

# GitHub



Presented by SCU Association for Computing Machinery

# Weekly Challenge Of the Week

Given an array of  $n$  integers where  $n > 1$ , `nums`, return an array `output` such that `output[i]` is equal to the product of all the elements of `nums` except `nums[i]`.

Solve it **without division** and in  $O(n)$ .

For example, given `[1,2,3,4]`, return `[24,12,8,6]`.

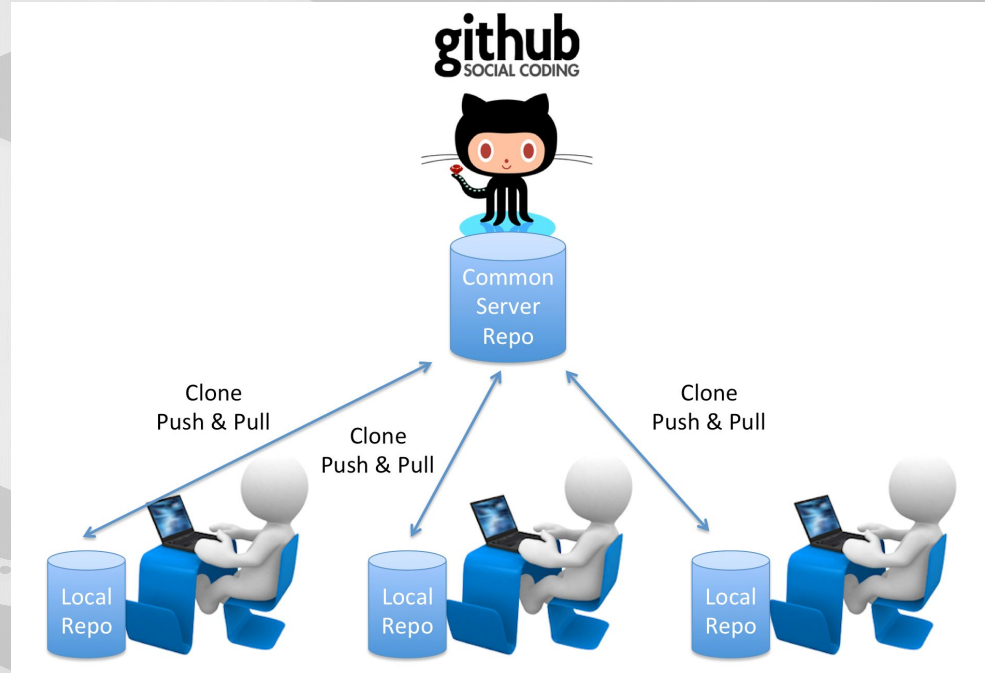
# What is version control?

- Git vs Github

- Git is a program that tracks Changes to projects (**version control**)
- Github **hosts** the projects
  - Other less used hosts:  
GitLab  
BitBucket

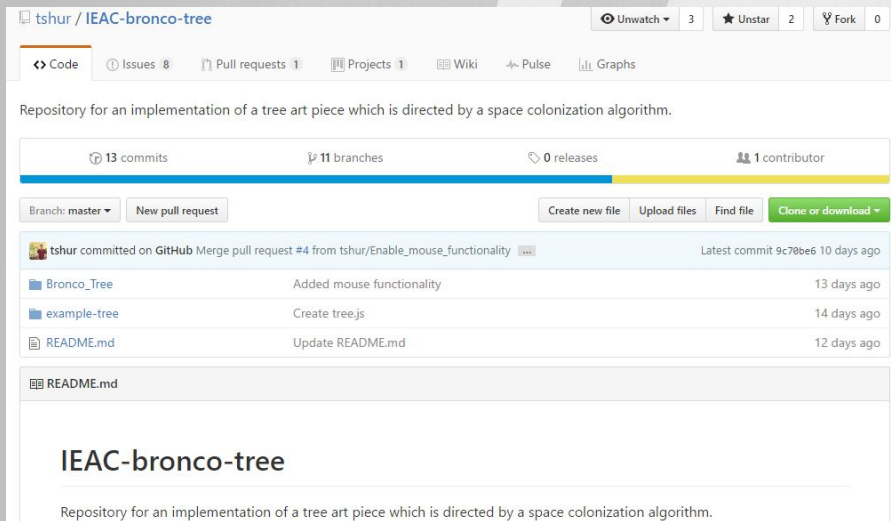
- Useful on resumés!

