can't do with your code. [Learn more](#).

READMEs

A README file is used to communicate important information about a repository including:

- What the project does
- How to run the project
- Why it's noteworthy
- Who maintains the project

What your project does, whether you're working on some API, or some Web Applications, video games. Even if you're just government using GitHub to make law legislativley.

Whatever your repos is, the readme file should include information about what the project does, what its for, who is it for. How to get started with ur project. How to run and install dependencies if necessary. Why its noteworthy. Who maintains the project, how to collaborate the project, how to get help.

Not everything needs to be there. But if there is important info that you need to convey.

Somebody comment to your project for the first time, then put it in the readme file.

READMEs

It will render

If you put a README in the root of your project,
GitHub will recognize it and automatically display
it on the repo's home page.

	.gitignore
	CODE_OF_CONDUCT.md
	CleanArchitecture.sln
	LICENSE
	README.md
	docker-compose.dcproj
	docker-compose.override.yml
	docker-compose.yml



.NET Core passing nuget 1.1.1 downloads

This is a solution template for creating a Single Page App (SPA) Clean Architecture. Create a new project based on this template installing and running the associated NuGet package (see Get

List of technologies.

Technologies

- ASP.NET Core 5
- Entity Framework Core 5
- Angular 10
- MediatR
- AutoMapper
- FluentValidation
- NUnit, FluentAssertions, Moq & Respawn
- Docker

How to get started

How to configure

How to configure db

How to migrate.

Support

Getting Started

license

The easiest way to get started is to install the [NuGet package](#)

Support

If you are having problems, please let us know by [raising a new issue](#).

License

This project is licensed with the [MIT license](#).

Link for above: [click](#)

Link: [Click](#)

spaCy models



This repository contains [releases](#) of models for the [spaCy](#) NLP library. For more info on how to download, install and use the models, see the [models documentation](#).

⚠ Important note: Because the models can be very large and consist mostly of binary data, we can't simply provide them as files in a GitHub repository. Instead, we've opted for adding them to [releases](#) as `.whl` and `.tar.gz` files. This allows us to still maintain a public release history.

Quickstart

To install a specific model, run the following command with the model name (for example `en_core_web_sm`):

```
python -m spacy download [model]
```

- [spaCy v3.x models directory](#)

Readme file always place at root of your project always. Sometimes .git folder.

Link: [Click](#)

React

• [license MIT](#) [npm v17.0.2](#) [circleci passing](#) [PRs welcome](#)

React is a JavaScript library for building user interfaces.

- **Declarative:** React makes it painless to create interactive UIs. Design simple application, and React will efficiently update and render just the right components. Declarative views make your code more predictable, simpler to understand.
- **Component-Based:** Build encapsulated components that manage their own complex UIs. Since component logic is written in JavaScript instead of XML, it's easier to reuse components through your app and keep state out of the DOM.
- **Learn Once, Write Anywhere:** We don't make assumptions about the rest of your stack. You can use React to quickly prototype mobile apps, and then reuse your code to build desktop apps and power mobile apps using [React Native](#).

README.md

READMEs are Markdown files, ending with the .md extension. Markdown is a convenient syntax to generate formatted text. It's easy to pick up!

A Markdown Crash Course

Link:[Markdown webpage](#)

Markdown is a text-to-HTML conversion tool for web writers.

Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).

File browser screenshot showing a README.md file. The file content is visible in the preview pane, showing headings like React, Installation, Documentation, Examples, Contributing, Code of Conduct, Contributing Guide, Good First Issues, and License. A blue bar highlights the word 'React'.

← how much structure is important.

Code editor screenshot showing a README.md file. The content includes a circular diagram with the words 'EXPLOSION' around the top and 'NOISE' in the center, and the text 'nd use' at the bottom. The editor has buttons for Raw, Blame, and file operations.

<pre>"spacy_models" This repository contains [releases](https://github.com/explosion/spacy-models/releases) of models for the [spaCy](https://github.com/explosion/spaCy) NLP library. For more information on how to download, install and use the models, see the [models documentation](https://spacy.io/usage/models).</pre>			
Raw			
> **⚠ Important note:** Because the models can be very large and consist of binary data, we can't simply provide them as files in a GitHub repo. Instead, we've opted for adding them to [releases](https://github.com/explosion/spacy-models/releases) as `.whl` and `.tar.gz` files. This allows us to still maintain a public release history.			
 Update README.md	4 years ago	13	> Instead, we've opted for adding them to [releases](https://github.com/explosion/spacy-models/releases) as `.whl` and `.tar.gz` files. This allows us to still maintain a public release history.
 Update docs for spaCy v3 (#105)	3 months ago	14	> [releases](https://github.com/explosion/spacy-models/releases) as `
		15	> `.tar.gz` files. This allows us to still maintain a public release history.
 Update README.md	4 years ago	16	
		17	## Quickstart
Blame			

Link: [Markdown-it](#)

markdown-it demo

html xhtmlOut breaks linkify typographer highlight language- CommonMark strict

```
---
__Advertisement :)__

- __[pica](https://nodeca.github.io/pica/demo/)__ - high quality and fast
image
  resize in browser.
- __[babelfish](https://github.com/nodeca/babelfish/__ - developer
friendly
  i18n with plurals support and easy syntax.

You will like those projects!

---

# h1 Heading 8-
## h2 Heading
```

clear [permalink](#)

Advertisement 😊

- **pica** - high quality and fast image resizing
- **babelfish** - developer friendly i18n with plurals support

You will like those projects!

h1 Heading 😎

h2 Heading

```
# i am a heading!
```

```
# h1 Heading 8-
## h2 Heading
### h3 Heading
#### h4 Heading
##### h5 Heading
###### h6 Heading
```

```
<h1>i am a heading!</h1>
<h1>h1 Heading 😎</h1>
<h2>h2 Heading</h2>
<h3>h3 Heading</h3>
<h4>h4 Heading</h4>
<h5>h5 Heading</h5>
<h6>h6 Heading</h6>
```

h1 Heading 😎

h2 Heading

h3 Heading

```
``` js
console.log("HELLO!");
console.log("HELLO!");
```

```

```
console.log("HELLO!");
console.log("HELLO!");
```

h1 Heading 😎

h2 Heading

Reddit use markdown.

Adding a ReadMe To a project

Alot of time, you working on a project. You don't expect for anybody to ever see it.

They are the probably the one thing, that a recruiter or a non technical person would see, if they are looking at your projects.

The screenshot shows a GitHub repository named "Bookish Disco". The repository has two files: "README.md" and "artists.txt". The "README.md" file contains the following content:

```
1 # Boo
2 pro
```

The "Preview README.md" tab shows the rendered markdown:

Bookish Disco

Bookish Disco is a repository name randomly generated by Github. This project does nothing. It is purely a demo for a course on Git/Github.

Recent activity shows a user named "Colt" adding the README file.

Creating Github Gists

Github Gists

Github Gists are a simple way to share code snippets and useful fragments with others. Gists are much easier to create, but offer far fewer features than a typical Github repository.

The screenshot shows a list of four gists created by a user named Colt:

- _app.js**: Secret, Last active 2 years ago. Contains React code for a MyApp component.
- hook.js**: Secret, Created 2 years ago. Contains code for a useLocalStorageReducer hook.
- hello.js**: Secret, Created 2 years ago. Contains a single line of code: "sadasd".
- app.css**: Secret, Created 2 years ago. Contains CSS for a header element.

