Google Developer Student Clubs

$ git init

Intro to Git and GitHub

Big thanks to Krish Chowdhary (GDSC Alumni) for allowing us to re-use some of his old slides!

If you haven’t already installed it... (~8 min)

Creating GitHub Account: github.com/join

Download Git for...

Windows: git-scm.com/download/win

Linux:

- **sudo apt install git-all** (if you use a Debian-based Linux)
- *OR* **sudo dnf install git-all** (if you use a Red Hat-based Linux)

MacOS:

- git-scm.com/download/mac
- *OR* type **brew install git** in your terminal (if you have homebrew)

Google Developer Student Clubs
University of Toronto Mississauga

Some icebreaker questions in the meantime

1. What year are you in?
2. What POSt are you in or planning on pursuing?
3. Have you ever used Git before?
4. What is your favourite programming language?
5. Any questions for us? 😄

Google Developer Student Clubs
University of Toronto Mississauga

git

Google Developer Student Clubs
University of Toronto Mississauga

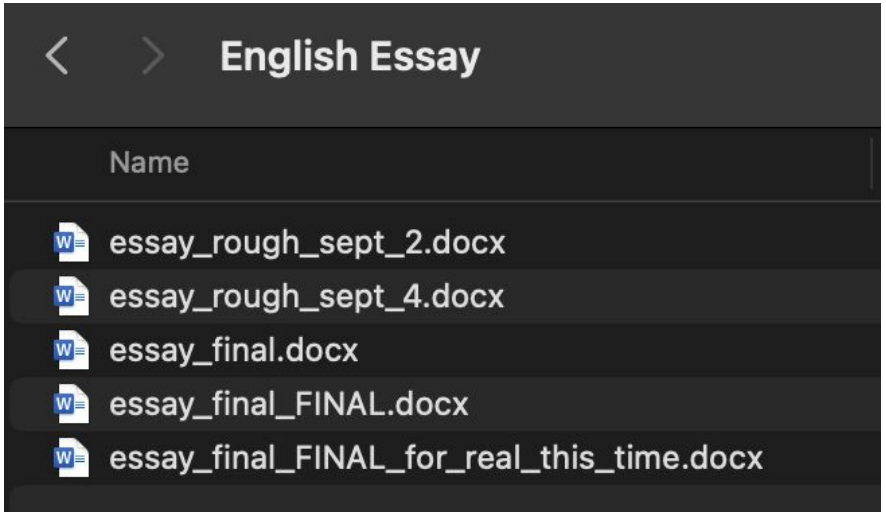
Collaboration

Version control

More

- Git is a **version control** system
- A software that you interact with through your **terminal/command line**
- Allows you to **keep track of changes** made to your project
- **Facilitates effective collaboration** for multiple individuals working on one project

Ever done something like this?



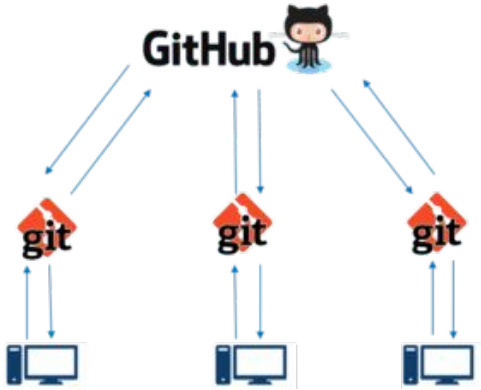
A better way? **use Git**

Google Developer Student Clubs
University of Toronto Mississauga

GitHub

Google Developer Student Clubs
University of Toronto Mississauga

What is GitHub?

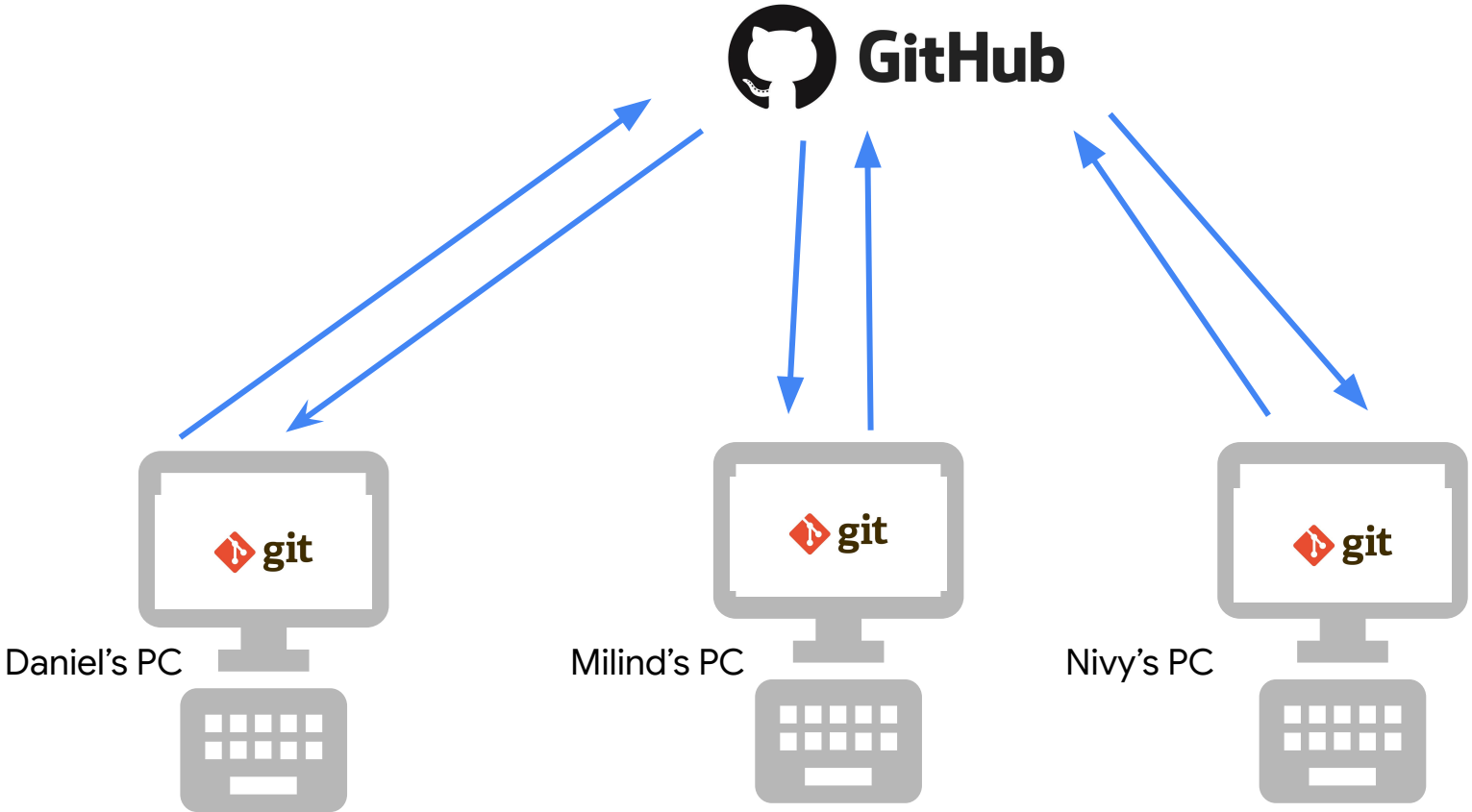


- **A website** that is the central location on the internet **where you push** (upload) and **pull** (download) **code from**
- From there people can view the project and its version history and starting contributing to it
- Think of it as a *super advanced* Google Drive
- Let's take a look at an example GitHub **repository:** github.com/utmmcss/mcss-website-frontend
- ***Repository:*** *the collection of all the versions of the files in a project*

Git != GitHub

Git is a **command line tool**. GitHub is a **website**

GitHub Diagram



GitHub

Let's *git* started.

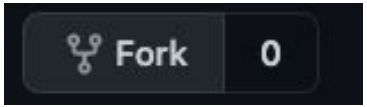
Google Developer Student Clubs
University of Toronto Mississauga

Hands-on activity: **fork**ing and **clon(e)**ing (~5min)

Your task:

Don't forget the .git suffix

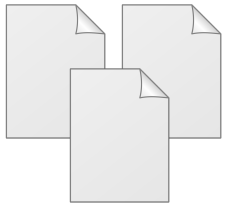
Remove the "<>" before using

1. Visit **github.com/Daniel-Laufer/GDSCUTM-git-workshop**
2. Click the "Fork" button 

    - This will create a copy of this repository where **your** GitHub account is the owner
3. Then in your terminal (on your local computer) type: **git clone <your forked repository url>.git**
    - This will essentially download your repository from GitHub to your computer
4. Type: **ls** (first letter is a lowercase 'L'). This command lists out everything in your current directory
    - You will see a new directory listed (aka a folder)
5. Type: **cd <the name of this new directory>** to "move" into that directory

Version control basics with Git

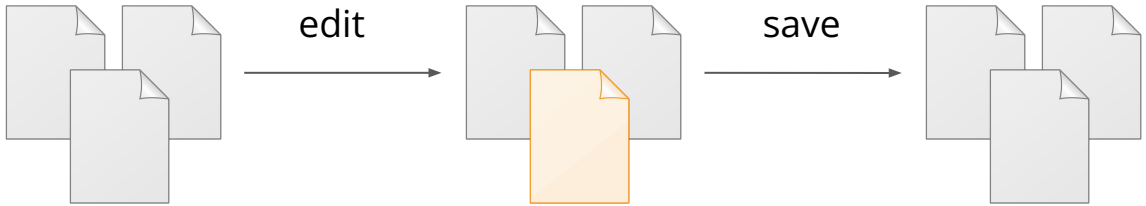
1. Staging & Committing



1. Staging & Committing

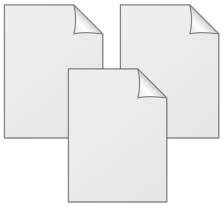
Developer Student Clubs
University of Toronto Mississauga