- You need to know *some* form of version control at your job



# Github repository

- What is a **repository** on Github?
  - Simply a project on Github which someone has made and people can add to
- What is a **local repository** on your computer?
  - All the code for a project + a **.git repository file** which tracks changes that have been made



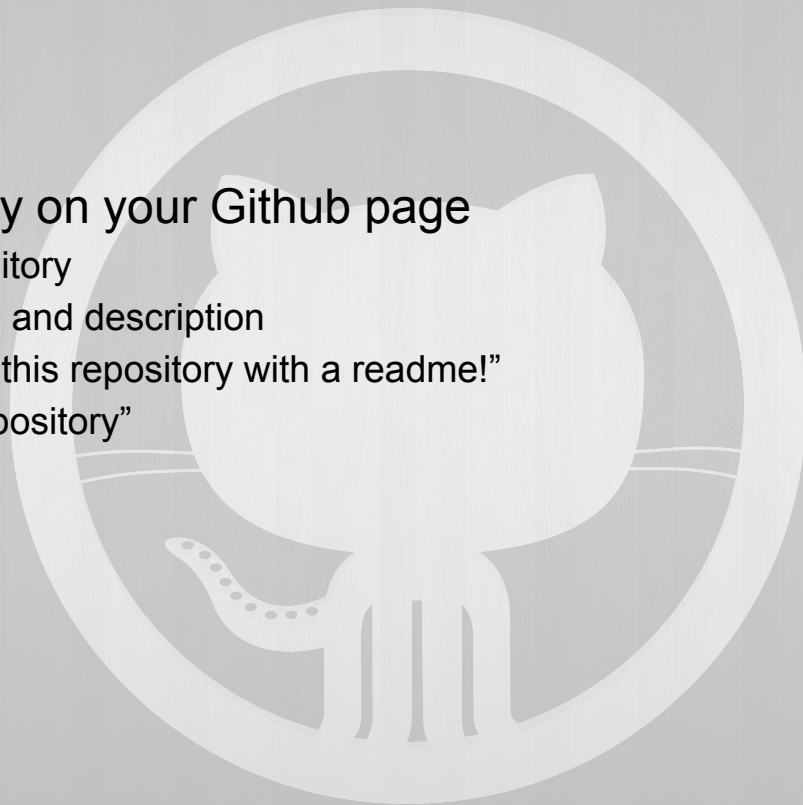
The screenshot shows the GitHub interface for the repository 'IEAC-bronco-tree'. At the top, it indicates the user 'tshur' and repository stats: 3 Unwatch, 2 Unstar, and 0 Fork. Below this is a navigation bar with links to Code, Issues (8), Pull requests (1), Projects (1), Wiki, Pulse, and Graphs. The repository description is 'Repository for an implementation of a tree art piece which is directed by a space colonization algorithm.' Below the description, it shows 13 commits, 11 branches, 0 releases, and 1 contributor. There are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A commit history table is visible, showing the latest commit by 'tshur' on '2/13/2017 10:04 PM' with the message 'Merge pull request #4 from tshur/Enable\_mouse\_functionality'. Below the commit history, there is a section for the 'README.md' file, which contains the repository name 'IEAC-bronco-tree' and its description.

File/Folder	Commit Date	Commit Message
.git	2/13/2017 10:04 PM	File folder
Bronco_Tree	2/13/2017 10:00 PM	File folder
example-tree	2/5/2017 2:20 PM	File folder
README.md	2/5/2017 2:20 PM	MD File

.git	2/13/2017 10:04 PM	File folder
Bronco_Tree	2/13/2017 10:00 PM	File folder
example-tree	2/5/2017 2:20 PM	File folder
README.md	2/5/2017 2:20 PM	MD File

# Making a project on Github

- Make a repository on your Github page
  - Click new repository
  - Choose a name and description
  - check “initialize this repository with a readme!”
  - Click “create repository”



# Local Repository

- Now, we need somewhere to on our computer to work on this project
  - Create create a folder on your computer with the same name as the project you create on Github
  - Open “Github Desktop” that you downloaded before the tutorial
  - If you haven’t already, login with your Github username and password
  - Select the “+” at the top left and select the “clone” tab
  - You should see the repository we created on Github
  - Select it and press “clone”
  - Select a folder to clone into
    - Clone into something like “coding projects.” We don’t need to make a folder with the repo name, Github GUI will do that when we clone it!

master ▾

Changes

History

 Pull request



Compare ▾

 Sync

master



0 changes

No changes

# No local changes

Would you like to [open this repository](#) in Explorer?