Signed in as IDK9911

Your gists

Starred gists

Help

Your GitHub profile

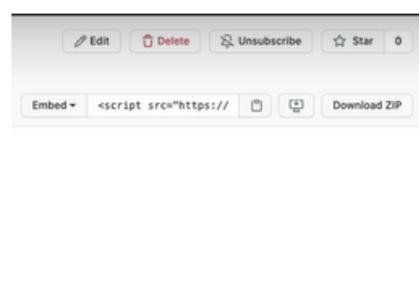
Sign out

The screenshot shows a single gist titled "nav_exercise.html" with the following content:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnE" crossorigin="anonymous">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.3.1/css/all.css" integrity="sha384-mzrmE5qonljUremFsqc015B4QjzG8RlY5PbI7" crossorigin="anonymous">
```



The ability to leave comments, I can personally delete this file, edit this file. I can download it and share with people, fork it and star it

All gist

Kordietta / Squirrels and nuts
Created 16 seconds ago
N squirrels found K nuts and decided to divide them equally. Find how many nuts each squirrel gets.
after each of the squirrels takes the equal amount of nuts.

```
1 import java.util.Scanner;
2
3 class Main {
4     public static void main(String[] args) {
5
6         Scanner scanner = new Scanner(System.in);
7
8         int squirrels = scanner.nextInt();
9         int nuts = scanner.nextInt();
10
11    }
12 }
```

You can see gist what other people have made.

So they range from mark down files.

We use git for teaching and share stuff.



You don't have any gists yet.

Your public gists will show up here on your profile. [Create a gist](#) to get started.

Gist description...

Filename including extension... Spaces 2 No wrap

1

Add file

Create secret gist

✓ Create secret gist
Secret gists are hidden by search engines but visible to anyone you give the URL to.

Create public gist
Public gists are visible to everyone.

[oanhnn / laravel-csp](#)

app.js

