INTRO TO VERSION CONTROL

GIT & GITHUB SERIES

VIDEO #1

@LEARNINGWITHJELLY

AGENDA

01

VERSION CONTROL

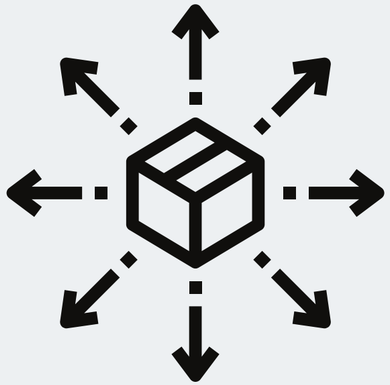
02

GIT

03

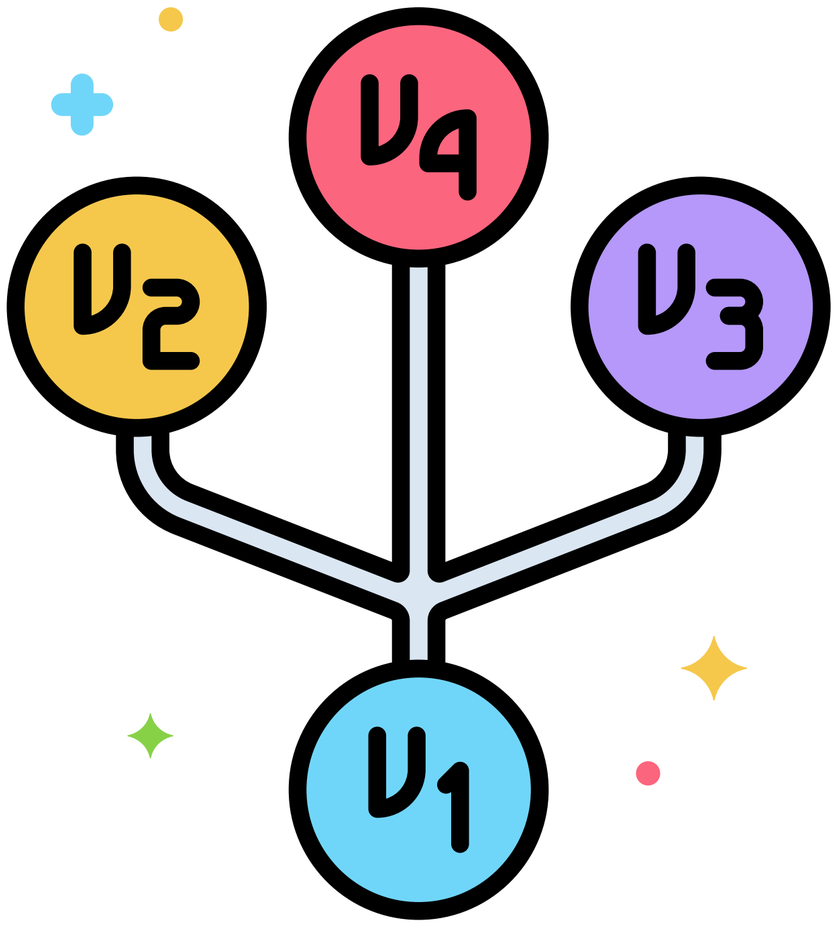
GITHUB

04



DISTRIBUTED COMPUTING

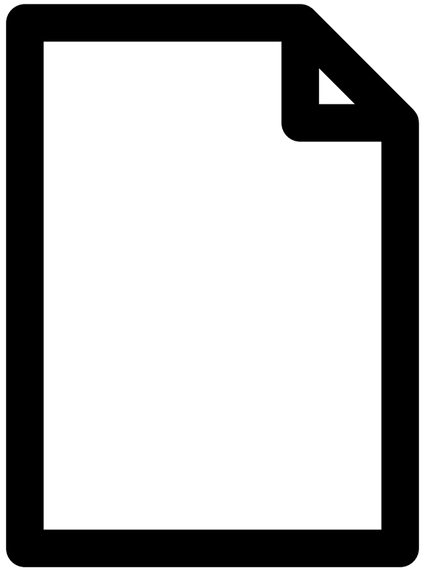
WHAT IS VERSION CONTROL?