1. Staging & Committing



Developer Student Clubs
University of Toronto Mississauga

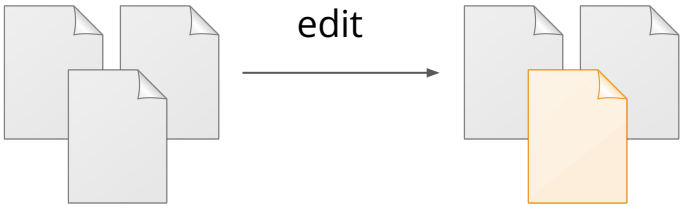
1. Staging & Committing



“Main” git branch

Initial commit

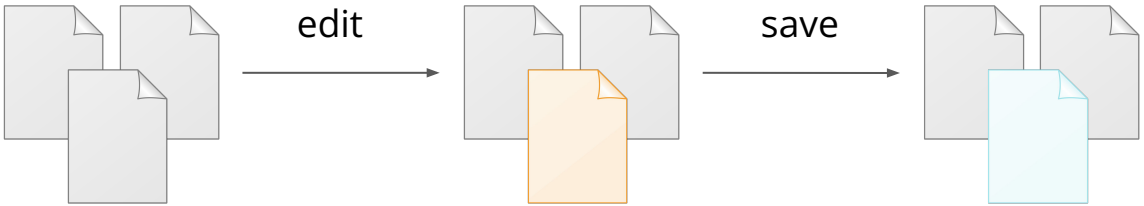
1. Staging & Committing



“Main” git branch

Initial commit

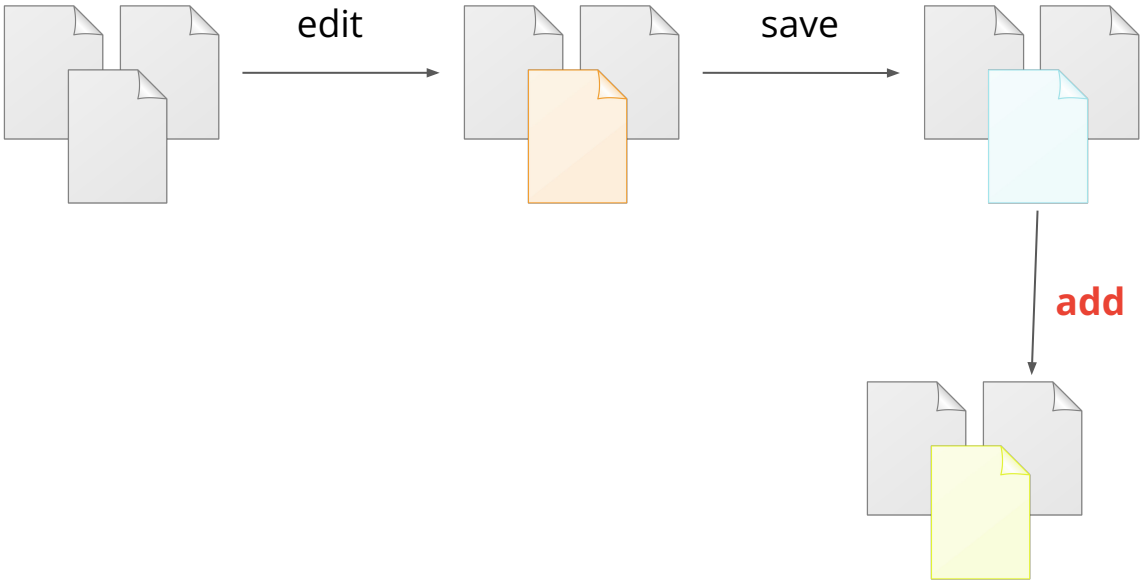
1. Staging & Committing



“main” git branch

Initial commit

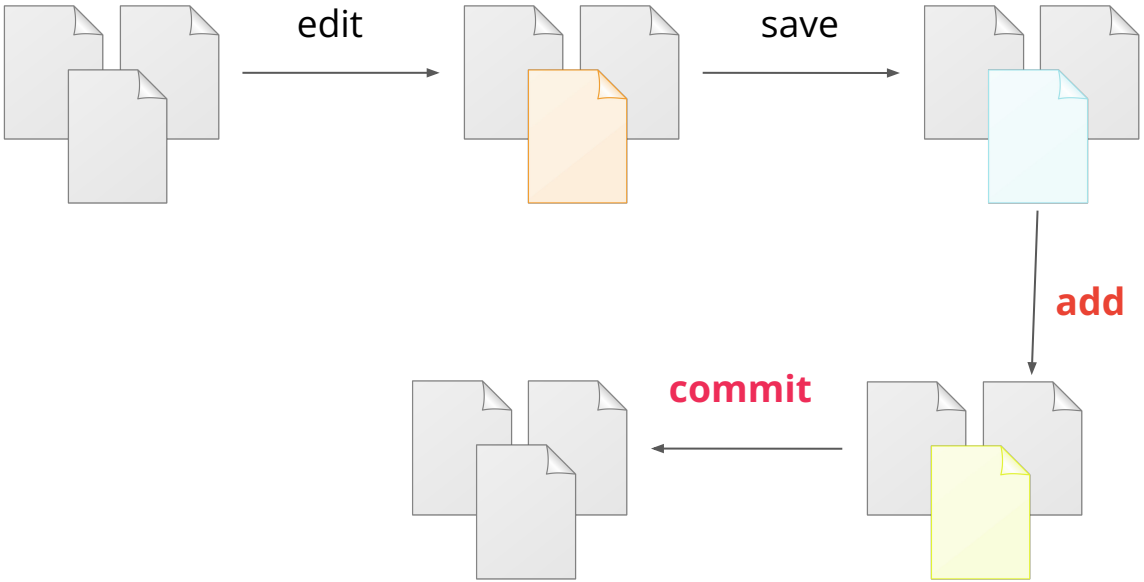
1. Staging & Committing



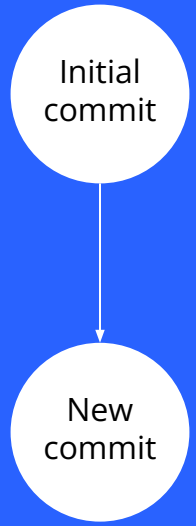
“main” git branch

Initial commit

1. Staging & Committing



“main” git branch

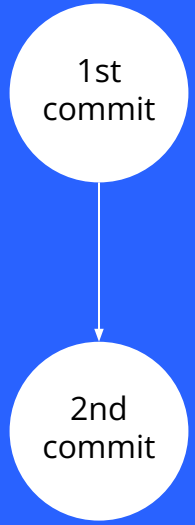


Hold up... what's a commit again?

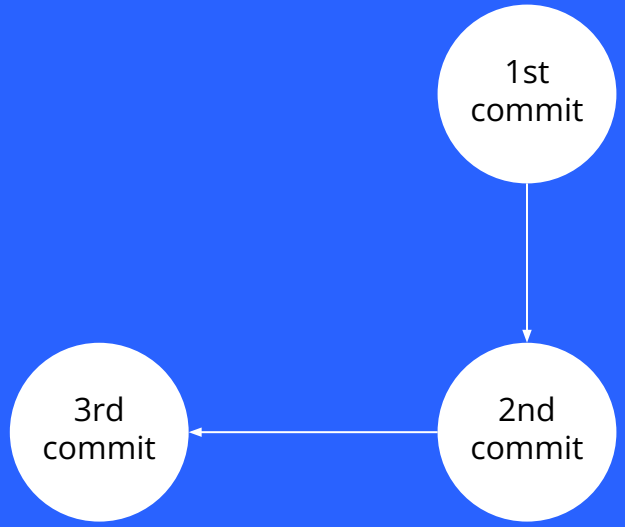
It's simply a **change** to your project that **git records** in its version history.