```
1 console.log("hello")
2
```

Introducing Github Pages

pages.github.com

Link: [Click](#)

static sites

Github Pages is a hosting service for static webpages, so it does not support server-side code like Python, Ruby, or Node. Just HTML/CSS/JS!

gh pages

Github Pages are public webpages that are hosted and published via Github. They allow you to create a website simply by pushing your code to Github.

[Github Pages](#)

Websites for you and your projects.

Hosted directly from your [GitHub repository](#). Just edit, push, and your changes are live.

For static web sites, for portfolio websites or documentation website for your repositories. If you build a js game , you can use github pages to host that.

You don't expect github pages for high capacity web traffic. Its not really meant for that. Its free and its built for github.

ovolve.github.io/2048-AI/



User Site

You get one user site per Github account. This is where you could host a portfolio site or some form of personal website. The default url is based on your Github username, following this pattern: `username.github.io` though you can change this!

Project Sites

You get unlimited project sites! Each Github repo can have a corresponding hosted website. It's as simple as telling Github which specific branch contains the web content. The default urls follow this pattern: `username.github.io/repo-name`

Github Pages Demo

gh pages

Github Pages are public webpages that are hosted and published via Github. They allow you to create a website simply by pushing your code to Github.

main ▾ 1 branch 0 tags Go to file Add file ▾ Code ▾

Colt add some chicken breeds 6f1a778 26 seconds ago 2 commits

chickens.txt add some chicken breeds 26 seconds ago

Help people interested in this repository understand your project by adding a README. Add a README

Now , if I have to setup github pages website, all that I have to do is select some branch, I am gonna tell github contains a website.

Its gonna look for a file name “index.html”

GitHub Pages

Pages settings now has its own dedicated tab! [Check it out here!](#)

GitHub Pages

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

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

[None ▾](#)

[Save](#)

Theme Chooser

Select a theme to publish your site with a Jekyll theme using the `gh-pages` branch. [Learn more.](#)

[Choose a theme](#)

You can setup your website on any branch whether its main or not.

Your site is ready to be published at <https://colt.github.io/chickens/>.

source

our GitHub Pages site is currently being built from the `gh-pa`

[Branch: gh-pages ▾](#)

[/ \(root\) ▾](#)

[Save](#)

Select branch

[Select branch](#)

main

gh-pages

None

Select folder

/ (root)

/docs



gh-pages are standard default names.

Git Collaboration Workflows

Different ways of configuring repositories, different ways of working on branches,
When to create branches, how to merge changes, how to ask to merge changes
How to get permission.

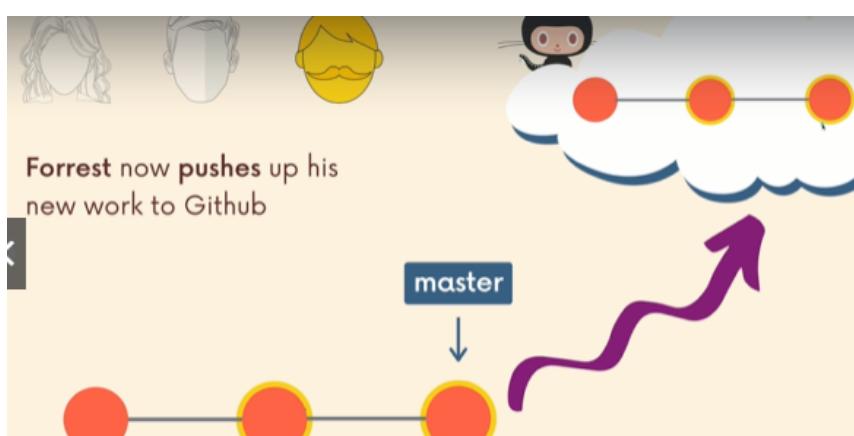
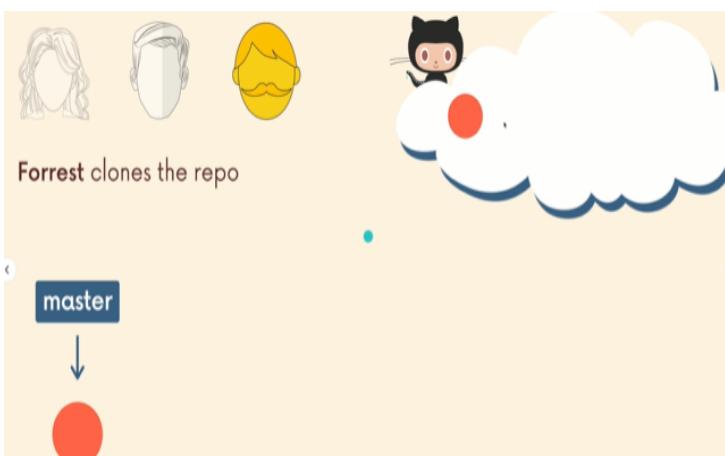
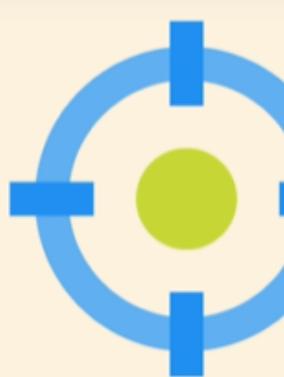
The Pitfalls of a Centralized Workflows

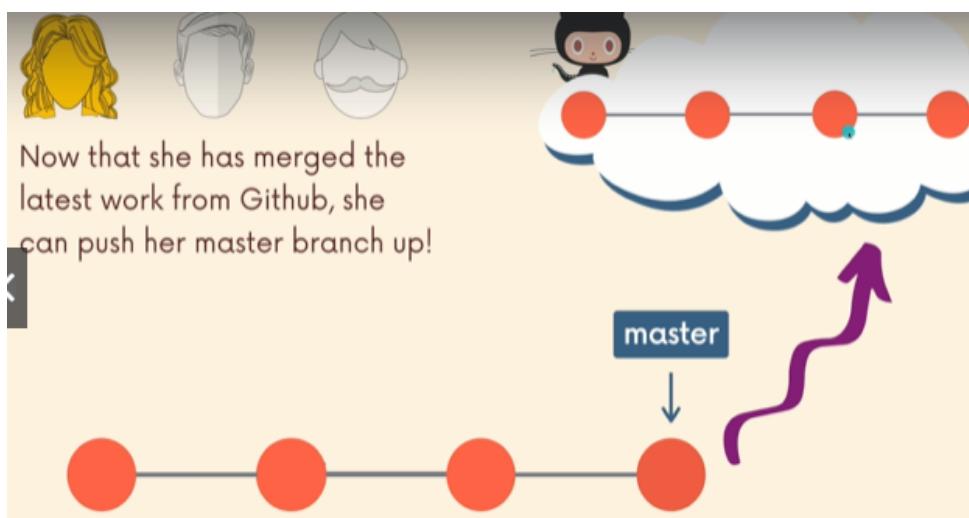
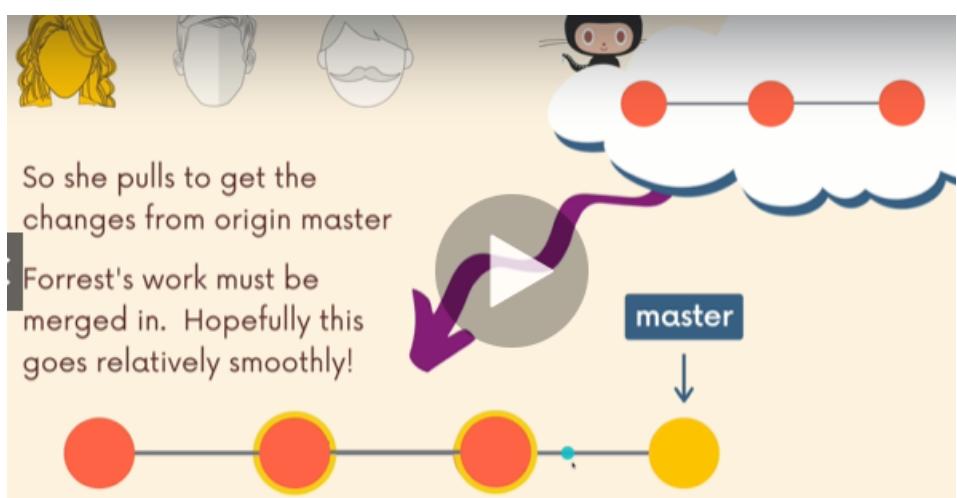
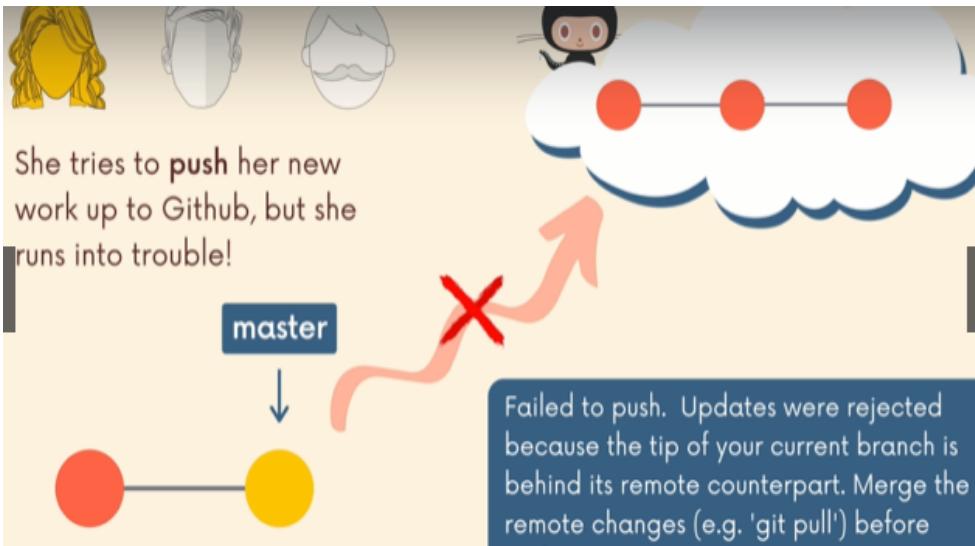
Centralized Workflow

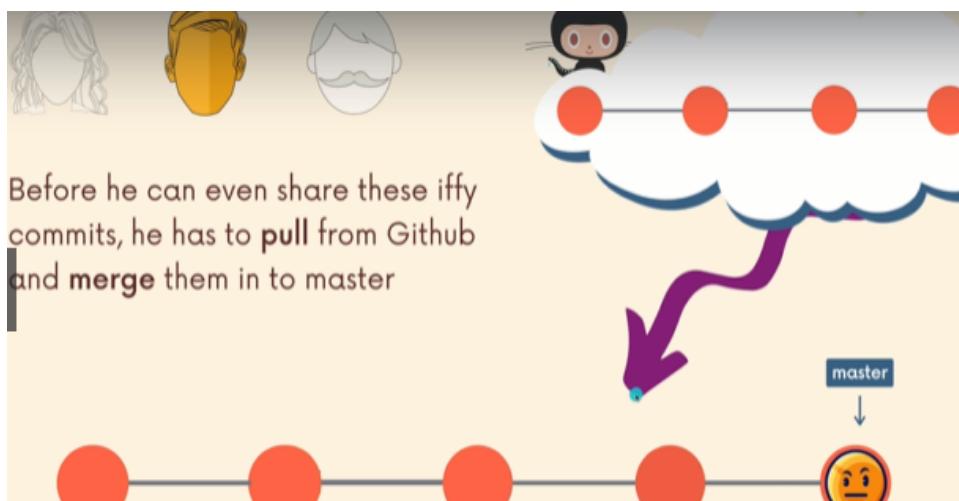
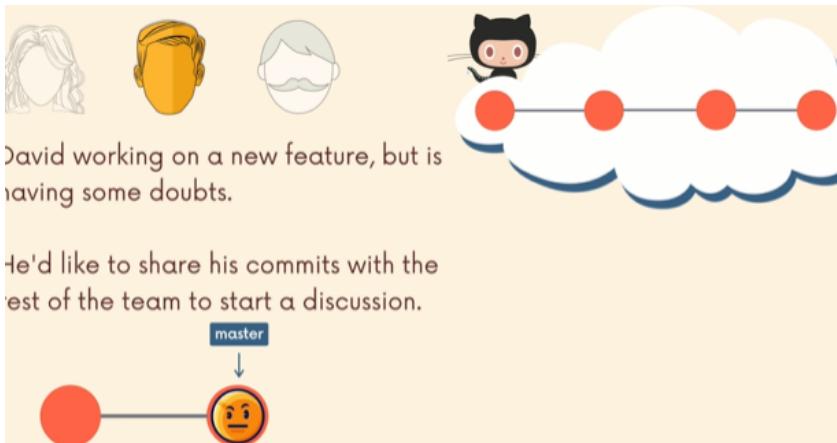
AKA Everyone Works On Master/Main
AKA The Most Basic Workflow Possible

The simplest collaborative workflow is to have everyone work on the master branch (or main, or any other SINGLE branch).

It's straightforward and can work for tiny teams, but it has quite a few shortcomings!







if this is incomplete, so everyone has a incomplete code. He broke the codebase.

The Problem

While it's nice and easy to only work on the master branch, this leads to some serious issues on teams!

- Lots of time spent resolving conflicts and merging code, especially as team size scales up.
- No one can work on anything without disturbing the main codebase. How do you try adding something radically different in? How do you experiment?
- The only way to collaborate on a feature together with another teammate is to push incomplete code to master. Other teammates now have broken code...