VERSION CONTROL

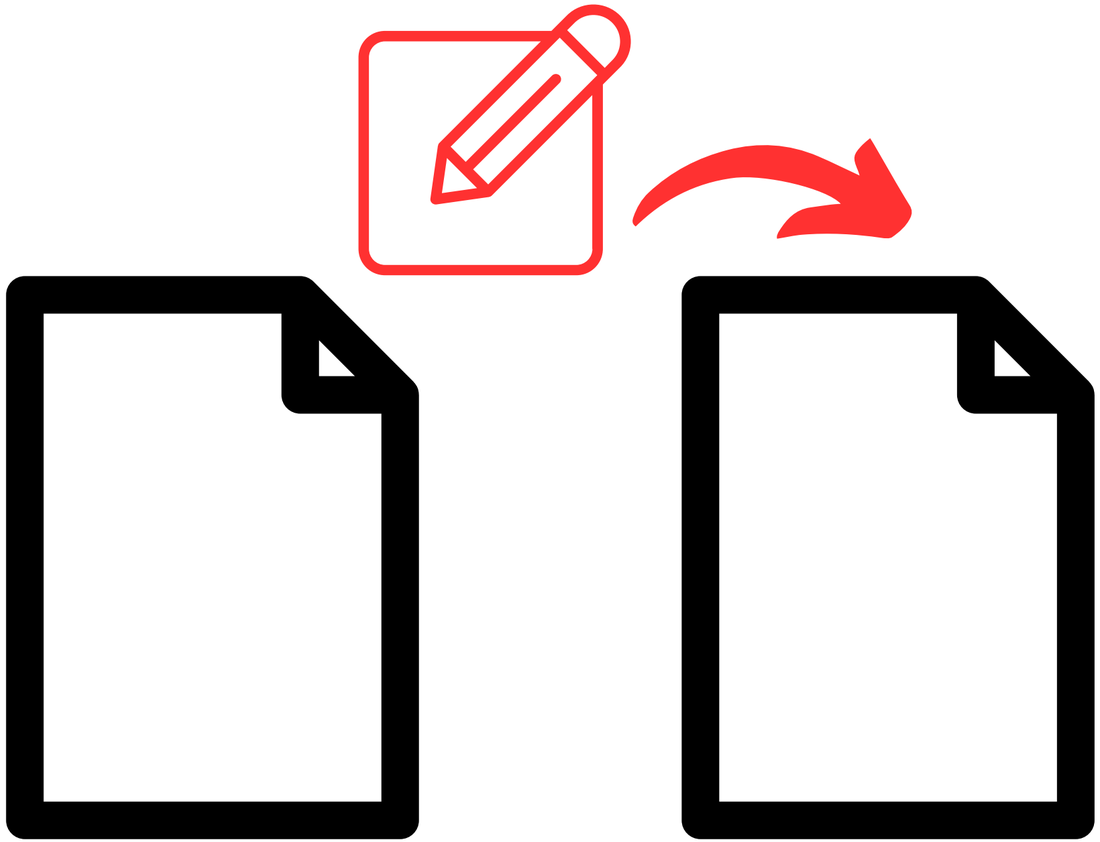
METHOD OF TRACKING AND MANAGING CHANGES TO FILES (OR CODE) ACROSS TIME

PRACTICAL EXAMPLE



ROUGH DRAFT #1

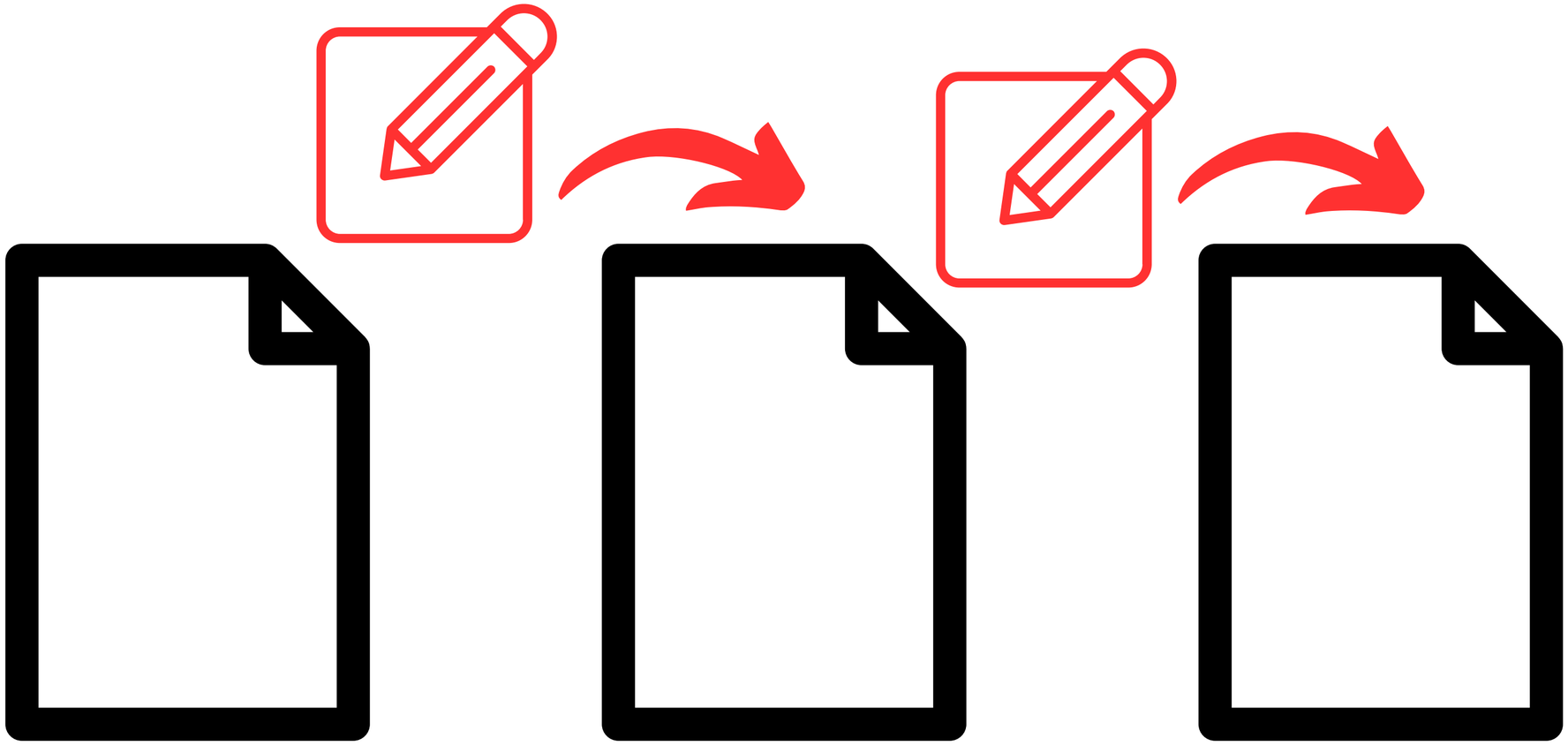
PRACTICAL EXAMPLE



ROUGH DRAFT #1

ROUGH DRAFT #2

PRACTICAL EXAMPLE

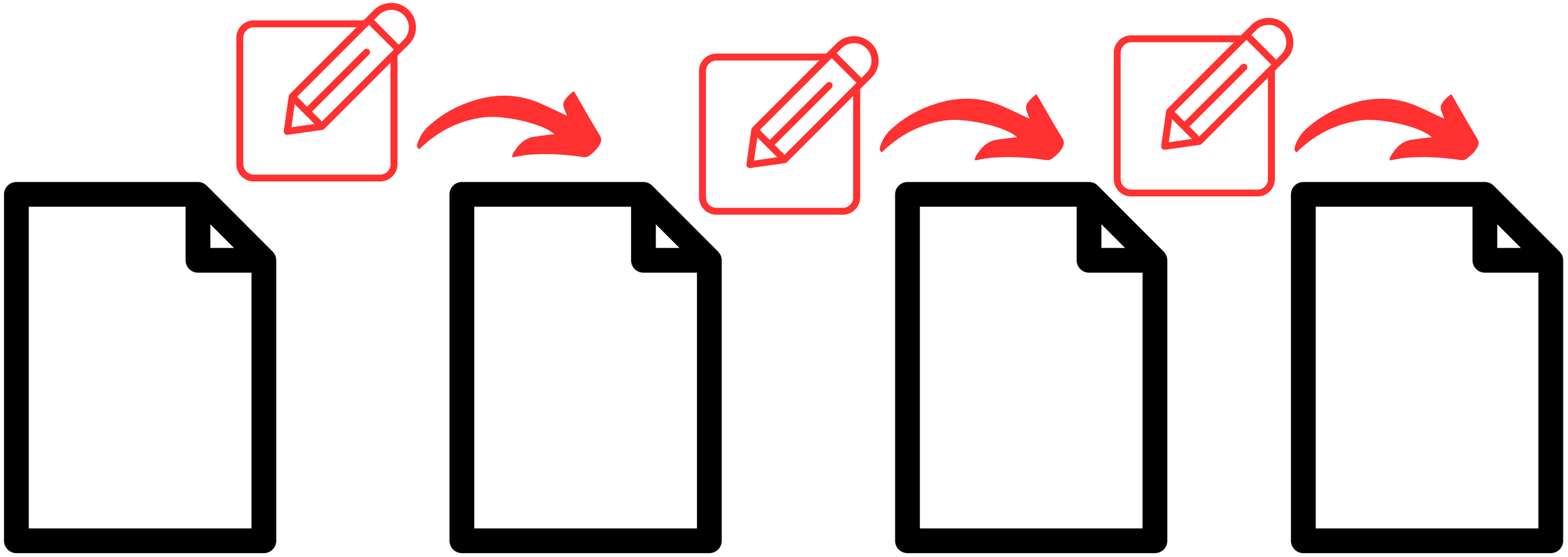


ROUGH DRAFT #1