Summary

Description



Commit to master

master ▾

Changes

History

 Pull request



Compare ▾

 Sync

master



Initial commit

5 minutes ago by Julian Callin

Initial commit



Julian Callin

 9715aea

 GitHub

 Revert

 Collapse all

▼ README.md



...	...	@@ -0,0 +1,2 @@
1	+	# ACM-github-2017
2	+	a tutorial repo



Coding Projects > ACM-github-2017

Name

Date modified

Type



.git

2/14/2017 4:52 PM

File folder



README.md

2/14/2017 4:44 PM

MD File

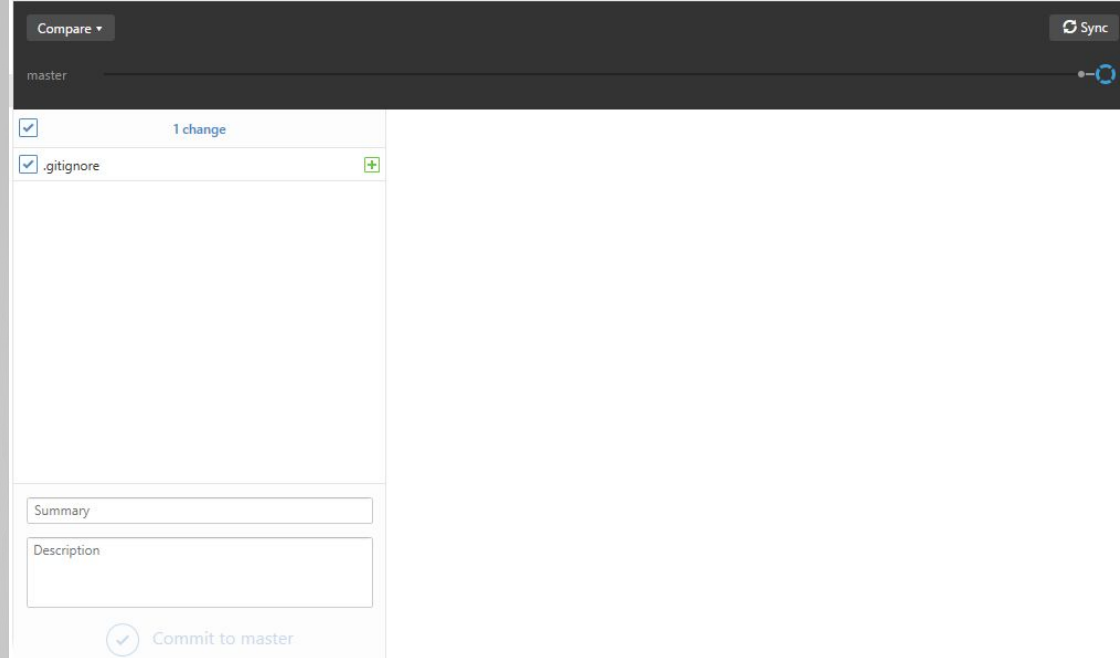
# Tracking

- Need to tell .git which files you actually want included in your workflow
- Click the settings icon in the top right of the Github GUI and click “repository settings”
- On the “ignored files” tab click the “add a default .gitignore” button
- Click “ok”

## Ignored files

The .gitignore file controls which files are tracked by Git and which are ignored. Check out some [examples](#) on GitHub, or ignore a file by right clicking on it in the uncommitted changes view.