```
centralized-workflow > git push origin main
To github-stevie:Colt/centralized-workflow.git
  ! [rejected]          main -> main (fetch first)
error: failed to push some refs to 'git@github-stevie:Colt/centralized-workflow.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first
integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Above is Central workflow problem. Stevie is trying to push broken code in the master branch while his fellow coworker has already made commits on the remote repo. So stevie is behind some “n” commits. In order to show his coworkers and start discussion he has to merge “source of truth” or “official branch”.

Leading to “broken code” at “source of truth”

```
demonstration
centralized-workflow > git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done
```

```
centralized-workflow > git status      main
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)
```

```
demonstration
centralized-workflow > git push origin main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 16 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 2.14 KiB | 2.14
```

Next day, colt receive changes from stevie to help with his code.

He has changes in his local repositories which aren't committed.

modified: index.html

no changes added to commit (use "git add" and /or "git commit -a")

Activate Windows
Go to Settings to activate Windows.

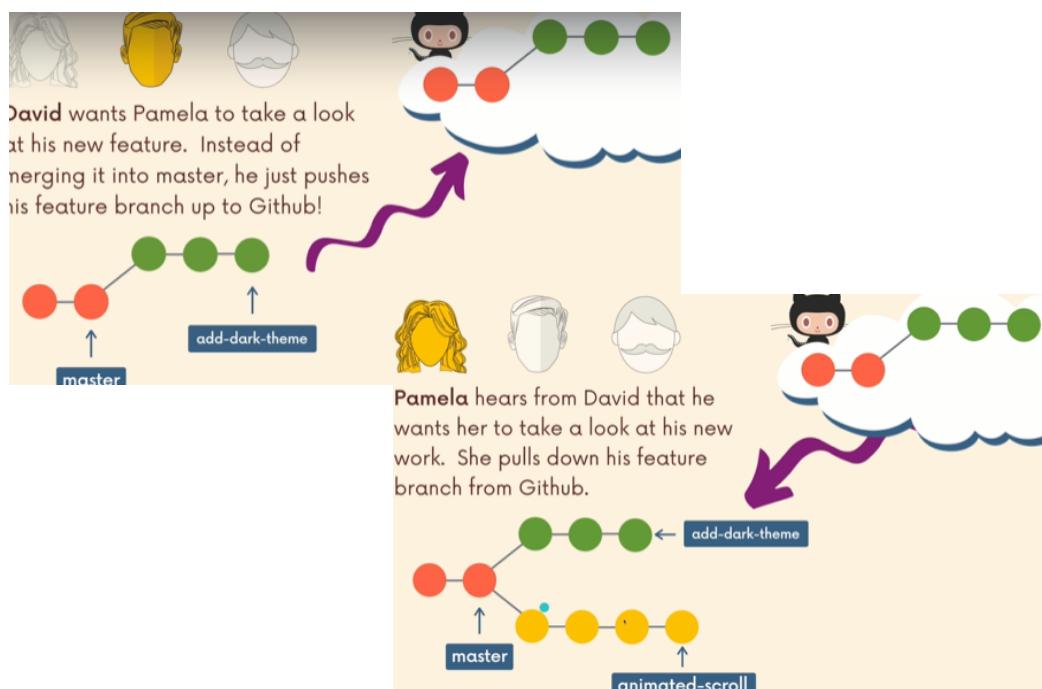
He needs to stash those changes, since he doesn't wanna commit yet.

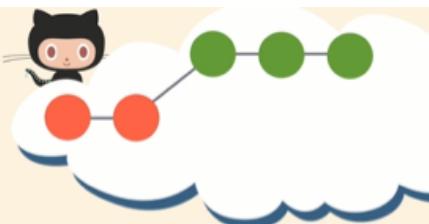
The All-important Feature Branch Workflow

Feature Branches

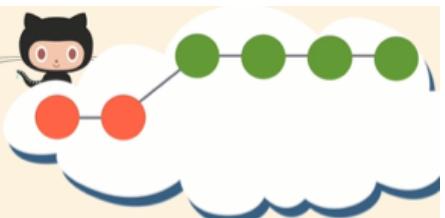
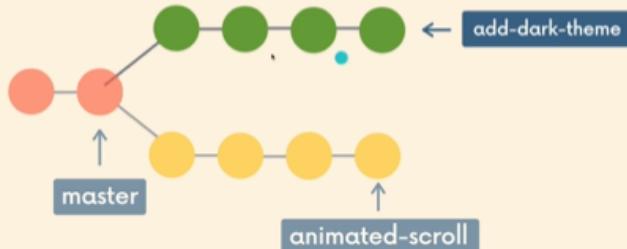
Rather than working directly on master/main, all new development should be done on separate branches!

- Treat master/main branch as the official project history
- Multiple teammates can collaborate on a single feature and share code back and forth without polluting the master/main branch
- Master/main branch won't contain broken code (or at least, it won't unless someone messes up)



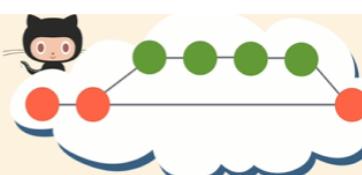
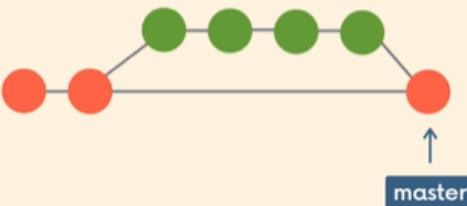


Pamela takes a look at the code and makes a couple improvements of her own. She adds/commits on the same feature branch.

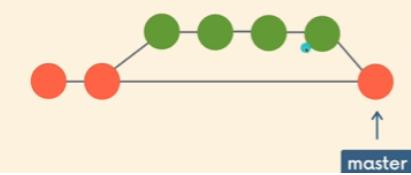


David decides he is happy with the new feature, so he merges it into master!

*At a real company, you don't just decide you are "happy with" a feature. There are mechanisms for code approval and merging we'll discuss shortly!