Developer Student Clubs
University of Toronto Mississauga

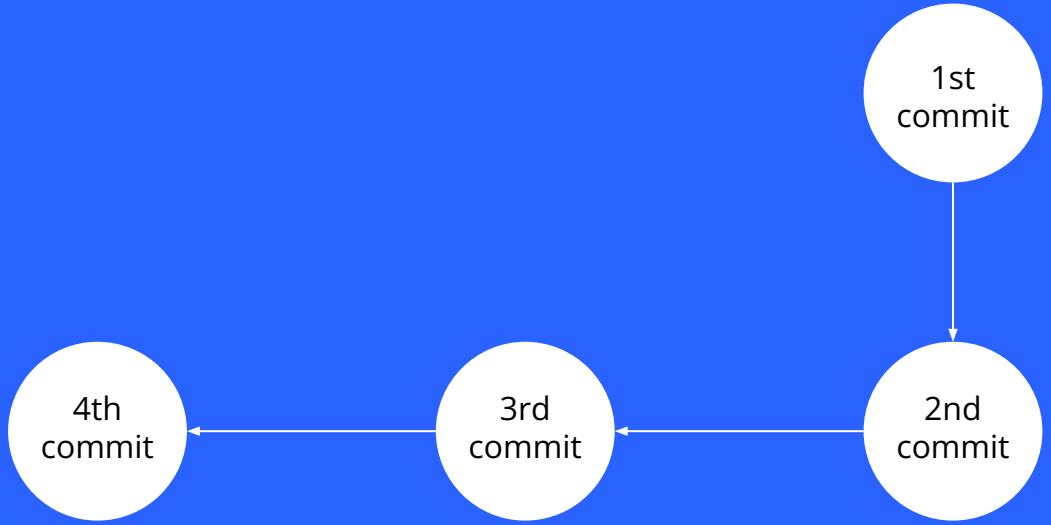
Staging & Committing



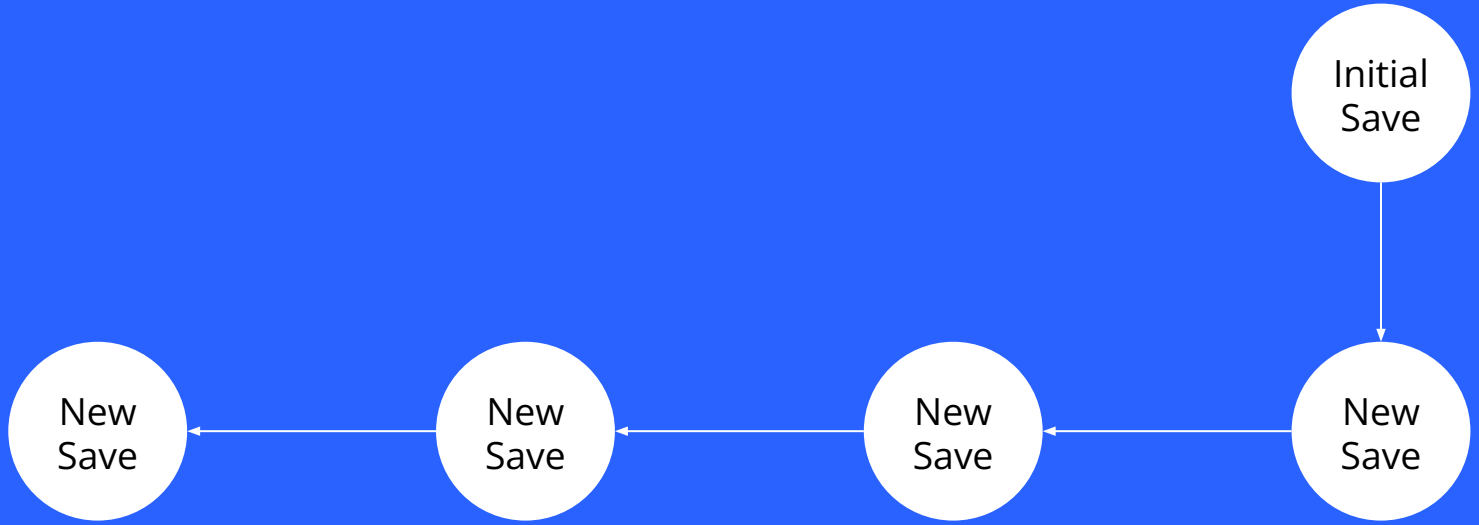
Staging & Committing



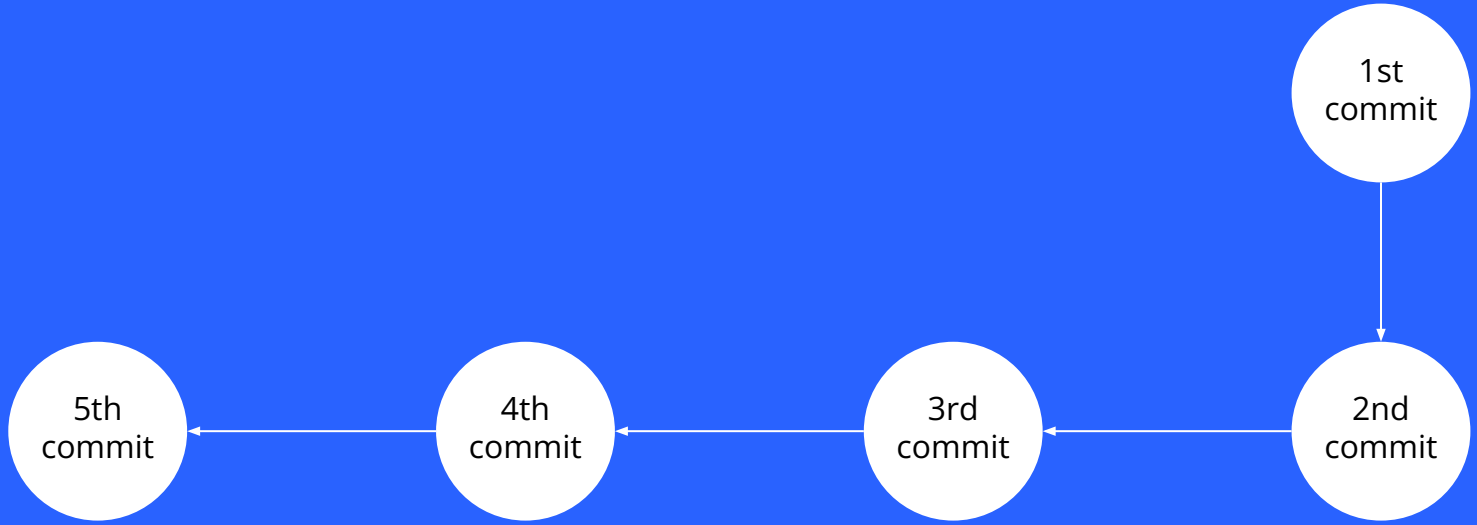
Staging & Committing



1. Staging & Committing



Staging & Committing



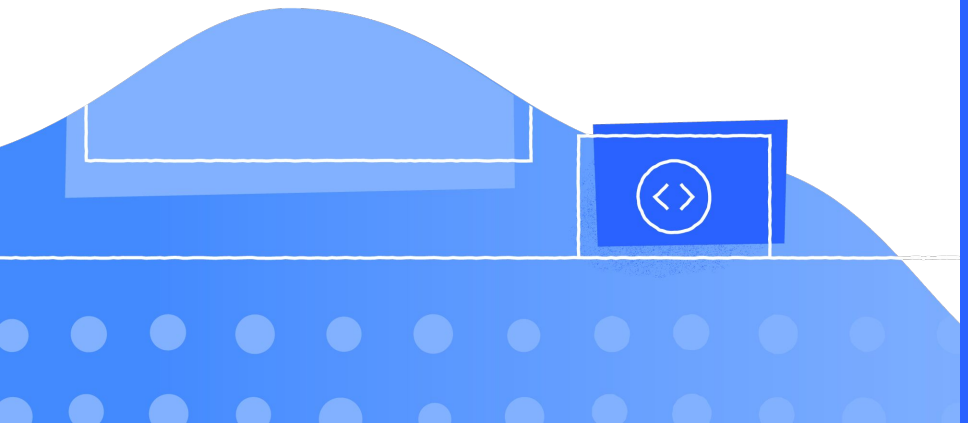
Hands-on activity: ***add****ing* & ***commit****ting (~5 min)*

Instructions (ensure you are in the directory we downloaded last time)

1. Type **git status**
    - Git should tell you that everything is up to date
2. On **your** computer, make a change to any of the files and save that file
3. Now again type **git status**
    - You should see that git saw the change you just made
4. Type **git add <your filename>**
    - This will “stage” or in other words “prepare” your file for commiting
5. Type **git commit -m “<whatever message you want goes here :)>”**
    - This will tell git to “save” your file modifications in its version history
6. Type **git log** to see a list of your recent commits
7. (Optional) type **git push** to “push”, or in other words upload, your changes to your github

Hands On!

10 minutes

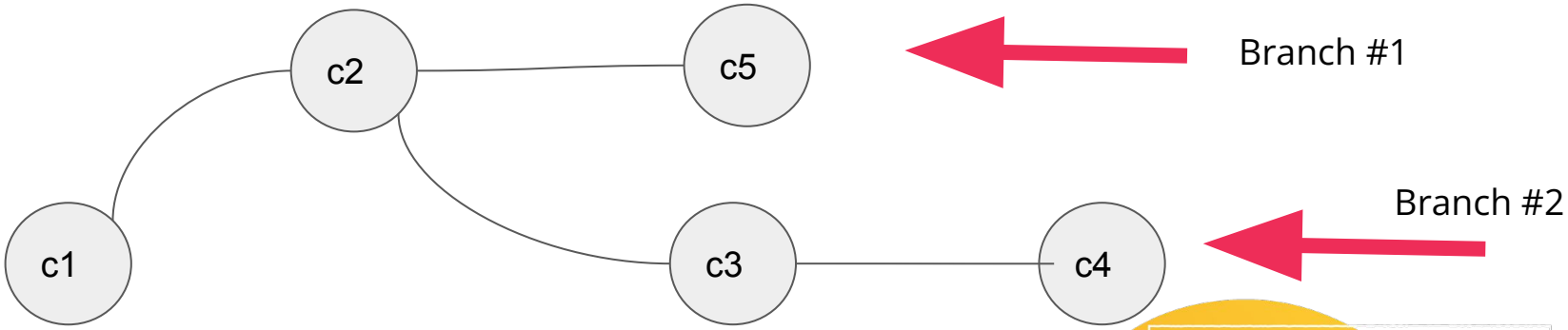


$ cd into the repository
$ Open index.html and make some changes
$ git status
$ git add index.html
$ git commit -m “msg”
$ git log

Git Branching and Merging

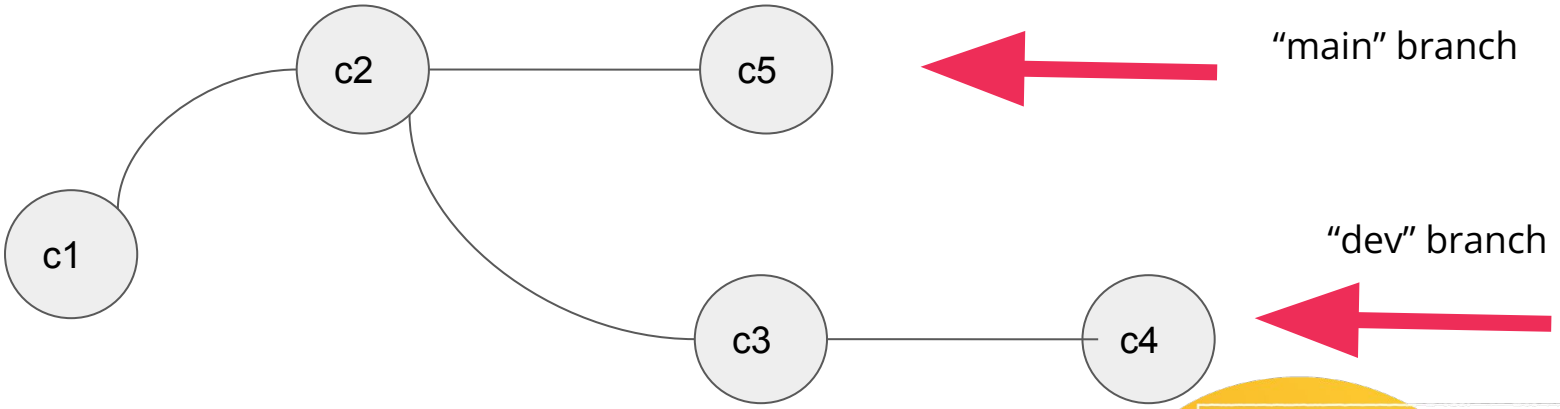
Git branches

- Think of Git's version history as a tree (the weird CS type of tree)
- Git branches are basically "independent lines/sections of development"



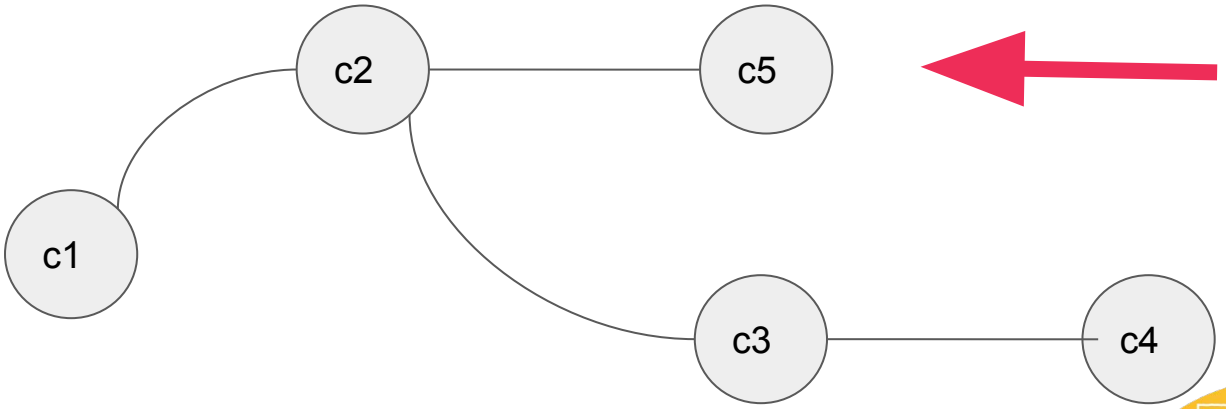
Why use Branches?

Example: a different branch used for different versions of a codebase



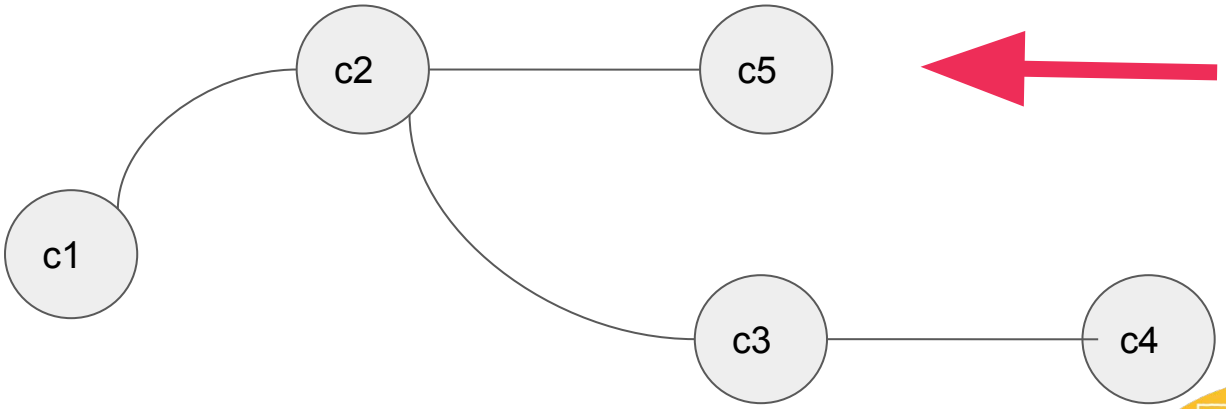
Another example

Using a separate branch to develop a new feature.