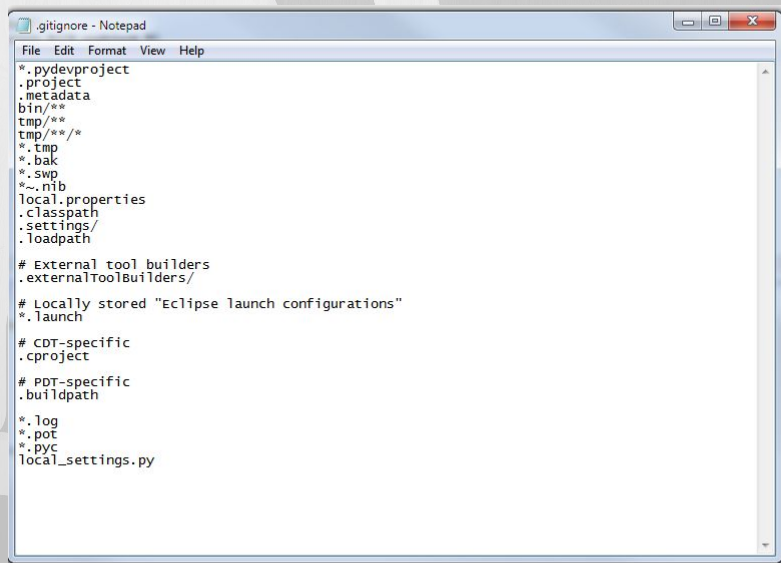
```
#####  
## Eclipse  
#####  
  
*.pydevproject  
.project  
.metadata  
bin/  
tmp/  
*.tmp  
*.bak  
*.swp  
*~.nib  
local.properties  
.classpath  
.settings/  
.loadpath  
  
# External tool builders  
.externalToolBuilders/  
  
# Locally stored "Eclipse launch configurations"  
*.launch  
  
# CDT-specific  
.cproject  
  
# PDT-specific  
.buildpath
```



Coding Projects > ACM-github-2017			
	Name	Date modified	Type
	.git	2/14/2017 4:52 PM	File folder
	.gitignore	2/14/2017 4:51 PM	Text Document
	README.md	2/14/2017 4:44 PM	MD File

# Tracking

- The Github GUI automatically adds things that are changed
  - Hover your mouse over the “+” sign next to .gitignore and see that it says “added”
- Some of the time, there won't be nice things like this in place!
  - Command line: “git add <file> <file> ...”
- .gitignore
  - Tells git NEVER to track things that look like .pyc .out etc...



```
.pydevproject
.project
.metadata
bin/**
tmp/**
*.tmp
*.bak
*.swp
*.nib
local.properties
.classpath
.settings/
.loadpath

# External tool builders
.externalToolBuilders/

# Locally stored "Eclipse launch configurations"
*.launch

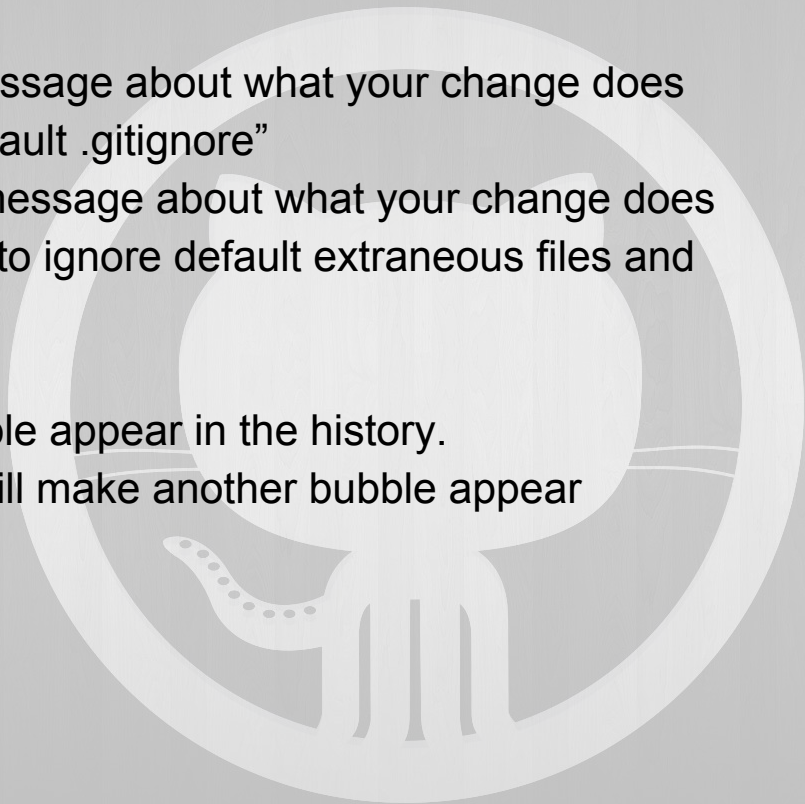
# CDT-specific
.cproject

# PDT-specific
.buildpath

*.log
*.pot
*.pyc
local_settings.py
```