David pushes up the updated master branch to Github. The others can now pull down the changes.



we dont merge broken stuff in the main branch.

Unlike the centralized workflow, so everyone works on a branch.

You can have multiple branches too. It's not necessarily like Pamela's branch and David's branch. It's more one branch per feature or bugfix.

Branches are feature oriented.

we share those branches, collaborate on branches.

And keep our master or main branch as our source of truth. As a stable, always working and functioning , non-broken branch in our project.

these are pattern that org uses. its just a name, it could be called just navbar

Feature Branch Work Flow

We havent really talk about naming feature branch in general. For e.g

feat/navbar <!-- By feat he meant features

refactor/navbar

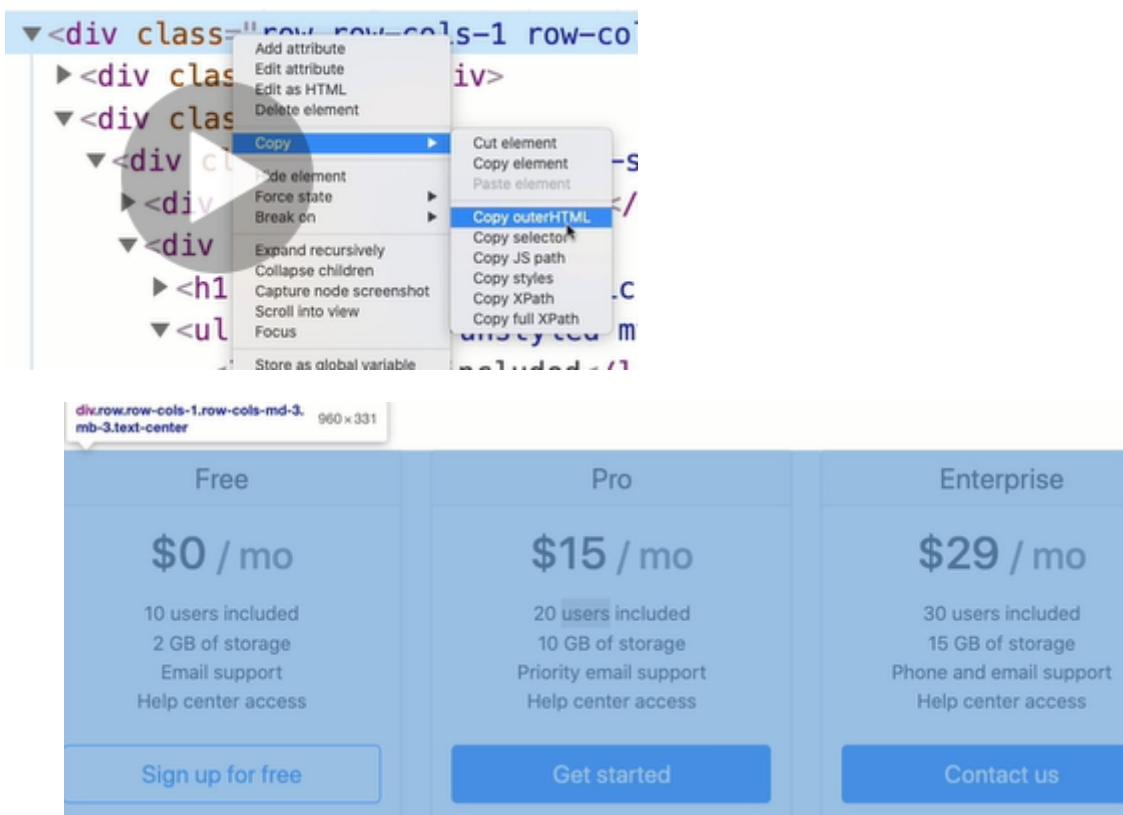
bug/navbar

fix/navbar

```
feature-branch-workflow > git pull origin main
From github.com:Colt/feature-branch-workflow
 * branch           main      -> FETCH_HEAD
Already up to date.
```

Its always a good idea, to see is there any new commit. If there is, you can start your day with fast forward.

```
feature-branch-workflow > git switch -c pricing-table
Switched to a new branch 'pricing-table'
```



its a project with pricing model, this isn't github

Advantage of feature branch is that in contrast to centralized workflow, we wont have to merge code with others on the main branch(sot) just so we can collaborate who are working with us to fix the bug.

Everytime you make a feature branch, you don't have to push into github.

Alot of time , you wont do that. Unless you have a particular reason to share it to somebody.

If you look at organization repositories, they are not gonna be 1000s of feature branch. Even though there have been 1000s of feature branches over a couples of months.

Alot of time, we remove them.

Alot of time, we merge it to master or main sometime or delete the branch when you done.

```
feature-branch-workflow > git fetch origin  
remote: Enumerating objects: 5, done.
```

To see if ,Has anything changed.

```
Unpacking objects: 100% (3/3), done.  
From github.com:Colt/feature-branch-workflow  
 * [new branch] navbar -> origin/navbar
```

```
h Workflow Demo  
origin/HEAD -> origin/main  
origin/main  
origin/navbar  
origin/pricing-table
```

```
feature-branch-workflow > git checkout origin/navbar
```

```
HEAD is now at fba59a8 add broken navbar  
feature-branch-workflow > git switch -  
Previous HEAD position was fba59a8 add broken navbar  
Switched to branch 'pricing-table'
```

i need to leave detached head.

```
feature-branch-workflow > git branch pricing-table  
feature-branch-workflow > git switch navbar  
  
feature-branch-workflow > git commit -m "fix navbar bug"  
  
feature-branch-workflow > git push origin navbar  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.
```

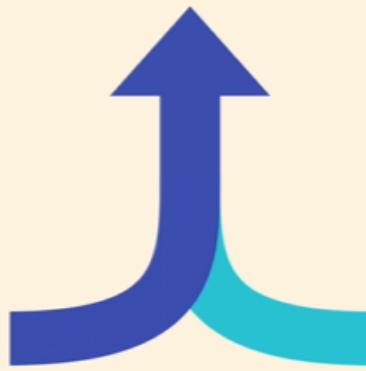
we have a lot of options how we intend to do this, one is merge at will. whenever anybody wants to merge their changes in, just go for it.

Merging Feature Branches

Merging In Feature Branches

At some point new the work on feature branches will need to be merged in to the master branch!
There are a couple of options for how to do this...

1. Merge at will, without any sort of discussion with teammates. JUST DO IT WHENEVER YOU WANT.
2. Send an email or chat message or something to your team to discuss if the changes should be merged in.
3. Pull Requests!



Delete the branch after merge if we don't need it anymore.

Introducing pull request

Pull Requests

Pull Requests are a feature built in to products like Github & Bitbucket. They are not native to Git itself.

They allow developers to alert team-members to new work that needs to be reviewed. They provide a mechanism to approve or reject the work on a given branch. They also help facilitate discussion and feedback on the specified commits.

"I have this new stuff I want to merge in to the master branch...what do you all think about it?"

So if you are in a company where the stakes are high, where you can't expect to screw up something in the main or master branch.

What if somebody did a bad commit or something that breaks the main branch. At the very least, you've got 100 more developers who now have a bad version of that main branch and

they have to get rid of commit or reset it.

It's just annoying.