How are branches used? Continued

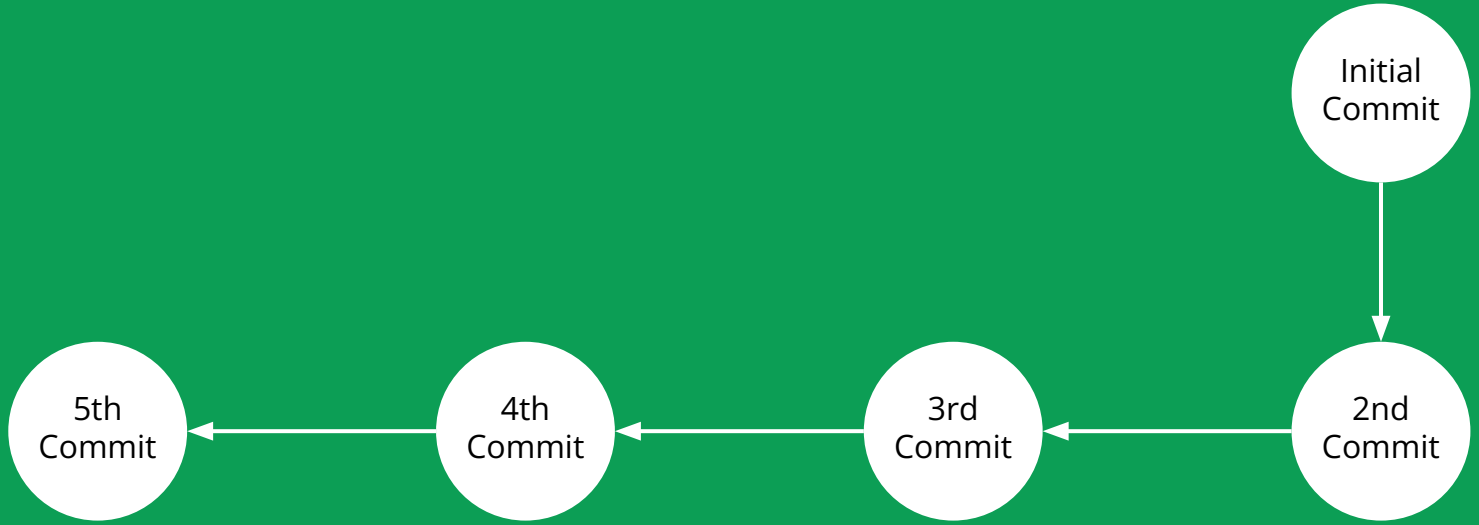
Example: a different branch used for different versions of a codebase