ROUGH DRAFT #2

ROUGH DRAFT #3

PRACTICAL EXAMPLE



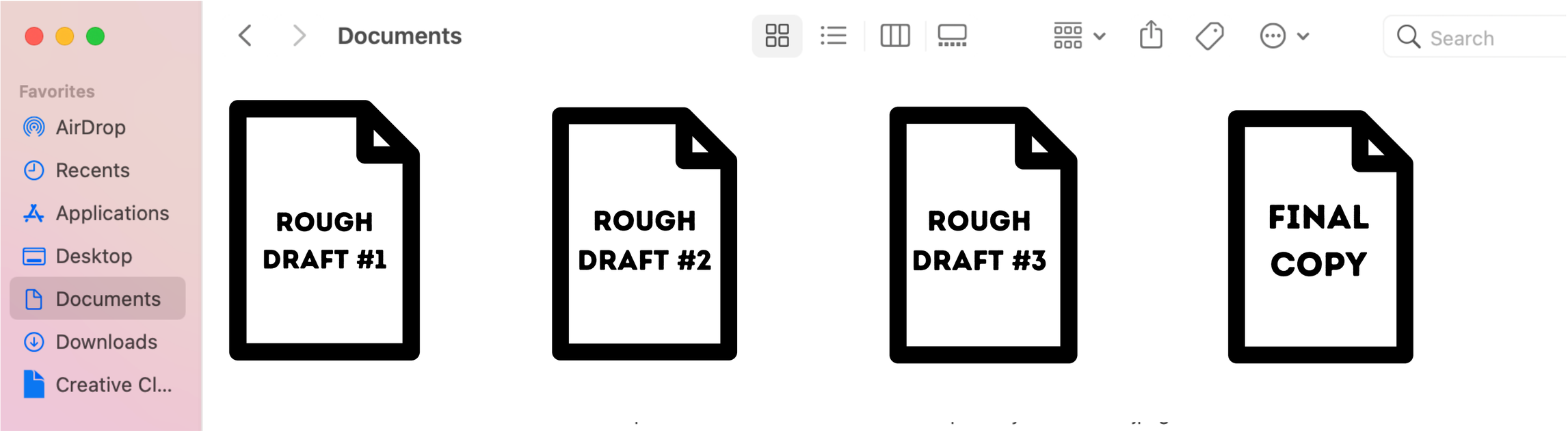
ROUGH DRAFT #1

ROUGH DRAFT #2

ROUGH DRAFT #3

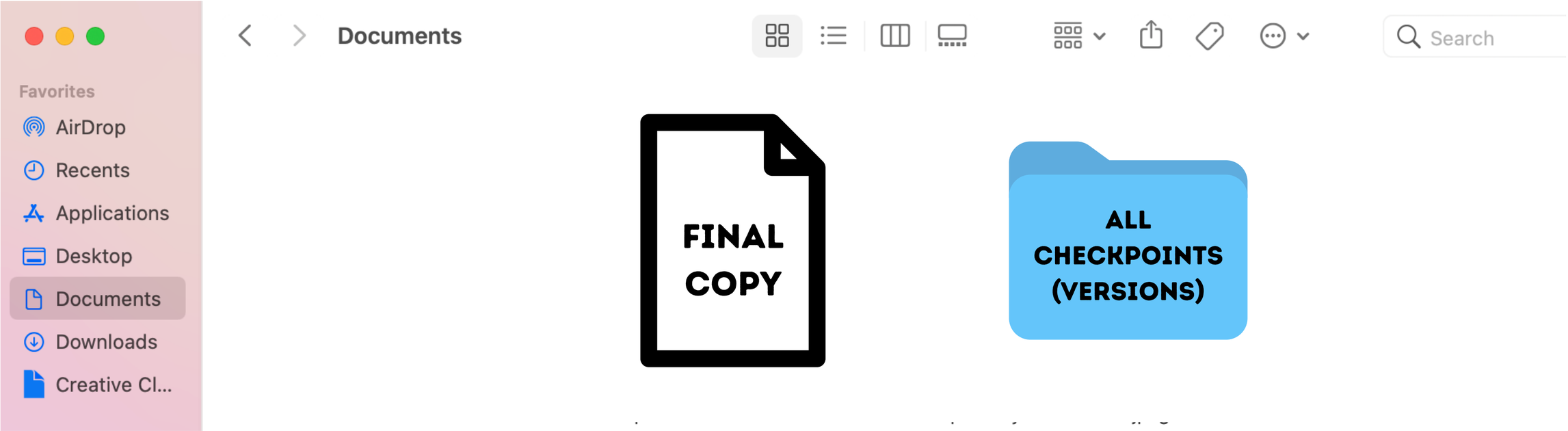
FINAL COPY

PRACTICAL EXAMPLE



VS.

PRACTICAL EXAMPLE



THIS IS WHERE GIT COMES INTO PLAY

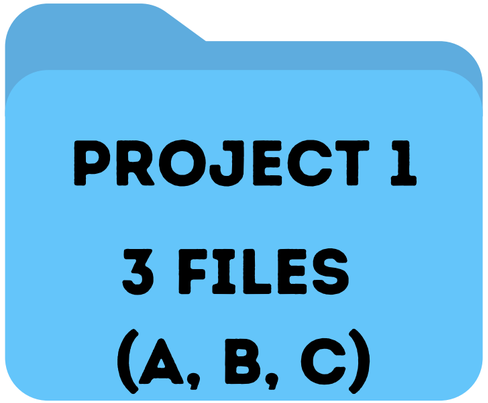
WHAT IS GIT?