You want to make sure that whatever you are merging in or whoever is doing the merging has some discussion around it, has some approval.

Maybe changes are made at the request of somebody else.

There's some code review process involved and pull requests are integral part of the process.

especially in larger teams with lots of collaborators and features. If there is no testing going on beforehand, if there is no discussion or conversation before hand, that could get way out of control. And you would never do that on a real product with users and things at stake.

so another way is to merge your own branches in, but only after you've discussed. You've sent an email or chat message or some sort of conversation with your team and somehow that whether you should merge those changes in or not.

3rd option is pull request:

we've talked about merging feature branches to the default branch/(sot)/main and how much it has made our life easier. But then at some point somebody has to merge those changes in from the feature branch into main branch.

so rather its better to have some code review process before someone merges their bad commit. so to protect the branch and keep it functioning, working at all times. we have PR.

So u have to make sure, whatever u are merging in, or whoever is doing the merging have some discussion around, it has some approval.

Maybe changes are made at the request of somebody else. Basically taking in feedback. So PR are not a native git feature. They are built on top of platforms like Github and bitbucket and some others.

Not exclusive to github and not 100% part of the git itself.

PR:

inform other collaborators that they have new work that needs to be reviewed before integrated.

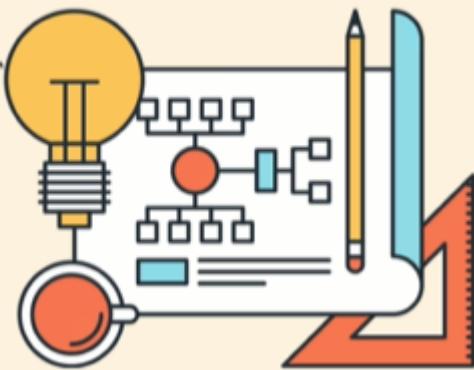
so we have discussion, feedback, update the work on the same PR.

So, Hey, everybody on my team, i have this new branch with this feature i completed or i think its complete and its on github. Im making this PR, can u take a look and somebody merge it in if its good or tell me if its not good.

The main branch is protected usually, u have to make a pull request.

The Workflow

1. Do some work locally on a feature branch
2. Push up the feature branch to Github
3. Open a pull request using the feature branch just pushed up to Github
4. Wait for the PR to be approved and merged. Start a discussion on the PR. This part depends on the team structure.



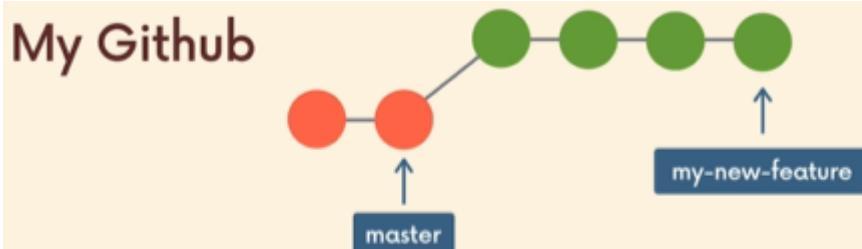
Maybe multiple ppl have to approve.

15.1. Introducing Pull Requests

I push my feature branch up to Github, so that I can open a Pull Request

master

my-new-feature



my-new-feature had recent pushes less than a minute ago

Compare & pull request

my-new-feature 5 branches 0 tags

Go to file Add file Code

This branch is 2 commits ahead of master.

Pull request Compare

Colt Steele and Colt Steele add another new file

anotherNewFile.txt add another new file 32 seconds ago

newfeature.txt add new feature 1 minute ago

playlist.txt merge 5 days ago

I click the PR button

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub Pull Request interface. At the top, there are dropdown menus for 'base: master' and 'compare: my-new-feature'. A green checkmark icon followed by the text 'Able to merge. These branches can be automatically merged.' is displayed. Below this, a user profile picture and the branch name 'My new feature' are shown. There are 'Write' and 'Preview' tabs, and a large text area for comments with a play button icon in the center. Below the comment area is a file upload section with the placeholder 'Attach files by dragging & dropping, selecting or pasting them.' A green 'Create pull request' button is located at the bottom right.

I am requesting my-new-feature to merge into master.

Then i need to provide some information, i need to come up with what my pull request is. Whats the reason for it. What does it includes. Depending on the company and organization. Theres gonna be pretty strict rules for what goes in here.

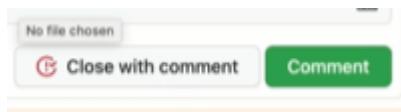
The screenshot shows a comment on a GitHub pull request. The comment text reads: 'Hey Colt, this is on the right track but it needs a bit more work before it's ready to be merged in. Please fix x & y and z and then report back. Add those commits to this PR. Ping me when it's ready. Thanks!' Below the comment is a file upload section with the placeholder 'Attach files by dragging & dropping, selecting or pasting them.' and a note 'No file chosen'. At the bottom right of the comment area are 'Close with comment' and 'Comment' buttons.

My boss leaves some feedback for me. She asks me to make a couple changes before she merges the pull request.

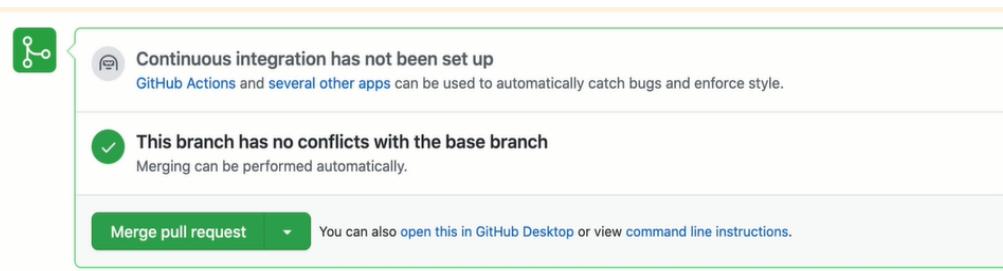
I as an employee dont have the ability to merge directly into master in github. I am supposed to make a pull request.

You can ask help of others in ur teams who are reviewing the PR. U can mention them in the comments.

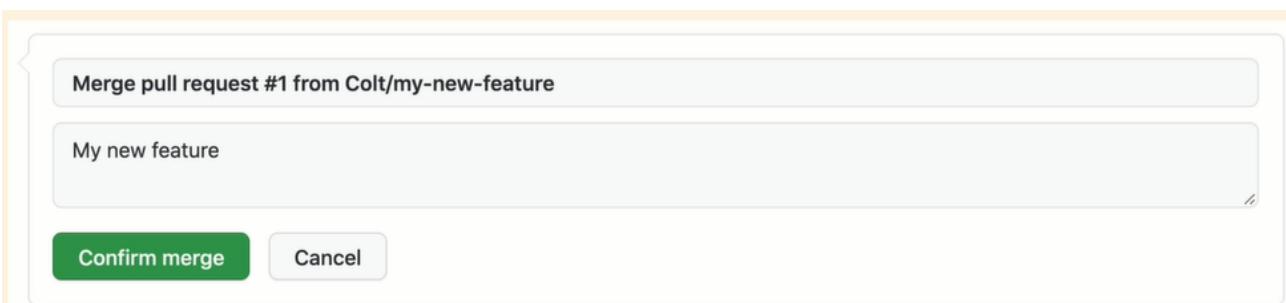
My boss find some issues, leaves me some feedback. She asked me to make some changes, ping me when its ready. I can notfy ppl and mention ppl in comments and i can attach files and she can also just closed it, its kinda blunt way of saying no.