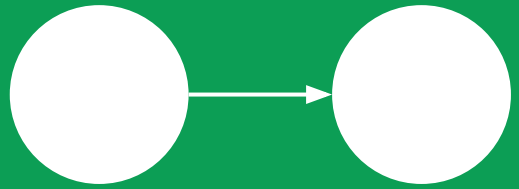
“Production” branch

“Development” branch

Branches



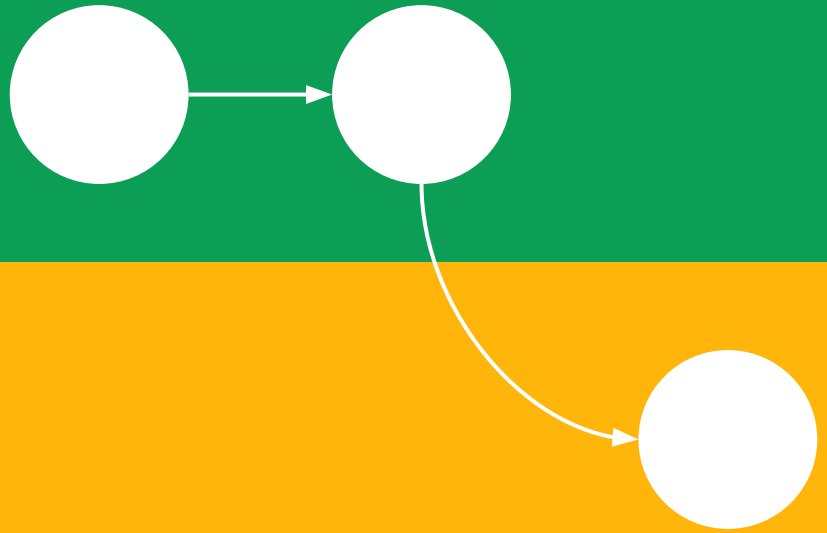
2. Branches



main

new branch

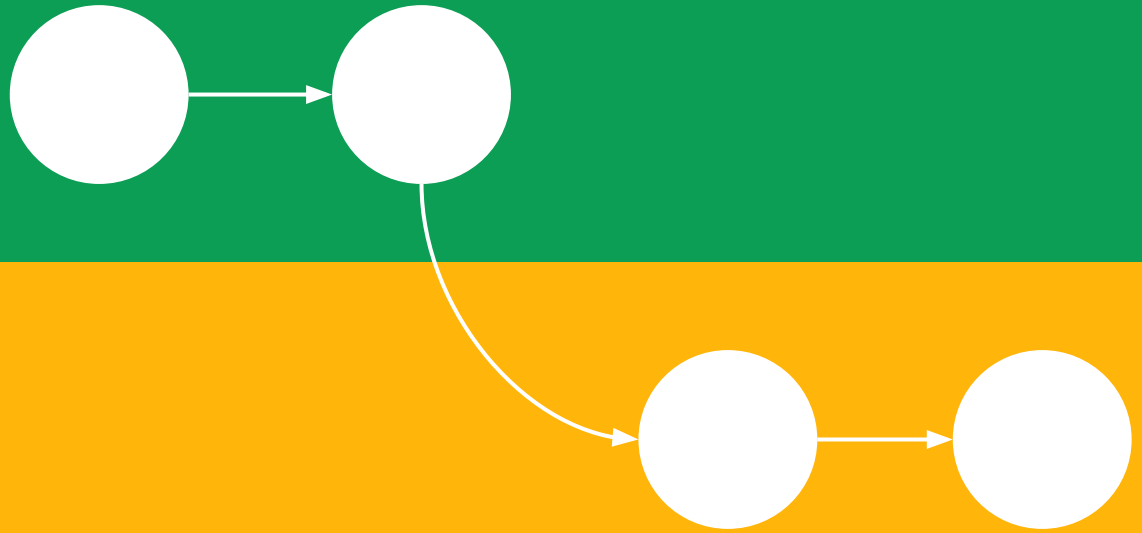
2. Branches



main

new branch

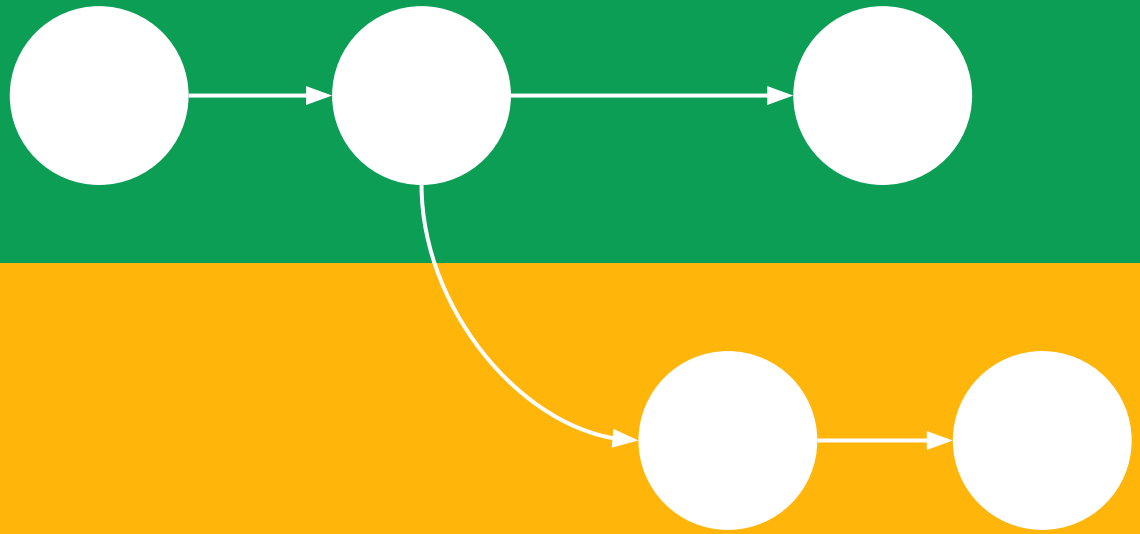
2. Branches



main

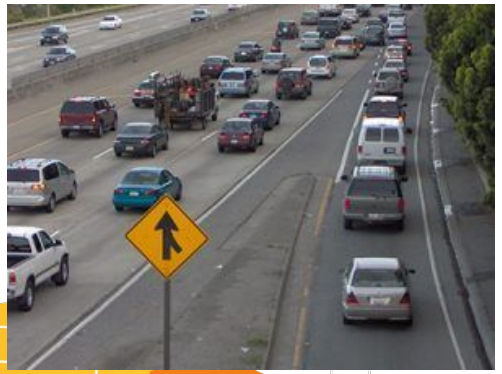
new branch

2. Branches

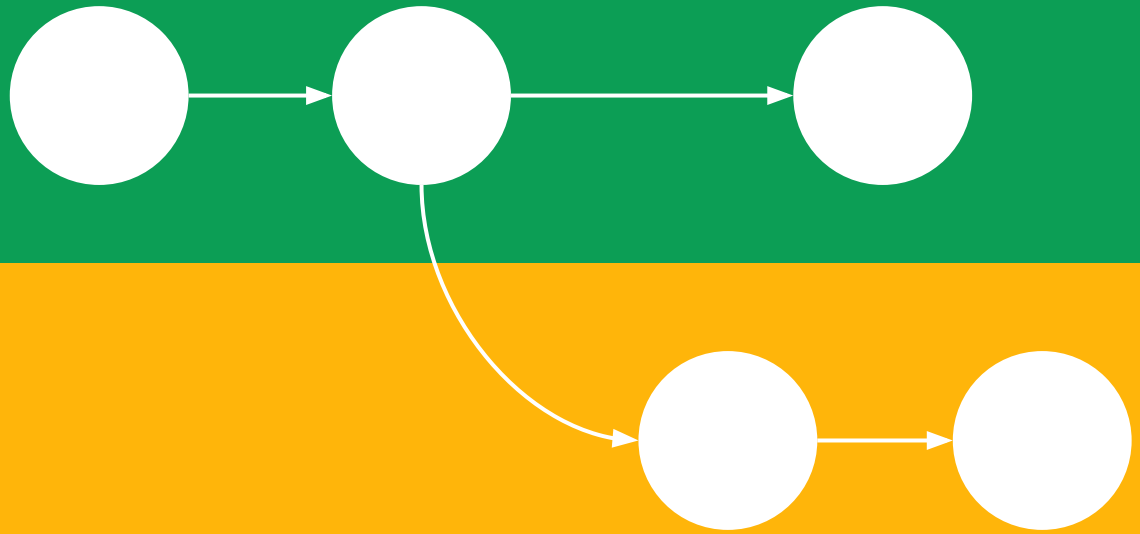


Merging

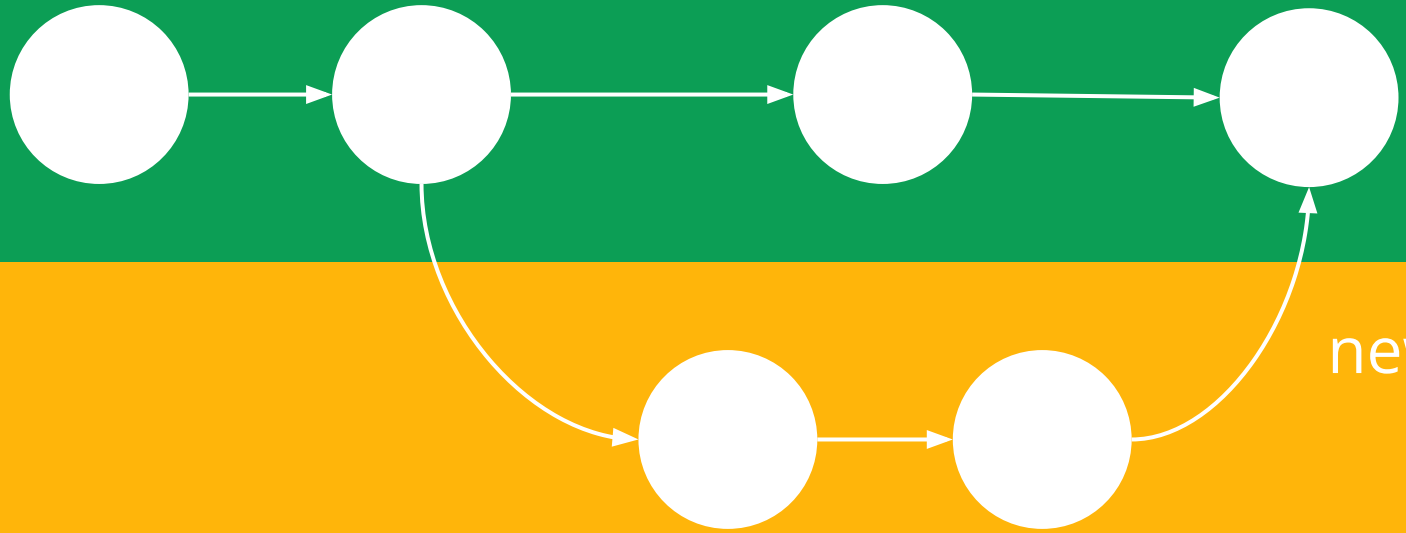
- To “merge” your current branch with another you use the command **git merge <other branch name>**
- Git will intelligently combine all changes from that one branch with another
- *foreshadowing*: Git might have some trouble doing this sometimes :(



Merging



Merging



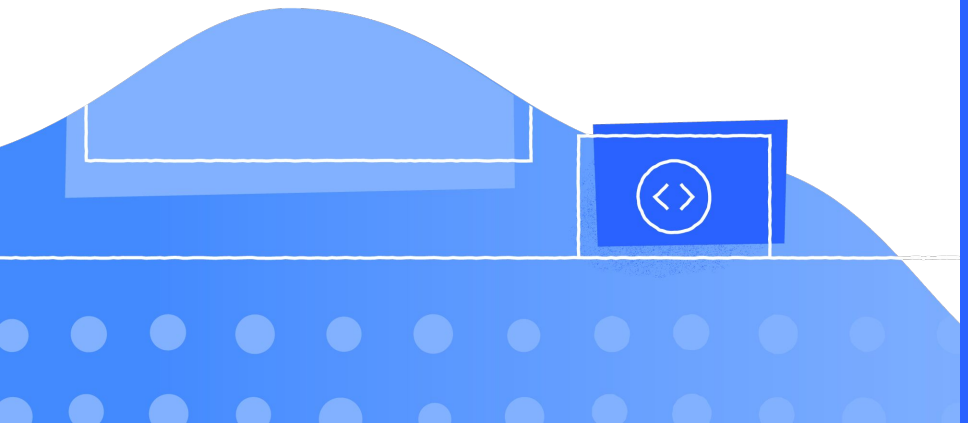
Hands-on activity: *Branching and Merging*

Instructions (ensure you are in the directory we used last time)

1. Type **git branch**
    - This will tell you what branch you are currently using
2. Type **git branch <some name>** to create a new branch
3. Type **git switch <some name>** to switch to this new name (redo step 1 to confirm this!)
    - *Note: git checkout <some name> does the same thing but is the old way of doing it*
4. Make some change to any file and **git add <file name>** and **git commit <file name>** that file
    - Note that these changes are **only** being committed to branch <some name>
5. Type **git switch main**
6. Type **git merge <some name>** (remember that main is the name of the other branch)

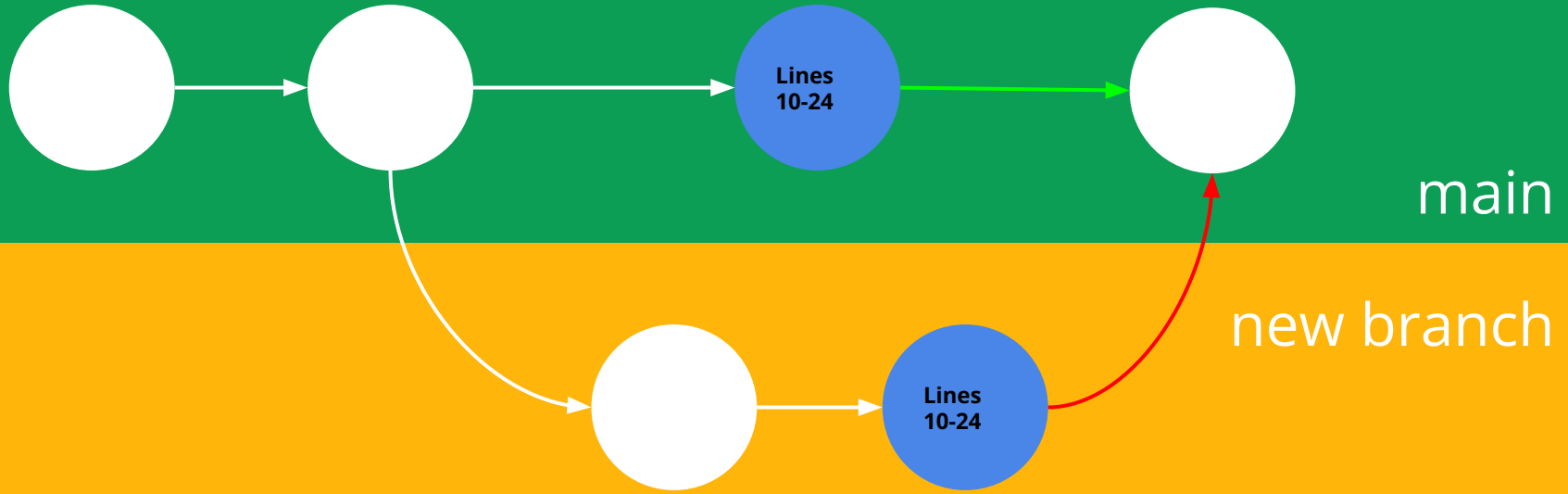
Hands On!

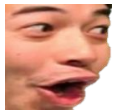
10 minutes



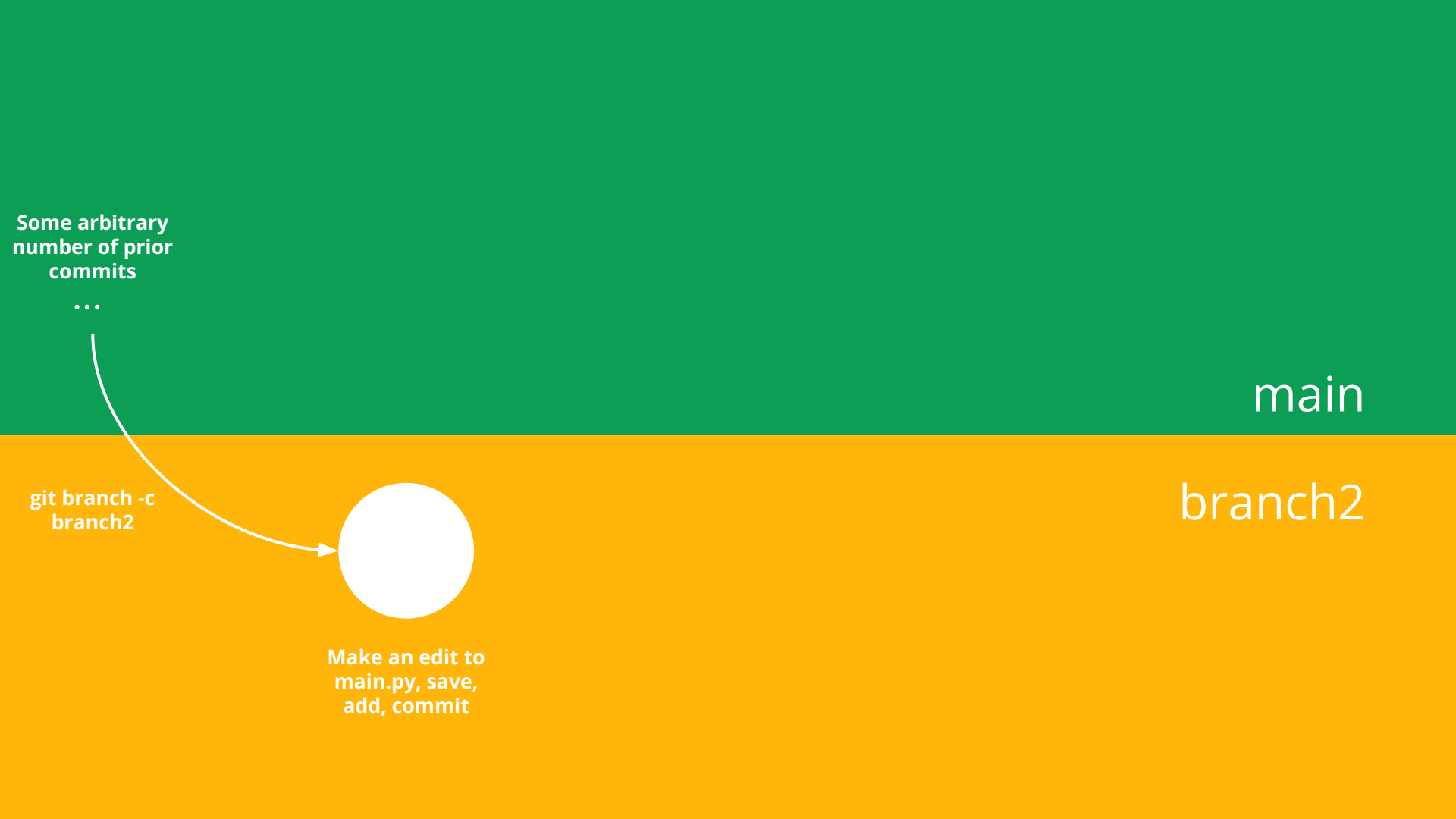
- $ git branch demo
- $ git checkout demo
- $ change lines 10-24, add, commit, log
- $ git checkout master
- $ make different changes on lines 10-24, add, and commit, log - notice how our last commit isn’t there?
- $ git merge demo
- $ uh-oh