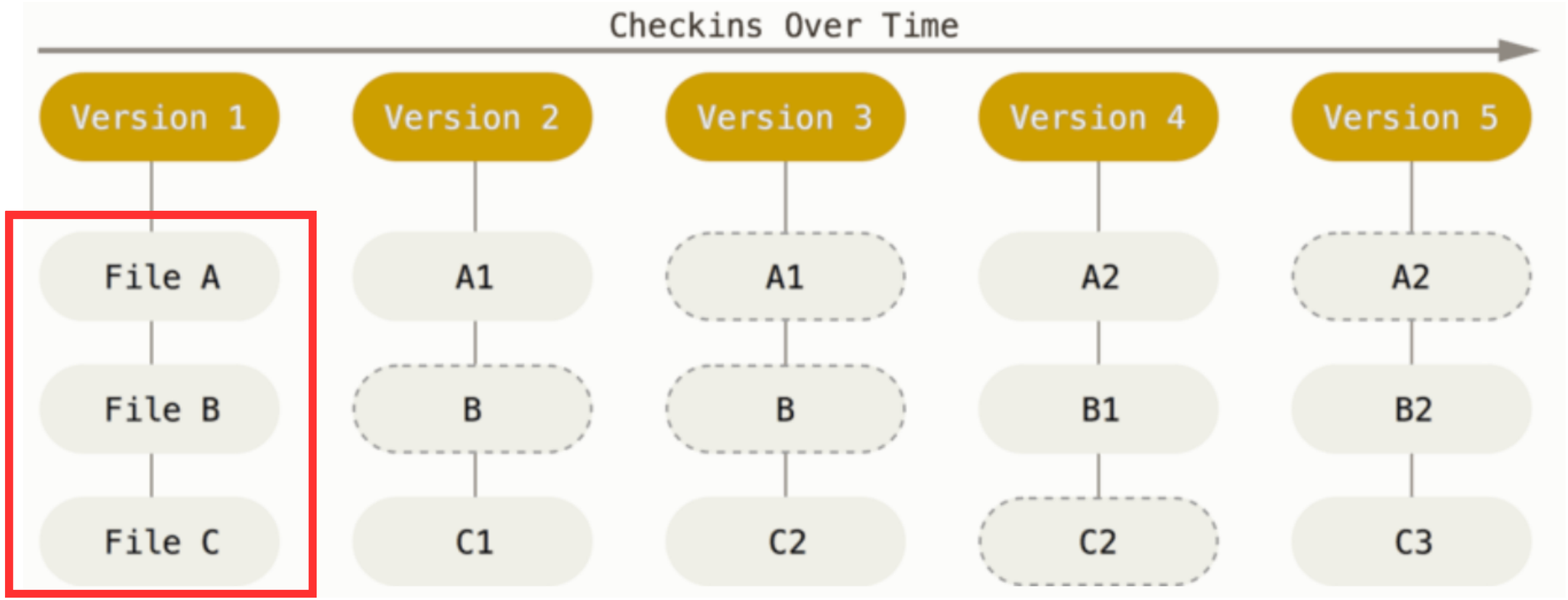
DOCUMENTATION: HTTPS://GIT-SCM.COM/

GIT

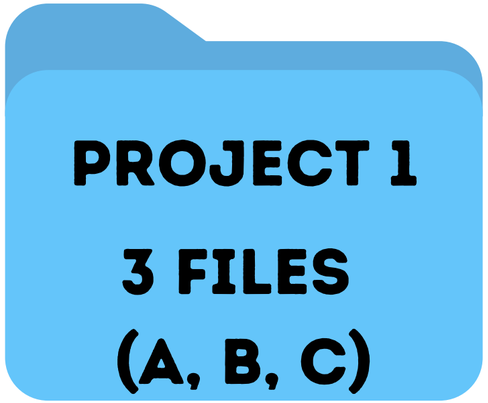
- OPEN SOURCE DISTRIBUTED VERSION CONTROL SYSTEM
- "CODING LANGUAGE" YOU USE ON YOUR LOCAL COMPUTER VIA THE TERMINAL
- TRACKS "SNAPSHOTS" OF YOUR FILE IN TIME (LOCALLY) AND SAVES THESE SNAPSHOTS INSIDE REPOSITORIES (ANY FOLDER CAN BE A GIT REPO!)
- SINCE IT IS LOCAL, ONLY YOU HAVE ACCESS TO YOUR GIT REPOS UNLESS YOU CONNECT IT TO A REMOTE PROVIDER
- IS NOT CONNECTED TO THE INTERNET