Colt: hm, r u sure you want me to do that. Whats the motivation and im not sure that i agree with that. What do u think, other developers. I can bring somebody else in and can have a full discussion on the team.



Once I make the requested changes, my boss (or whoever is in charge of merging) can merge in my pull request!



Once I make the requested changes, my boss (or whoever is in charge of merging) can merge in my pull request!

The above text will be used in the resulting merge commit.

Note: u dont have to create a new PR. if they give u some feedback or tell u to make some changes.

Making our First Pull request

The screenshot shows the GitHub interface for creating a pull request. At the top, a message says "navbar had recent pushes less than a minute ago". Below it, there's a navigation bar with dropdowns for "navbar", "2 branches", and "0 tags", and buttons for "Go to file", "Add file", and "Code". A green button on the right says "Compare & pull request".

The main area shows a list of commits:

- Colt add bootstrap boilerplate (583c48e, 13 hours ago, 4 commits)
- Update README.md (13 hours ago)
- index.html (add bootstrap boilerplate, 13 hours ago)

A blue arrow points from the "Compare & pull request" button to the "Compare" dropdown in the navigation bar. Another blue arrow points from the "Merge feature branch on main" text to the "base: main" dropdown.

In the middle section, there's a comment input field with the placeholder "Add in bootstrap Navbar" and two tabs: "Write" and "Preview". Below the input field, a comment is shown: "this is my comment! @Colt please take a look!!!". A blue arrow points from the "Create pull request" button to the "is:pr is:open" search bar at the top of the page.

The bottom section shows the pull request details:

- Filters: "is:pr is:open"
- 1 Open, 0 Closed
- Author: StevieChicks
- Labels: 9
- Milestones: 0
- New pull request

The pull request list shows one item:

- Add in bootstrap Navbar (1 opened 17 seconds ago by StevieChicks)
 - StevieChicks commented 21 seconds ago: "this is my comment! @Colt please take a look!!!"
 - StevieChicks and others added 2 commits 13 hours ago:
 - add broken navbar
 - fix navbar bug

A blue box highlights the commit log. A blue arrow points from the "Merge feature branch on main" text to the commit log.

At the bottom, a note says "Add more commits by pushing to the navbar branch on Colt/feature-branch-workflow.". A "Merge pull request" button is shown with a checkmark and the text "This branch has no conflicts with the base branch".

We see, Stevie has the permission to merge it in.

we make a PR and then we see PR is registered and we saw stevie has the permission to merge in the PR. This is not a recommended way. Merge approval or a team is required.

here, colt is from the pr approval team. we see colt image(not stevie the pr maker). since colt was mention in the PR he also get notified. he can close or merge the PR.

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also [open this in GitHub Desktop](#) or view command line instructions.

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

close pull request or post comment

Pull request successfully merged and closed
You're all set—the navbar branch can be safely deleted.

feature-branch-workflow > git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
feature-branch-workflow > git branch -D navbar
Deleted branch navbar (was 47bd0b5).

you can communicate here first and give feedback or involve more people or say looks good merging in 10mins.
All this before merging PR. So u basically now goto ur local machine and pull the changes. Then if u like u can delete branch from github and then the git.

Merging pull Request with Conflicts

base: main ▾ compare: new-heading ▾ **✗ Can't automatically merge** Don't worry, you can still create the pull request.

change heading to bock bock

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

You wanna merge ur feature branch and the main branch. But there's a conflict

we still can make a PR

Create pull request ▾

 StevieChicks commented now

I am a chicken and I changed the heading to say "bock bock" I hope you like it!

-o-  change heading to bock bock c75c3b0

Add more commits by pushing to the `new-heading` branch on [Colt/feature-branch-workflow](#).

 This branch has conflicts that must be resolved
Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files
`index.html`

[Resolve conflicts](#)

Two ways

Filters Labels 9 Milestones 0 New pull request

1 Open 1 Closed Author Label Projects Milestones Reviews Assignee Sort

 **change heading to bock bock**

#2 opened 42 seconds ago by StevieChicks

 This branch has conflicts that must be resolved
Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files
`index.html`

[Merge pull request](#) You can also open this in [GitHub Desktop](#) or view [command line instructions](#).

```
<<<<< new-heading
      <h1 class="display-1">Bock Bock</h1>
      =====
      <h1>Hello there Everyone!!!!</h1>
>>>>> main|
```

1. You can resolve the conflict on the browser
2. You can resolve using vs code so click on command-line instructions

Or

 This branch has conflicts that must be resolved
Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files
`index.html`

[Merge pull request](#) You can also open this in [GitHub Desktop](#) or view [command line instructions](#).

Checkout via command line

If you cannot merge a pull request automatically here, you have the option of checking it out via command line to resolve conflicts and perform a manual merge.

HTTPS

Git

Patch

<https://github.com/Colt/feature-branch-workflow.git>

Step 1: From your project repository, bring in the changes and test.

```
git fetch origin  
git checkout -b new-heading origin/new-heading  
git merge main
```

Step 2: Merge the changes and update on GitHub.

```
git checkout main  
git merge --no-ff new-heading  
git push origin main
```

git fetch origin

to bring the changes that our remote repo to our local repo

feature-branch-workflow > git switch new-heading
new-heading —> origin/new-heading

feature-branch-workflow > git merge main new-heading

Already up to date. new heading and main are the same

feature-branch-workflow > git switch main new-heading

Switched to branch 'main'

Your branch is behind 'origin/main' by 1 commit, and can
be fast-forwarded.

(use "git pull" to update your local branch)

Activate Windows
Go to Settings to activate Windows.

we used git fetch not git pull, since origin/master is ahead. so we fix it
below

feature-branch-workflow > git pull origin main

feature-branch-workflow > git switch new-heading main

Switched to branch 'new-heading'

Your branch is up to date with 'origin/new-heading'.

feature-branch-workflow > git merge main new-heading

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
===== HEAD (Current Change) =====
<h1 class="display-1">Bock Bock</h1>
=====
<h1>Hello there Everyone!!!!</h1>
===== main (Incoming Change)
<div class="container">
<div class="row row-cols-1 row-cols-md-3 mb-3 text-center">

above we resolve
conflicts and commit the
changes.

it doesn't open vs code by itself. it just gives us in this format, what is the conflict, and we
are supposed to see if we wanna accept incoming change or etc.

we have to open the file ourselves.
which file if u ask, just check with git status.

```
feature-branch-workflow > git status      new-heading
On branch new-heading
Your branch is ahead of 'origin/new-heading' by 2 commits

feature-branch-workflow > git switch main      new-heading
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
feature-branch-workflow >                                main
```

Step 2: Merge the changes and update on GitHub.

```
git checkout main
git merge --no-ff new-heading
git push origin main
```

--no-ff that's an indicator we telling git merge to not to fast forward, if it detects it can.

Bcoz when we do a fast forward, instead of making merge commit, it will move the branch pointer to some new commit. Sometimes you wanna prevent that from happening especially when we want to maintain a specific branch.