# Committing

- Type a summary message about what your change does
  - le “Added a default .gitignore”
- Type a description message about what your change does
  - le “A .gitignore to ignore default extraneous files and executables”
  - Click commit!
- Notice the grey bubble appear in the history.
  - Each commit will make another bubble appear



Compare ▾

Sync

master



0 changes

No changes

Summary

Description

# No local changes

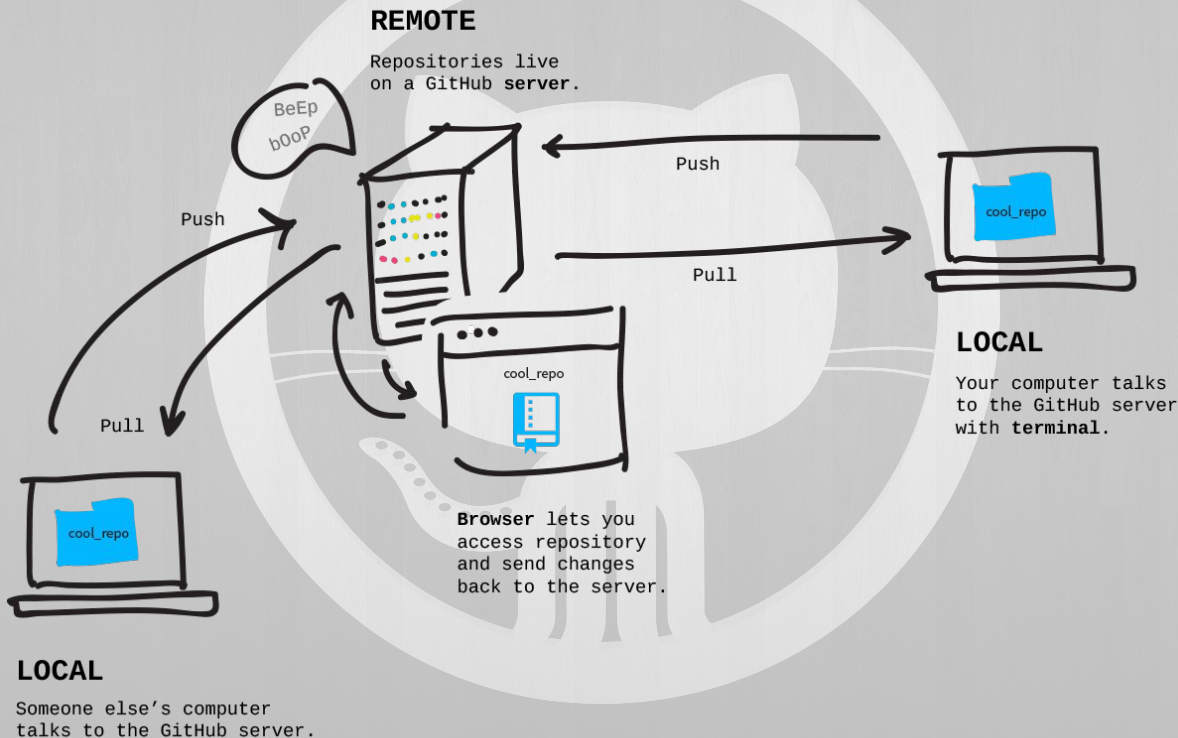
Would you like to [open this repository](#) in Explorer?

# Committing

- “Commit early, commit often!”
- Try and change only code that is relevant to the task at hand
- If you’re going to change other code, at least make it a separate commit
- Example:
  - Someone gives you the task of rewriting a class to accommodate storing stuff to a database
  - While writing, you notice there is another class with similar functionality, and you decide to merge them into one class