Merge conflict when Merging goes wrong

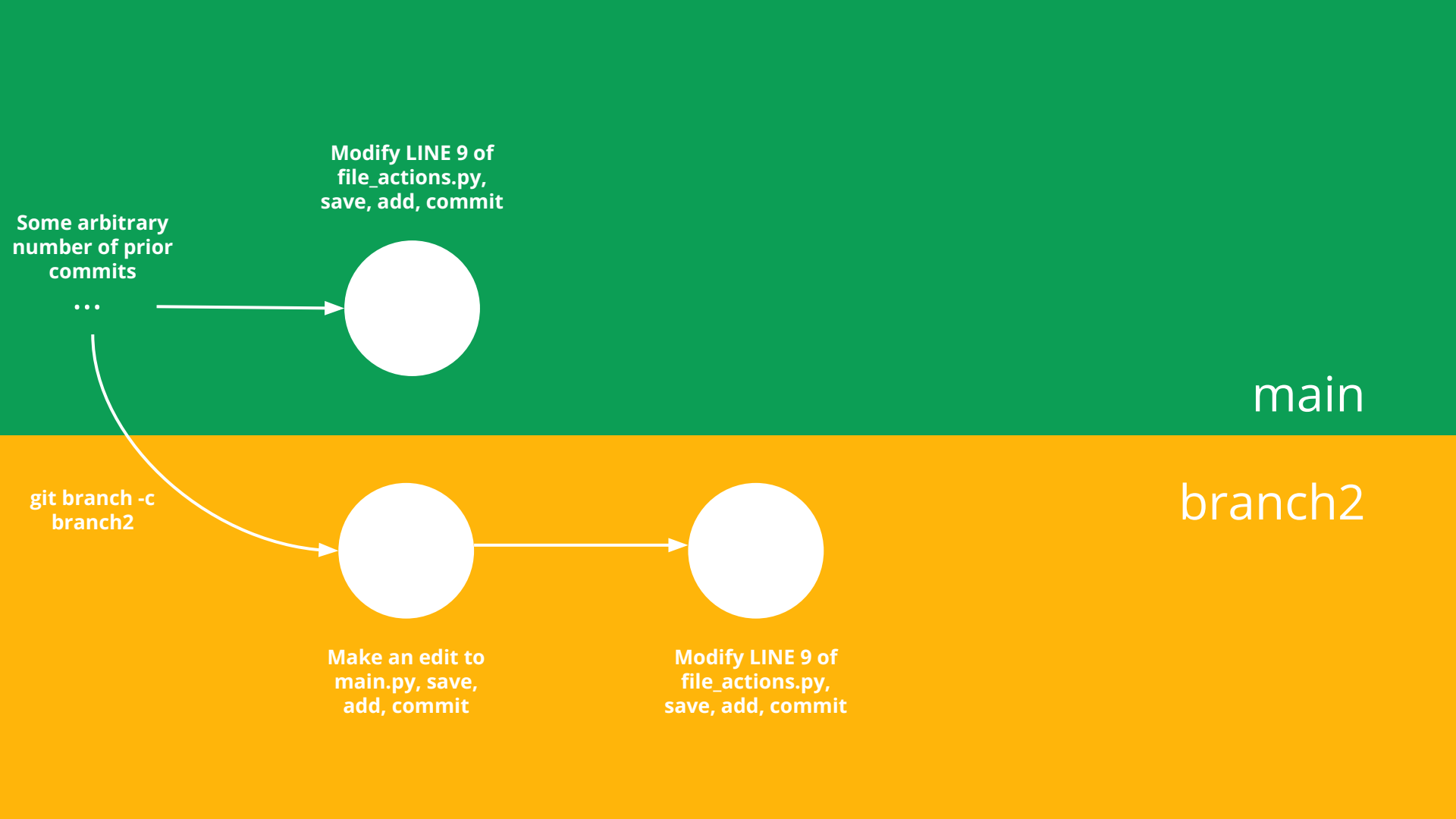


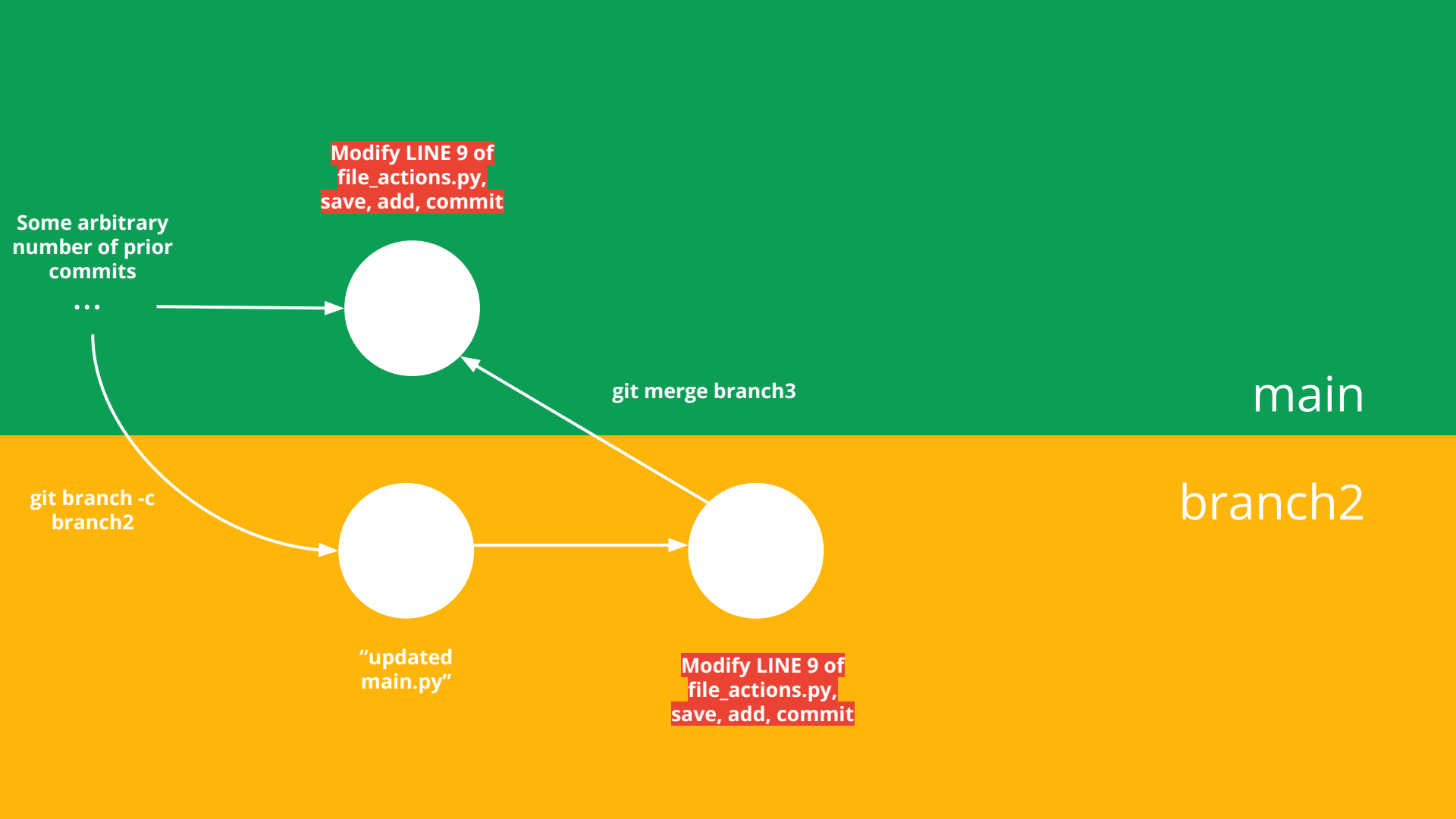
Merge Conflict Demo 

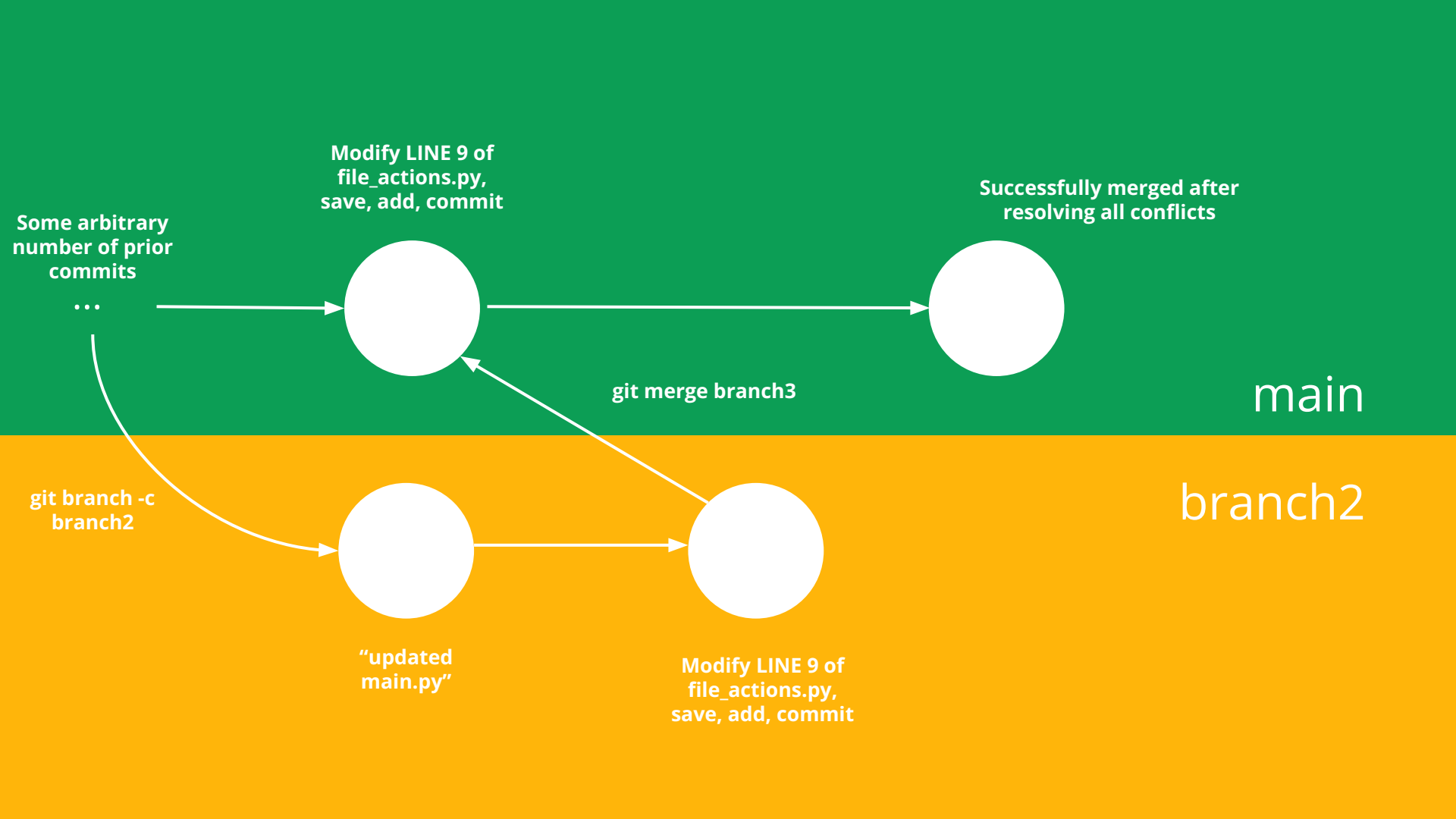
Google Developer Student Clubs
University of Toronto Mississauga