We want to make it clear that there was a branch merged in here just to have that history.

You dont have to do this part, but it's what github recommends.

Above , made 100% sense kazim.

```
feature-branch-workflow > git merge --no-ff new-heading
Merge made by the 'recursive' strategy.
index.html | 4 +-+-
 1 file changed, 2 insertions(+), 2 deletions(-)
feature-branch-workflow > git push origin main      main
```



My Boss's Local Machine

My boss can merge the branch and resolve the conflicts locally...

Switch to the branch in question. Merge in master and resolve the conflicts.

```
> git fetch origin  
> git switch my-new-feature  
> git merge master  
> fix conflicts!
```

Switch to master. Merge in the feature branch (now with no conflicts). Push changes up to Github.

```
> git switch master  
> git merge my-new-feature  
> git push origin master
```

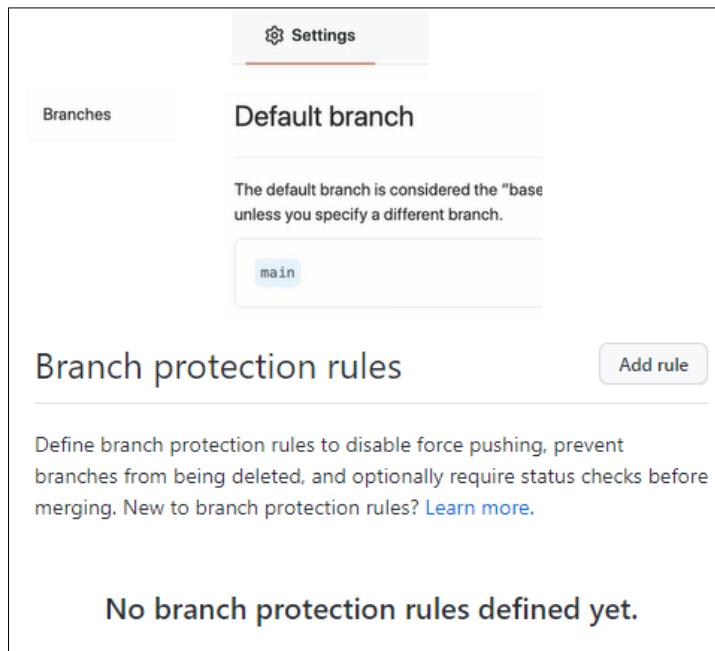
```
feature-branch-workflow > git branch    main  
feature-branch-workflow > git branch -D new-heading
```

```
feature-branch-workflow > git pull origin main  
remote: Enumerating objects: 11, done.  
remote: Counting objects: 100% (11/11), done
```

Configuring Branch protection Rules

Goto settings, if you are somebody who have access to setting on a repository.

Lets say you have main and master, if you want main to be “source of truth/ official branch”.



The screenshot shows the 'Branch protection rules' section of the GitHub repository settings. It includes a 'Default branch' section with 'main' selected as the default branch. Below it is a 'Branch protection rules' section with a heading 'No branch protection rules defined yet.' and a 'Add rule' button.

Since, there can be 1000s of branches. Organization enforce rules.



The screenshot shows the 'Branch protection rules' configuration for an organization. It lists three protected branches: '/feature/askdjlas', '/relase/', and 'main'. Each entry includes a 'Branch name pattern' field and a 'Require pull request reviews before merging' checkbox.

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 approvin 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.

This means that our commits need to be made to a non-protected branch, basically not main and submitted via pull request and atleast one person need to approve the PR before it can be merged

here we cant commit to main because it is a protected branch. you have to create a new branch for this commit and start a pull request.

The screenshot shows the GitHub 'Commit changes' interface. A user with the name 'StevieChicks' is committing changes to 'Update README.md'. There is an optional extended description field. A note at the bottom states: 'You can't commit to `main` because it is a **protected branch**. Create a new branch for this commit and start a pull request. Learn more about pull requests.' A dropdown menu shows the branch name 'StevieChicks-patch-1'.

Now here we are trying to merge main with StevieChick-patch-1

The screenshot shows the GitHub merge interface. A user with the name 'StevieChicks' has commented 'No description provided.' Below the merge button, it says 'Update README.md'. A note at the top says 'Add more commits by pushing to the **StevieChicks-patch-1** branch on **Colt/feature-branch**'. Below this, two errors are shown: 'Review required' (At least 1 approving review is required by reviewers with write access) and 'Merging is blocked' (Merging can be performed automatically with 1 approving review).

Now colt is here.

Add more commits by pushing to the **StevieChicks-patch-1** branch on **Colt/feature-branch-workflow**.

The screenshot shows the GitHub merge interface for an administrator. It displays the same 'Review required' and 'Merging is blocked' errors as the previous screenshot. An additional note says 'As an administrator, you may still merge this pull request.' A 'Merge pull request' button is present. To the right, handwritten text in red says: 'if i am the administrator i can simply merge in. But lets review, click on Add the review'.

The screenshot shows the GitHub pull request code diff. It includes a merge commit at the top with the message '# feature-branch-workflow'. The commit body reads: 'A simple, boring repo to demonstrate Git workflows involving feature branches.' Line 5 shows a merge commit with the message 'Bock Bock'.

Always with the Bock Bock, Stevie? Get a new catchphrase!

Attach files by dragging & dropping, selecting or pasting them.

[Cancel](#) [Add single comment](#) [Start a review](#)

Now, stevie is checking the review done by colt

 Colt reviewed now



README.md

```
... ... @@ -1,3 +1,5 @@
1 1 # feature-branch-workflow
2 2
3 3 A simple, boring repo to demonstrate Git wo
4 +
5 + Bock Bock
```

Colt now Owner

Always with the Bock Bock, Stevie? Get a new catchphrase!

0 / 1 files viewed (i) Review changes

Finish your review X

Write Preview H B I ↔ 🔗 ≡ ≡ ✓ @ ✉ ↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them. MB

Comment
Submit general feedback without explicit approval.

Approve
Submit feedback and approve merging these changes.

Request changes
Submit feedback that must be addressed before merging.

Submit review

Note u can edit the file by clicking on the pencil icon. and then if u r on the main branch.
u might not be able to do it.
Cuz of branch protection rules.
so u can create feature branch.
Then we do a PR.

We can see our PR being registered. But we cant merge that in. So steview will wait for colt and have a discussion with the team.

The screenshot shows a GitHub pull request review interface. At the top, there's a green circular icon with a checkmark and a gear symbol, followed by the text "Changes approved". Below it, it says "1 approving review by reviewers with write access. [Learn more.](#)". Underneath, there's a section with a checkmark and the text "1 approval" with a link icon. The next section has a gear icon and the text "Continuous integration has not been set up" followed by "GitHub Actions and [several other apps](#) can be used to automatically catch bugs and enforce style.". The final section has a checkmark and the text "This branch has no conflicts with the base branch" followed by "Merging can be performed automatically.". At the bottom left is a green button labeled "Merge pull request" with a dropdown arrow, and at the bottom right, it says "You can also open this in GitHub Desktop or view [command line instructions](#)".

It was just 1 person is approved. Its not merge. There is even more fancy permission if we have organization setup or enterprise account.

For the PR reviewer/approval giver, he first needs to review and then he can merge.