# “Remotes”

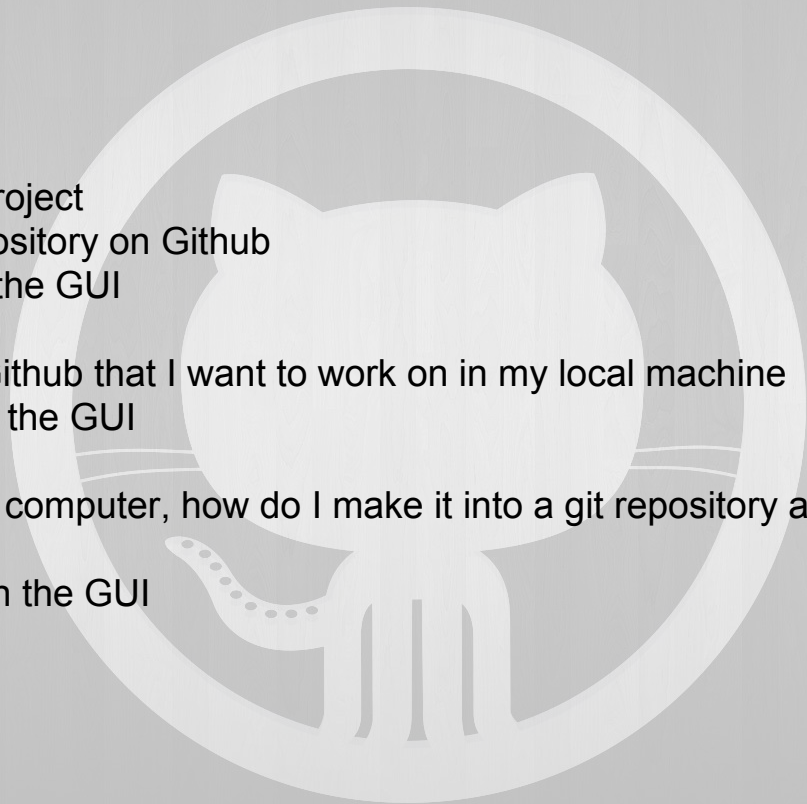
- A remote is somewhere that our new project gets synced with
  - In this case, Github servers





# Different workflows

1. I want to start a new project
  - a. Make a new repository on Github
  - b. See it pop up in the GUI
2. I have something on Github that I want to work on in my local machine
  - a. “Clone” option in the GUI
3. I have a project on my computer, how do I make it into a git repository and then put it up on Github??
  - a. “Create” option in the GUI



# Pushing

- Now that we've added committed our .gitignore, we need it to appear on Github. The changes are on our local files, but the files are also hosted on Github and need to be updated
- Click "Sync" on the top right
- Go to the Github page for your project and refresh the page. Tada!