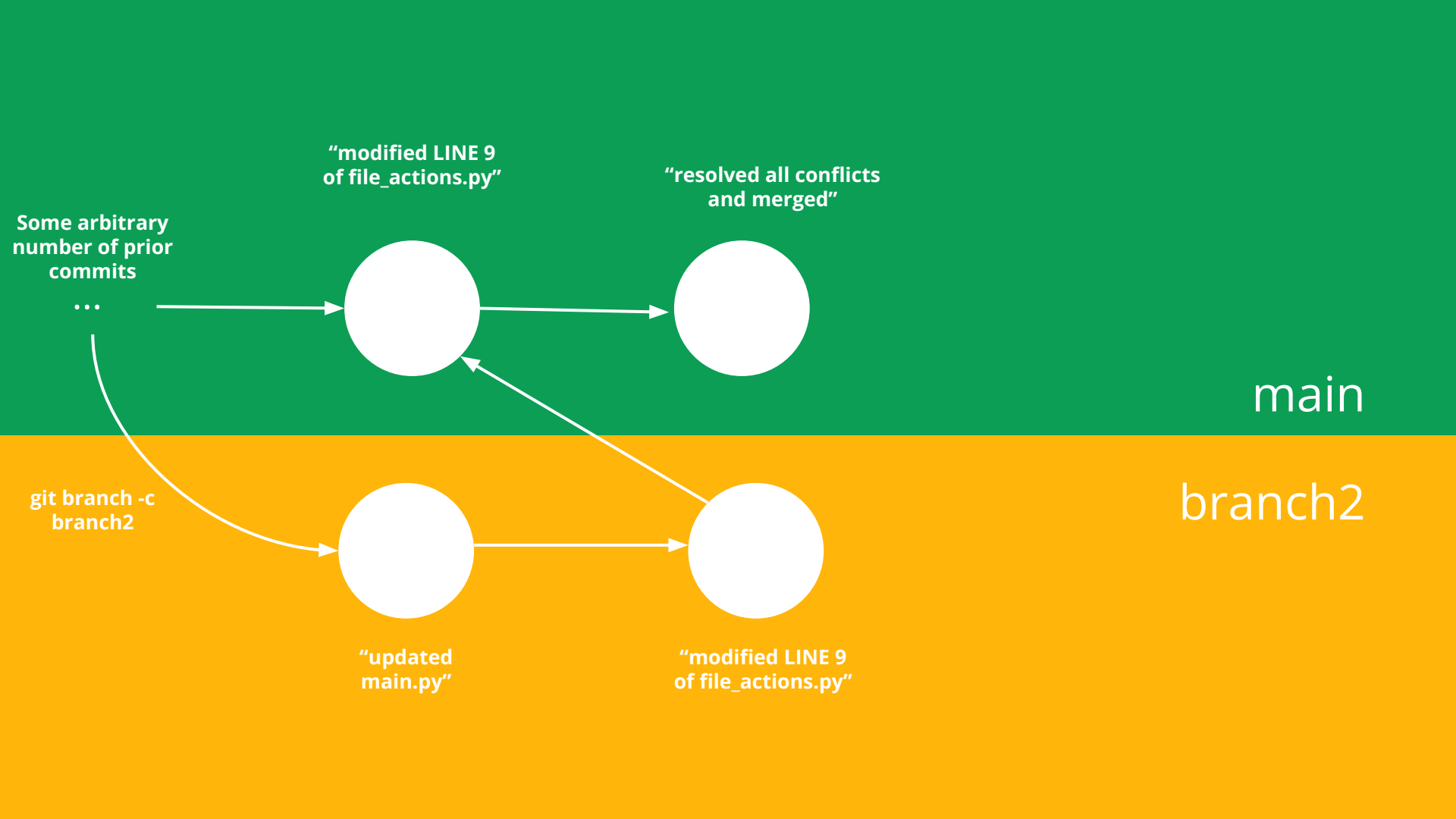

Some arbitrary number of prior commits
...
git branch -c branch2
Make an edit to main.py, save, add, commit
main
branch2


Modify LINE 9 of file_actions.py, save, add, commit
Some arbitrary number of prior commits
...
main
git branch -c branch2
branch2
Make an edit to main.py, save, add, commit


Modify LINE 9 of file_actions.py, save, add, commit
Some arbitrary number of prior commits
...
main
git branch -c branch2
branch2
Make an edit to main.py, save, add, commit
Modify LINE 9 of file_actions.py, save, add, commit

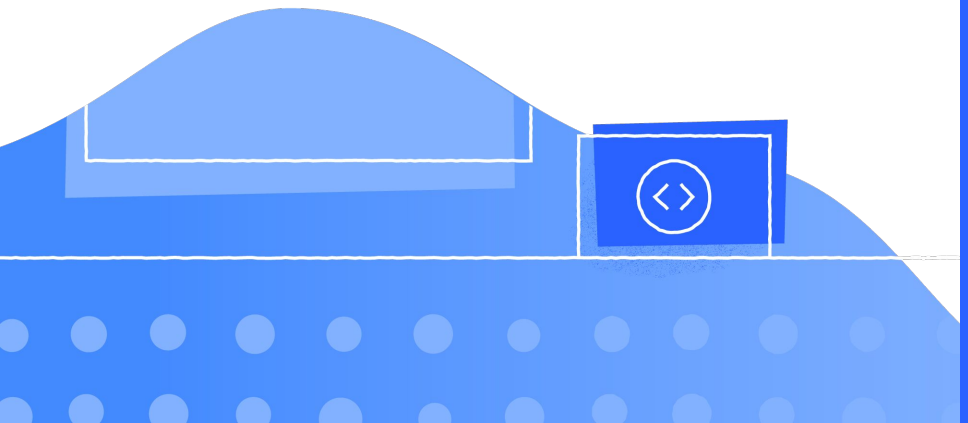

Modify LINE 9 of file_actions.py, save, add, commit
Some arbitrary number of prior commits
...
git merge branch3
main
git branch -c branch2
branch2
"updated main.py"
Modify LINE 9 of file_actions.py, save, add, commit


Modify LINE 9 of file_actions.py, save, add, commit
Successfully merged after resolving all conflicts
Some arbitrary number of prior commits
...
git merge branch3
main
git branch -c branch2
branch2
"updated main.py"
Modify LINE 9 of file_actions.py, save, add, commit


"modified LINE 9
of file_actions.py"
"resolved all conflicts
and merged"
Some arbitrary
number of prior
commits
...
main
git branch -c
branch2
branch2
"updated
main.py"
"modified LINE 9
of file_actions.py"

Merge Conflict

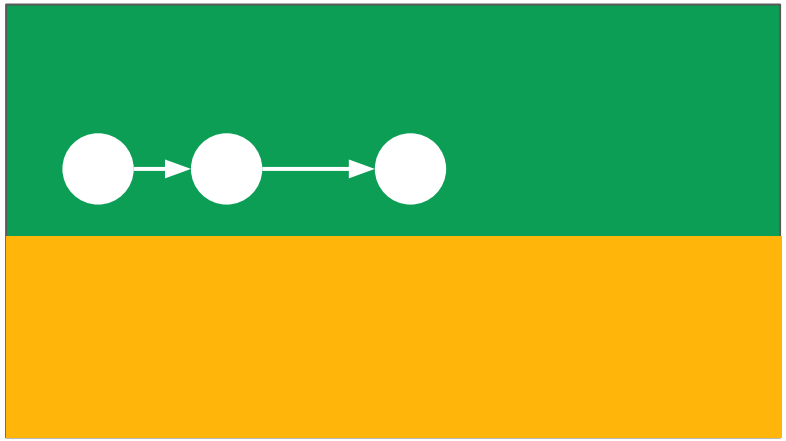
5 minutes



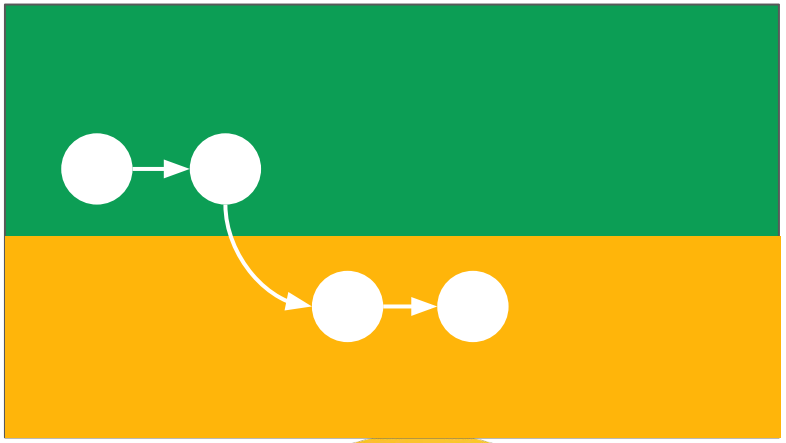
$ open up the index.html file and scroll to line 10
$ You should see some text generated by git
$ Fix the conflict.
$ git add index.html
$ git commit -m “merged”

How does this all work with GitHub?