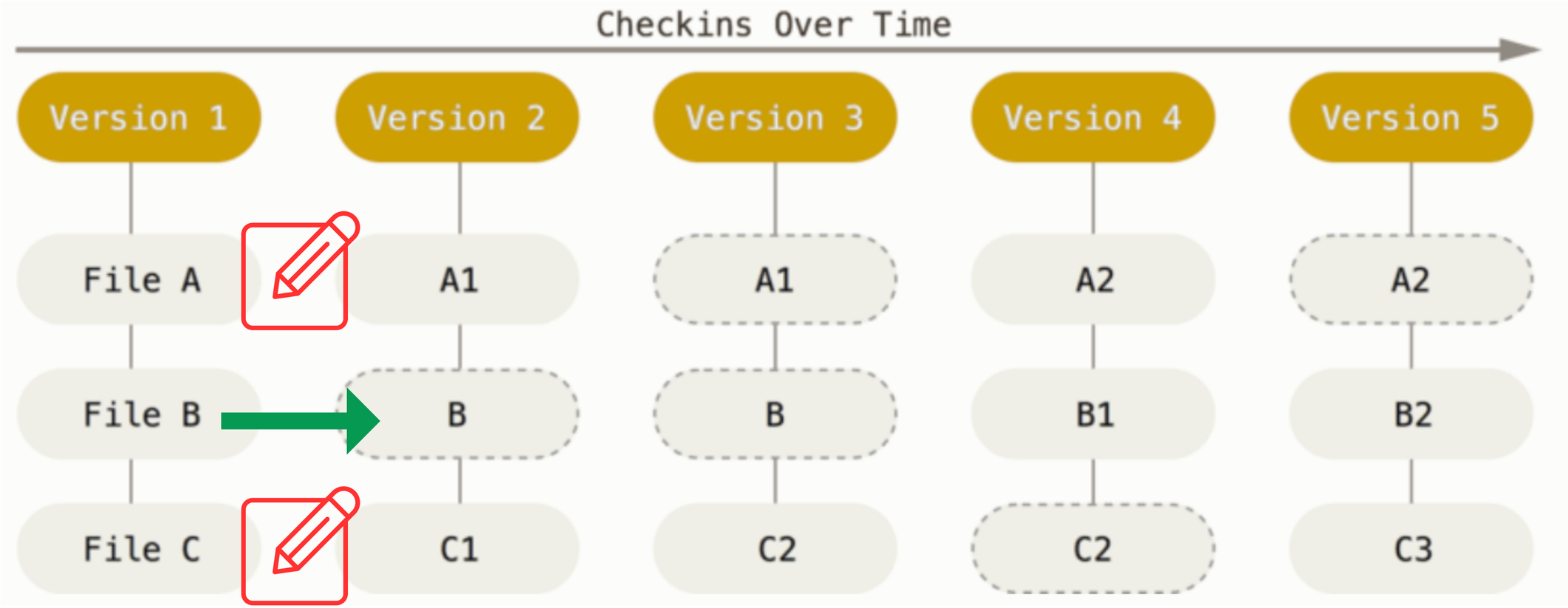
GIT


PROJECT 1
3 FILES
(A, B, C)

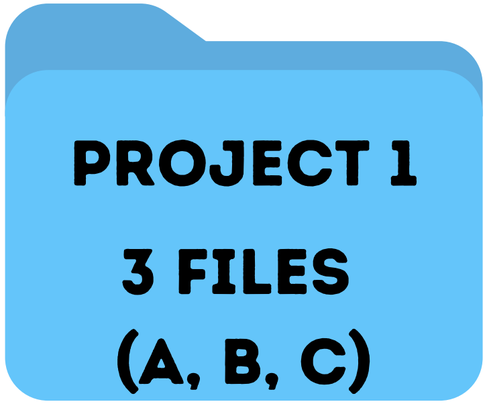

Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3

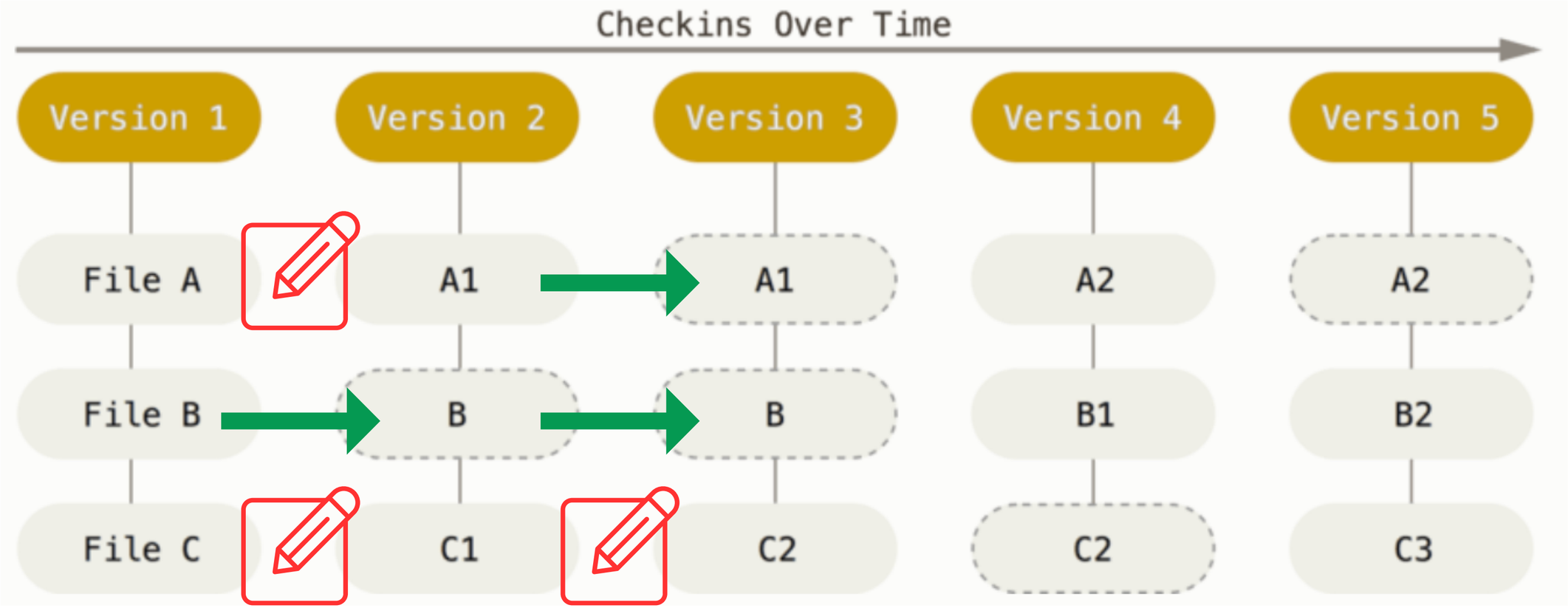
GIT


PROJECT 1
3 FILES
(A, B, C)


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
A1
A1
A2
A2
File B
B
B
B1
B2
File C
C1
C2
C2
C3