**In case of fire**



1. git commit



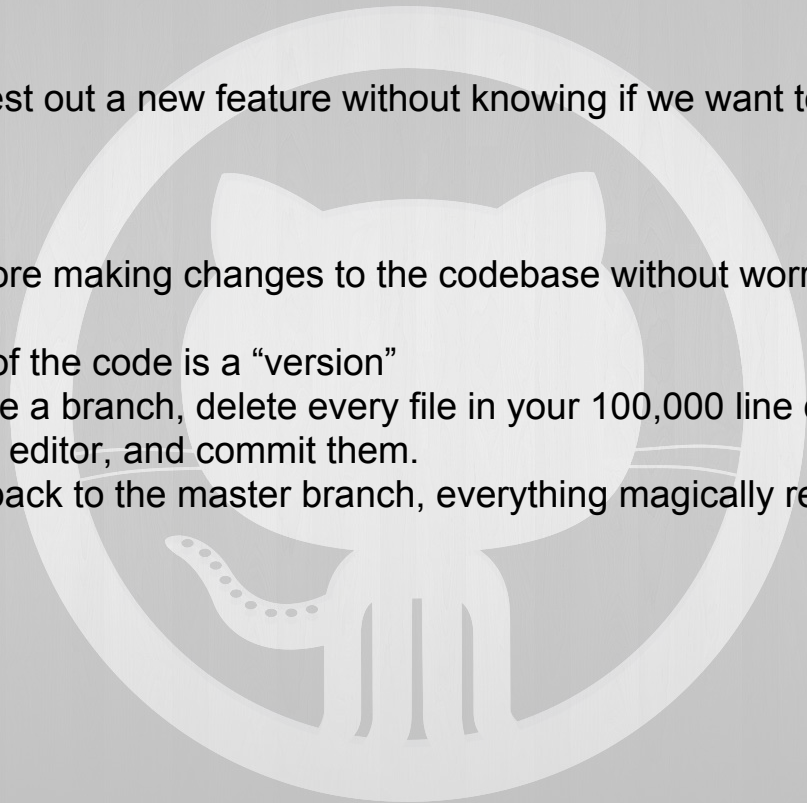
2. git push

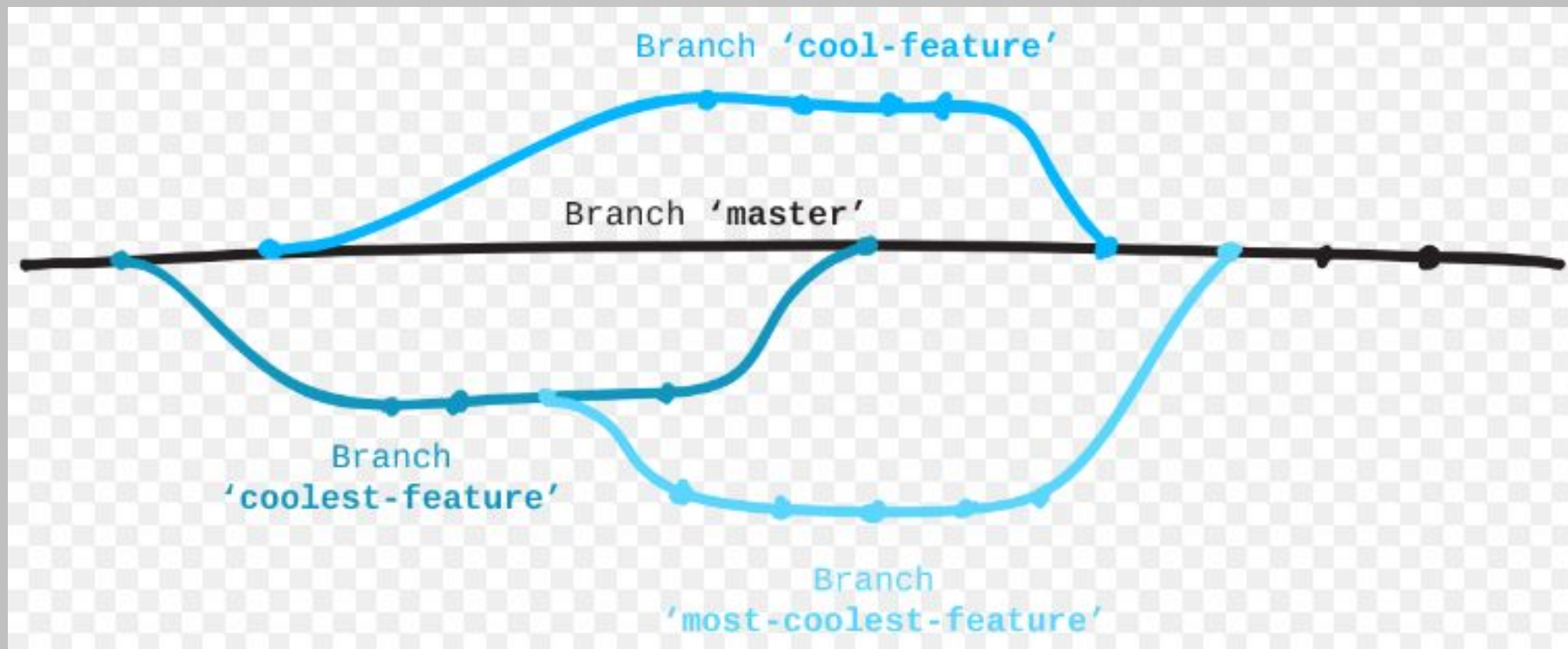


3. leave building


# Cool, congrats, now what?

- What if we want to test out a new feature without knowing if we want to keep it?
- Branching
  - Lets you explore making changes to the codebase without worrying about breaking something
  - Each branch of the code is a “version”
  - You can create a branch, delete every file in your 100,000 line code-base, save all the files in the editor, and commit them.
  - If you switch back to the master branch, everything magically reappears!