GitHub Repository

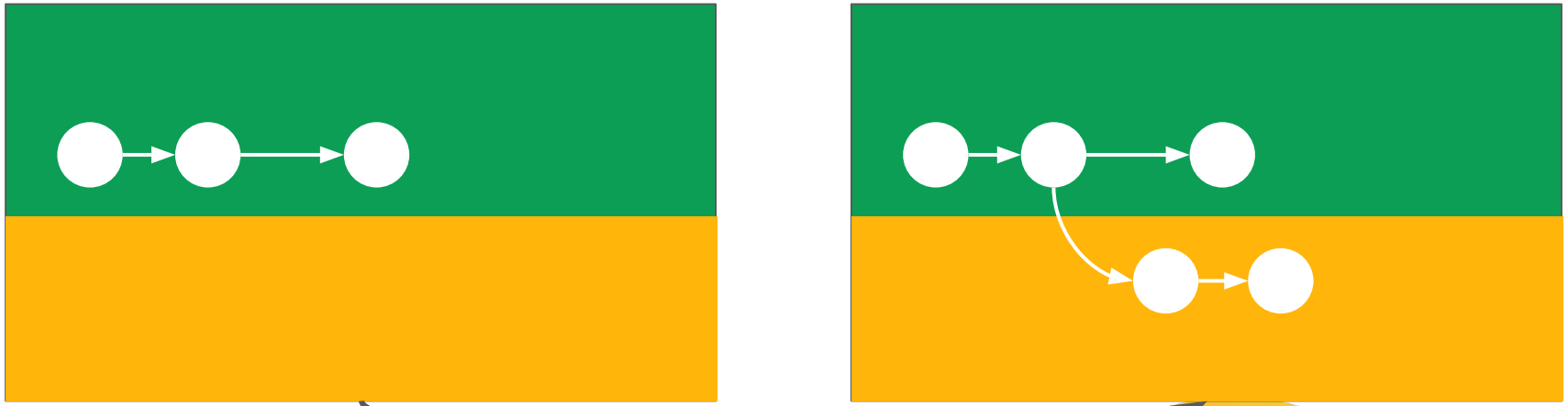


Local Repository



GitHub Repository

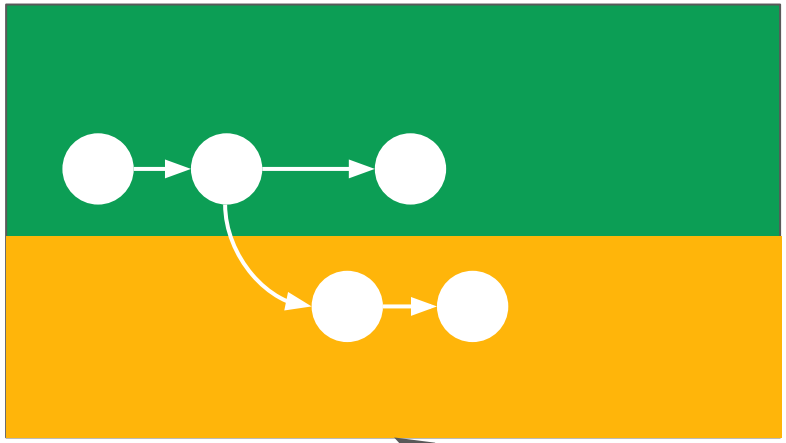
Local Repository

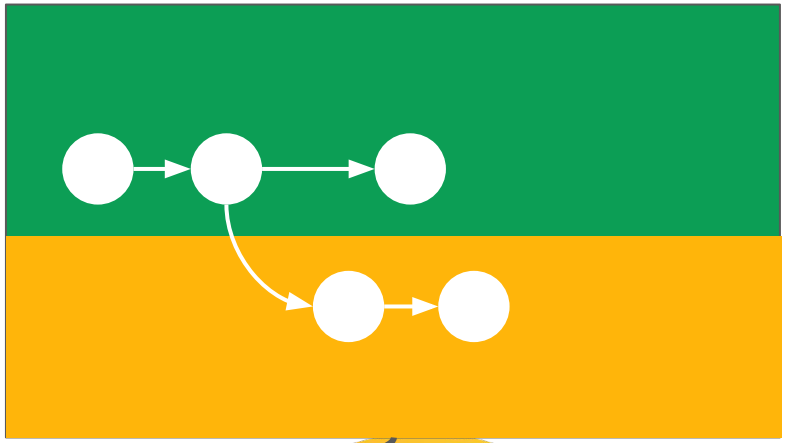


git pull

GitHub Repository

Local Repository

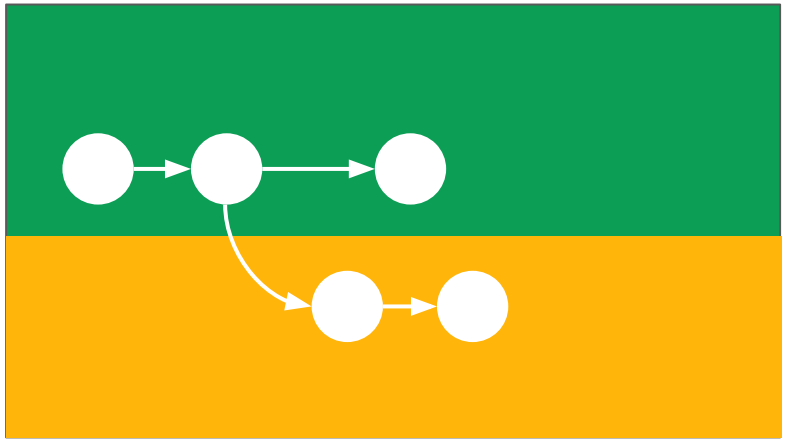




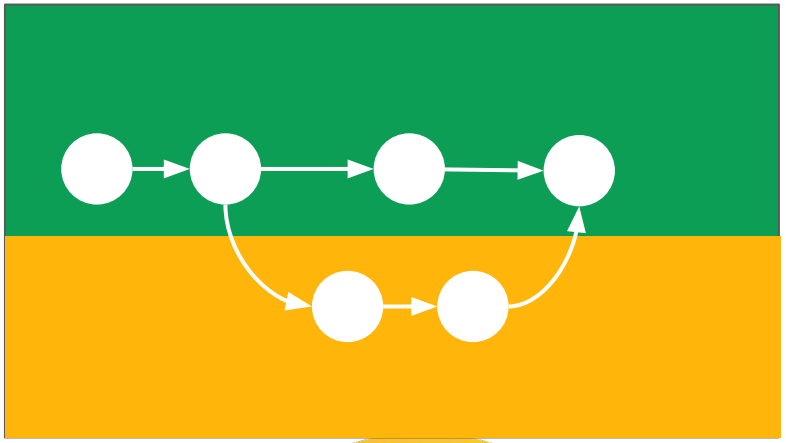
git push

Google Developer Student Clubs
University of Toronto Mississauga

GitHub Repository



Local Repository

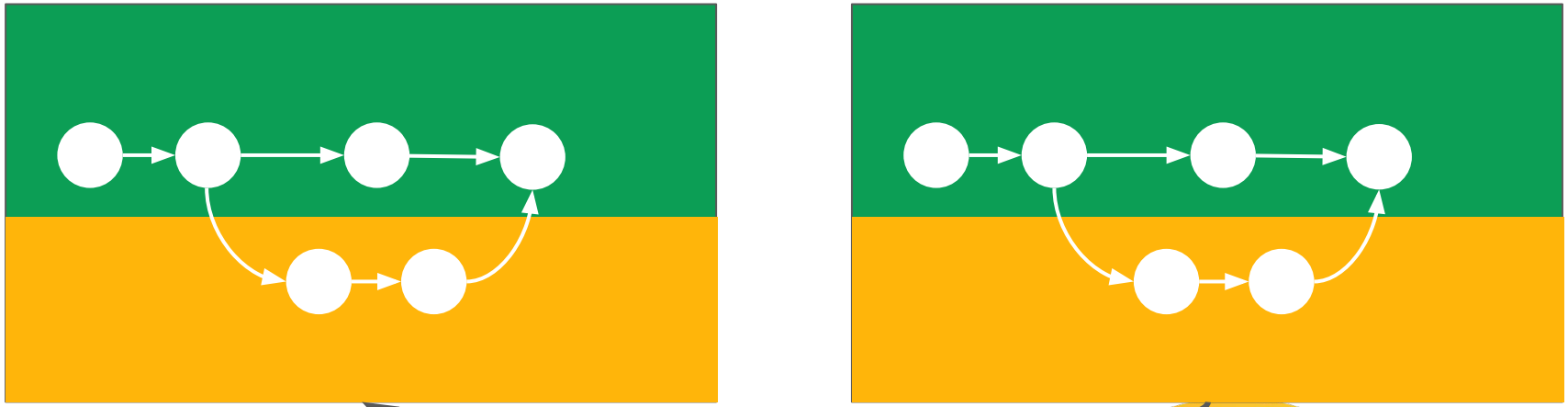


git merge <branch name>

Google Developer Student Clubs
University of Toronto Mississauga

GitHub Repository

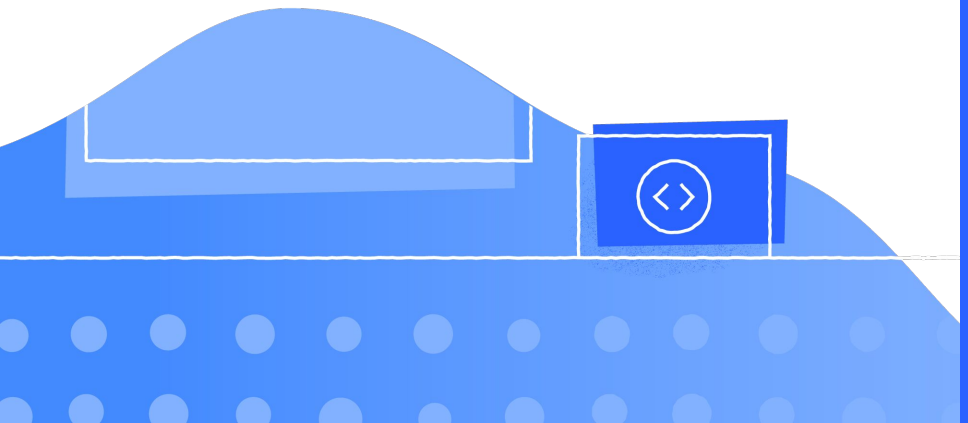
Local Repository



git push

Google Developer Student Clubs
University of Toronto Mississauga

What we talked about today



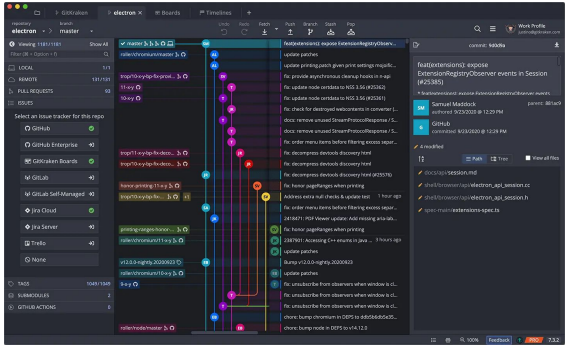
1. Cloning/Forking
2. Staging & Committing
3. Branches
4. Merging
5. Pushing & Pulling

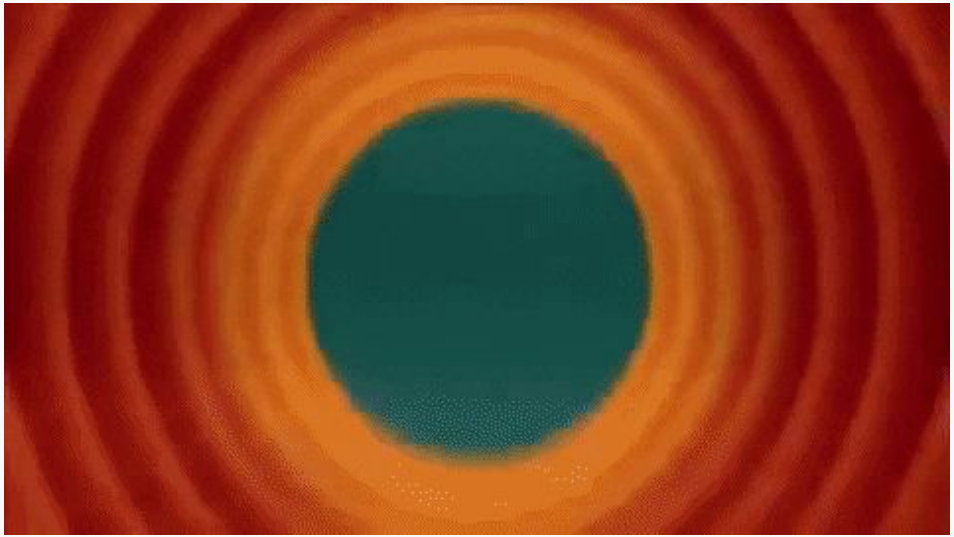
Other useful Git commands, concepts, and tools

- Rebasing:

  https://www.youtube.com/watch?v=7Mh259hfxJg
- Making pull requests:

  https://www.youtube.com/watch?v=8lGpZkjnkt4
- Git kraken: a GUI for Git
  - https://www.gitkraken.com/





Thank you all so much for coming!

Before you go...

- Would you like to join the GDSC team? We are hiring! Please apply here (deadline Friday Sept 24th at midnight)
  - https://docs.google.com/forms/d/e/1FAIpQLSfsTcH9VLxIDb4r3Alv0wWT2s60IKW-fAifeafe5E77BKJCQg/viewform
- What workshops would you like to see in the future? Please let us know here!
  - https://docs.google.com/forms/d/e/1FAIpQLScYpTz67WMiHqXYgH5uPobtSxryoww7yCchmF9goagGFZxEzg/viewform

We hope to see you next time at....

The **“Intro to web development, HTML/CSS, and Javascript workshop!”**

- October 2nd at 4 pm
- Milind will be leading that workshop

Google Developer Student Clubs
University of Toronto Mississauga