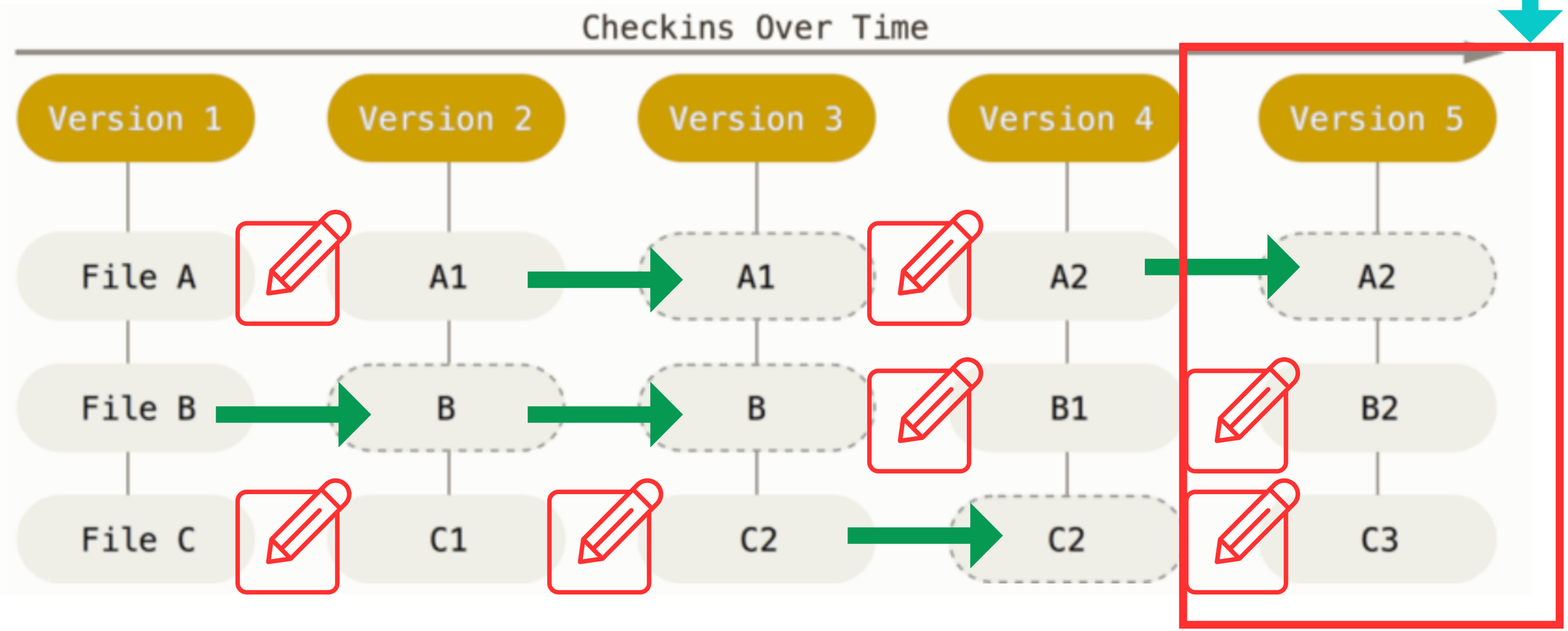
GIT


PROJECT 1
3 FILES
(A, B, C)


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
A1
A1
A2
A2
File B
B
B
B1
B2
File C
C1
C2
C2
C3

GIT

ALWAYS WORKING ON MOST RECENT VERSION BUT YOU CAN GO BACK IF NEEDED!



“

SO WHERE DOES GITHUB COME INTO PLAY?

SHARE FILES,
COLLABORATE,
PORTFOLIO PROJECTS

GITHUB

- REMOTE HOSTING SERVICE FOR VERSION CONTROL USING GIT

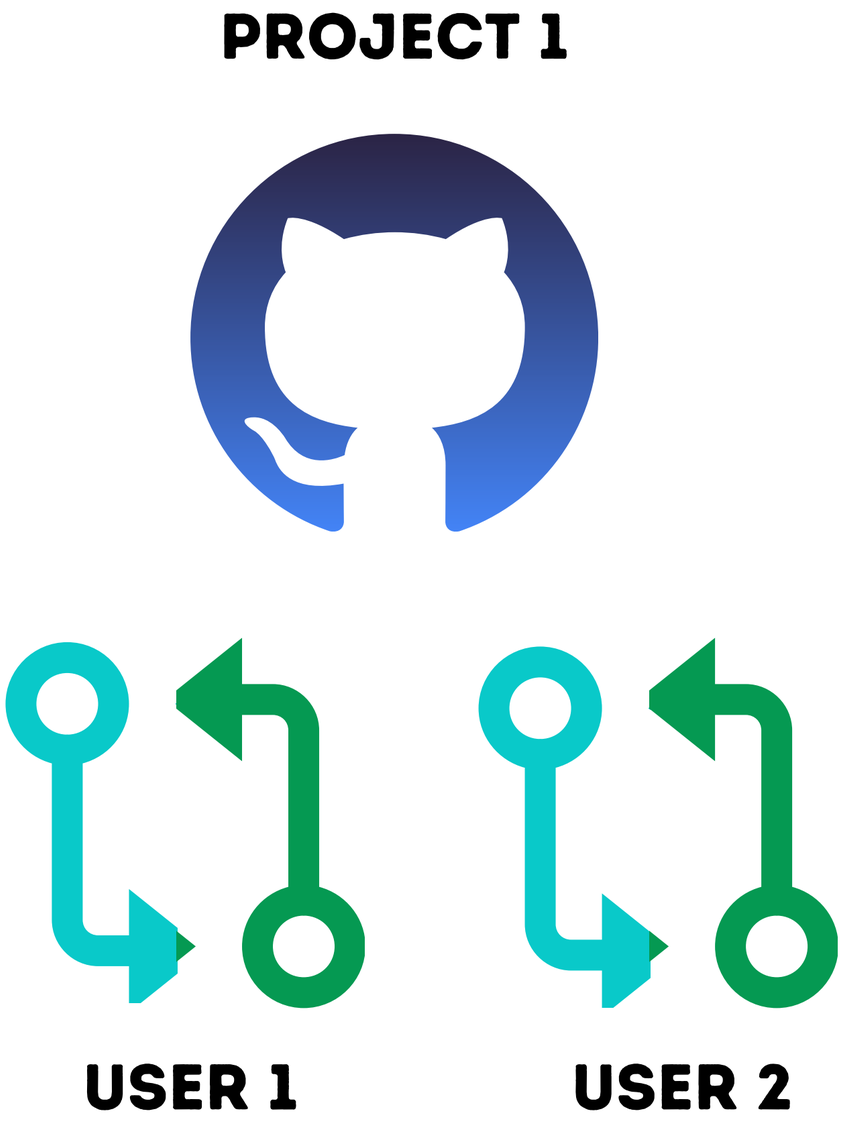
- IT'S REMOTE SO THAT MEANS ON THE INTERNET

- IF IT IS ON THE INTERNET THAT MEANS YOU CAN COLLABORATE/SHARE FILES (REPOS)
  - PUSH UP FILES/CODE (CHANGES TO FILES)
  - PULL DOWN FILES/CODE

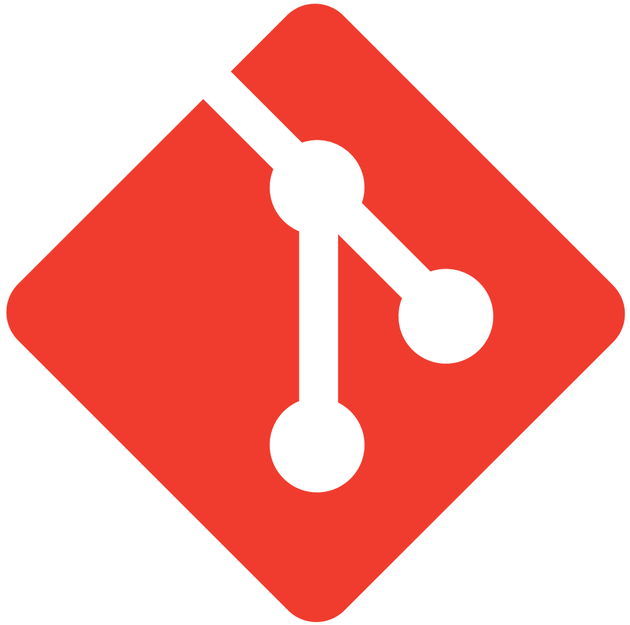
- CONNECT YOUR LOCAL GIT REPOS TO A REMOTE REPO

- SOCIAL NETWORK FOR SOFTWARE DEVELOPERS AND CODERS

- CAN'T BE USED WITHOUT GIT!!!