# Branching

1. Click on “create a new branch” 
2. Name your branch “add\_a\_main”
3. Go to your project folder on your computer, add a file called “main.c” and add some c code to it
  - a. Note the change in the tree structure of the timeline
4. Commit, and press “publish”

Update from master

View branch

 Sync

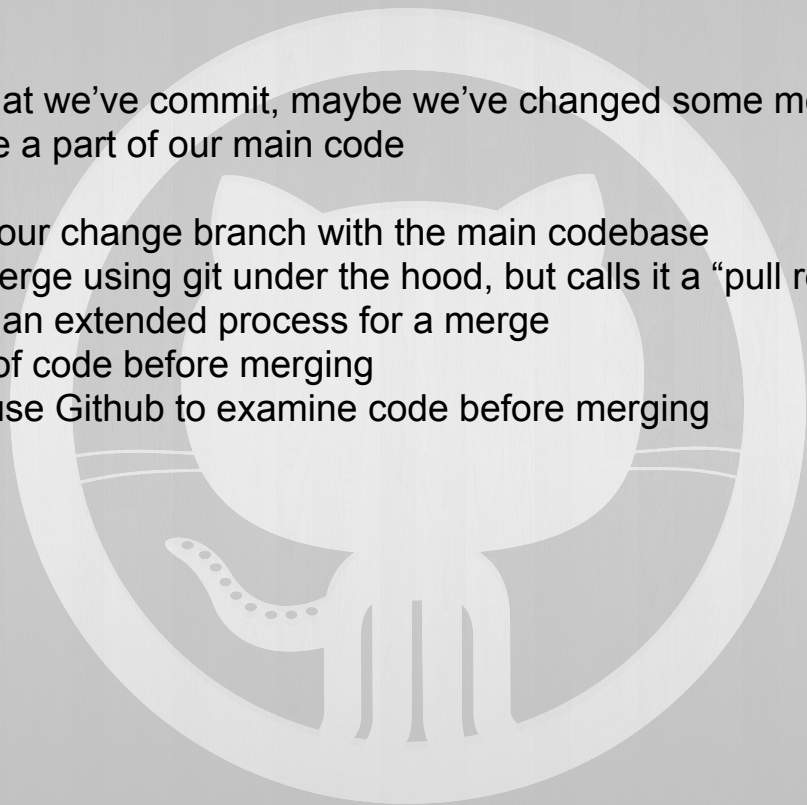
master ▼

add\_a\_main



# Merging

- We're happy with what we've commit, maybe we've changed some more stuff -- we're ready for our changes to be a part of our main code
1. We want to “merge” our change branch with the main codebase
  2. Github performs a merge using git under the hood, but calls it a “pull request”
  3. A pull request is just an extended process for a merge
    - a. Allows review of code before merging
    - b. Allows you to use Github to examine code before merging



# Pull Requests

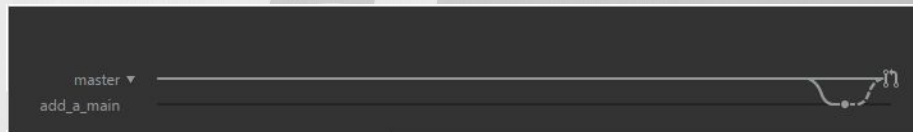
1. Click “pull request” in top right
  2. Make sure you are merging from your new branch *into* master
  3. Enter in a summary and description
  4. Send the pull request
  5. Click “view it on Github”
- Note that the timeline hasn’t changed yet, we are still waiting for the merge to be approved

Pull request

from add\_a\_main into master ▼

Adding a driver

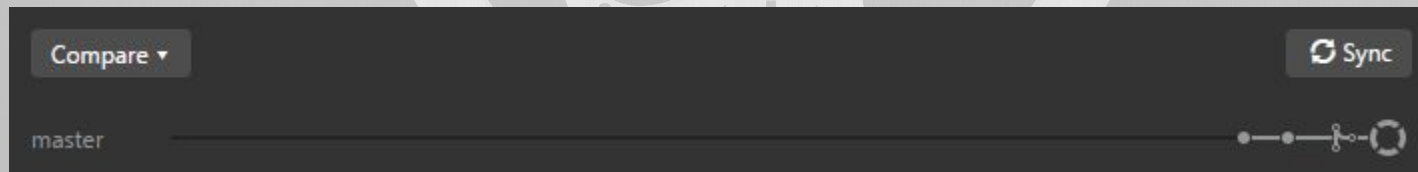
Our application now prints stuff





# Merging

1. On Github click on “merge pull request” → confirm merge → delete branch
  - a. Our changes are now part of the master branch
2. Go back to the GUI
3. Switch the master branch
4. Click “sync”
5. We can now see the symbol telling us that we just merged our branches





# Conclusion

- How to start a project on Github
- How to get the project onto your computer
- Tracking changes
- Committing
- Branching
- Merging (pull requests)