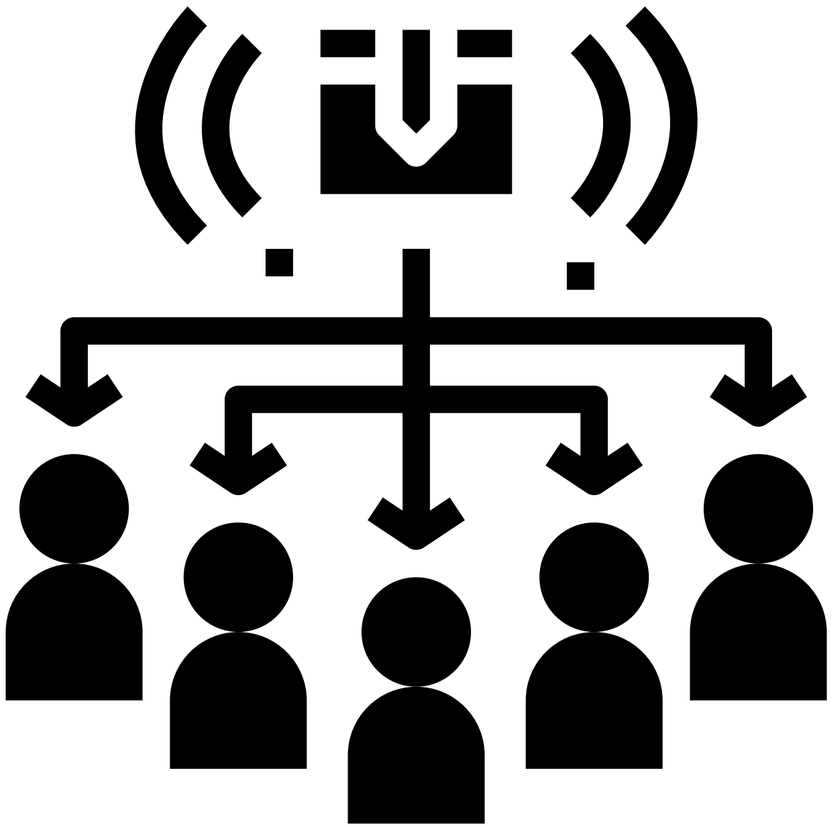


“WHATS THE DIFFERENCE BETWEEN THE TWO?



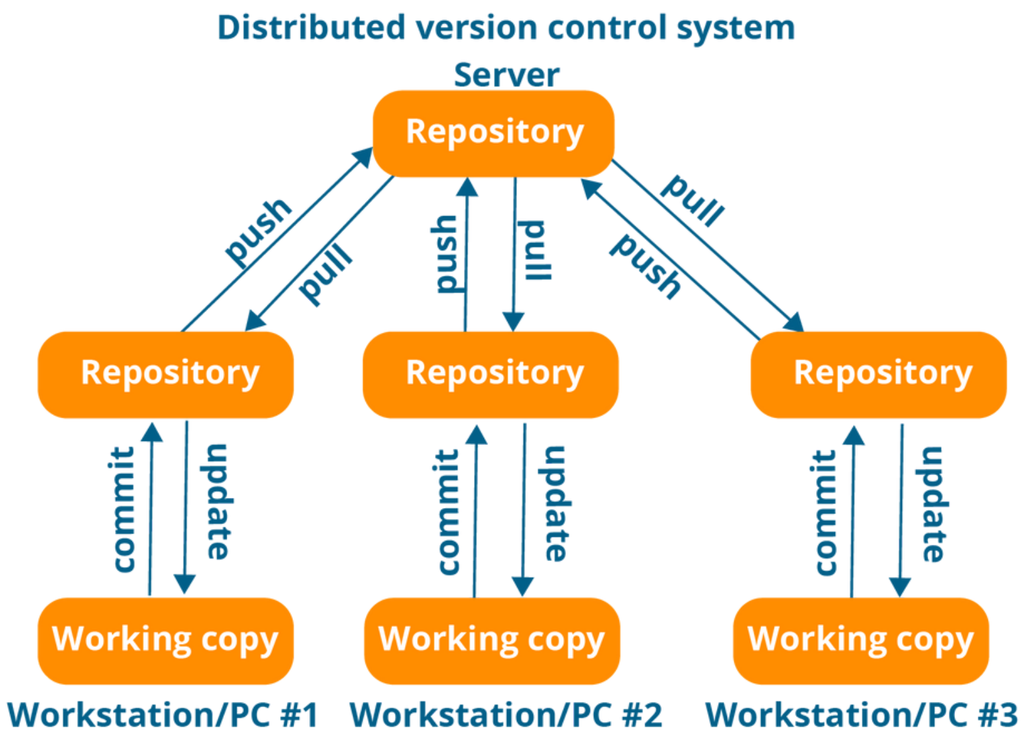
GIT	vs.	GITHUB
• Aides in Version Control		• Aides in Version Control
• Downloaded on local computer		• On the Internet
• Can be used without GitHub		• Can't be used without Git
• Great for independent projects		• Great for Collaboration
• Branching Model		• Branching Model
• Not a Social Network Feel		• Social Network Feel
• Local		• Remote

DISTRIBUTED COMPUTING



DISTRIBUTED VERSION CONTROL SYSTEM (DVCS)

- COPY OF THE COMPLETE REPOSITORY DISTRIBUTED TO EVERYONE ON THE TEAMS COMPUTER
- ALLOWS MEMBERS TO MAKE CHANGES AND WORK ON THEIR CODE LOCALLY THEN MERGE THOSE CHANGES IN REMOTELY
- MONITOR WHAT GETS MERGED IN AND CODE QUALITY



SOURCE: EUDREKA

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

GIT & GITHUB SERIES

CREATING A LOCAL GIT REPO

@LEARNINGWITHJELLY

VIDEO #2

AGENDA

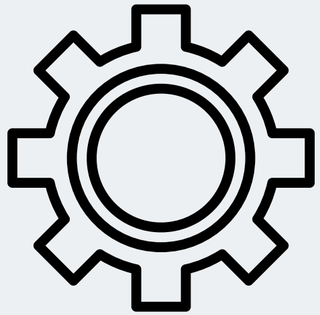
01

INSTALLING GIT/SIGN UP FOR GITHUB

02

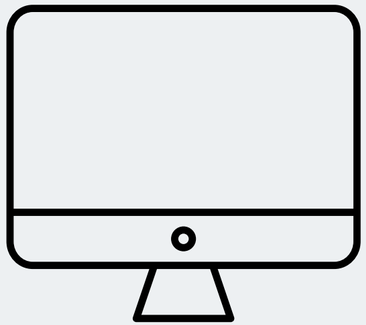
INSTALLING CODE EDITOR

03



GIT CONFIG

04



LOCAL REPO

INSTALLING GIT

- BUILT IN FOR MOST MAC COMPUTERS

- FOR PC, GO HERE AND FOLLOW THE PROMPTS:
  HTTPS://GIT-SCM.COM/DOWNLOAD/WIN

- TO CHECK WHAT VERSION OF GIT YOU HAVE, OPEN UP A TERMINAL AND TYPE:

  - GIT --VERSION (MAC)
  - $ SUDO DNF INSTALL GIT-ALL (LINUX)

SIGN-UP FOR GITHUB

- GITHUB RELIES ON GIT!

- GITHUB IS JUST A REMOTE PLACE TO PUSH UP YOUR REPOS OR PULL DOWN REPOS TO COLLABORATE (AND TO STORE YOUR PROJECTS)

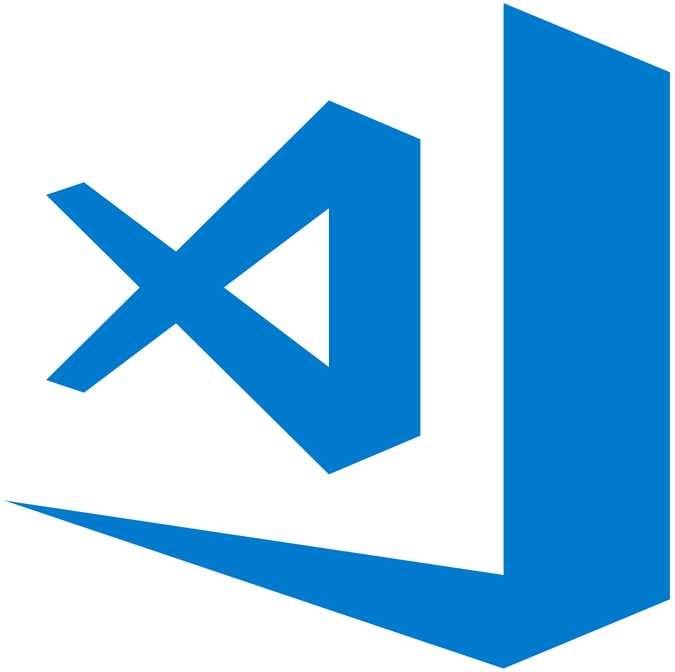
- SIGNUP AT: WWW.GITHUB.COM

- NEED TO ALSO CREATE A PERSONAL ACCESS TOKEN USED FOR YOUR "PASSWORD" WHEN ASKED FOR ONE IN THE TERMINAL

GETTING A PERSONAL ACCESS TOKEN

1. CLICK YOUR FACE (OR ICON)
2. CLICK SETTINGS
3. SCROLL DOWN TO DEVELOPER SETTINGS (CLICK IT)
4. CLICK PERSONAL ACCESS TOKENS (CLASSIC)
5. GENERATE NEW TOKEN (CLASSIC)
6. ADD NOTE
7. SET EXPIRATION
8. CHECK REPO BOX
9. GENERATE TOKEN
10. COPY AND PASTE TOKEN IN SAFE SPACE

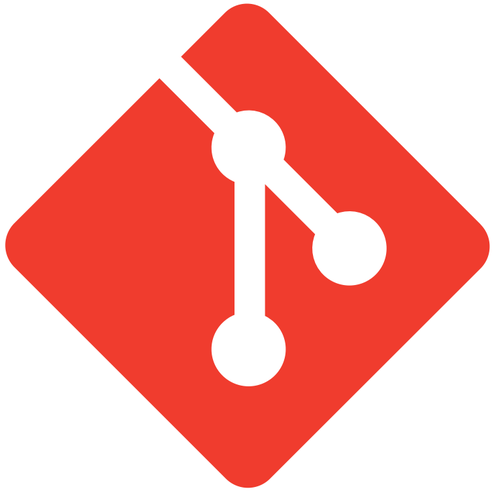
THE CODE EDITOR



INSTALLING VISUAL STUDIO CODE (VS CODE)

- VS CODE IS A CODE EDITOR THAT CAN BE USED AS A GUI FOR YOUR GITHUB REPOS
- HELPS YOU VISUALLY SEE AND TRACK CHANGES THAT YOU MAKE
- NOT REQUIRED BUT HIGHLY RECOMMENDED!
- DOWNLOAD HERE: HTTPS://CODE.VISUALSTUDIO.COM/DOWNLOAD

GIT CONFIG



CONFIGURING GIT

- EVERY GIT REPO WILL HAVE A FOLDER THAT HOUSES CONFIGURATION SETTINGS

- YOU MUST CONFIGURE YOUR NAME AND EMAIL (SAME EMAIL USED FOR GITHUB) WHEN FIRST USING GIT . <u>OPEN UP YOUR TERMINAL AND TYPE THE FOLLOWING:</u>

angelicaspratley — -zsh — 97×24

```
(base) angelicaspratley@Angelicas-MBP ~ % git config --global user.name "Angelica Spratley"
```

angelicaspratley — -zsh — 97×24

```
(base) angelicaspratley@Angelicas-MBP ~ % git config --global user.email YourGithubEmail
```

MAKE A LOCAL REPO

TERMINAL COMMANDS

SOME OF MANY COMMANDS
WHEN WE SAY TERMINAL WE MEAN

- ZSH SHELL
- BASH
- POWERSHELL
- ETC.

COMMAND	ACTION
PWD	PRINT YOUR DIRECTORY (WHERE YOU ARE)
CD	CHANGE YOUR DIRECTORY
MKDIR	MAKE NEW DIRECTORY
LS	LIST FILES IN THAT DIRECTORY
TAB	HITTING TAB COMPLETES YOUR TYPING
CTRL + L	CLEAR YOUR TERMINAL

MAKE A LOCAL REPO

1. Change your directory to your 'Documents' folder

2. Create a new folder in your documents called 'first_repo'

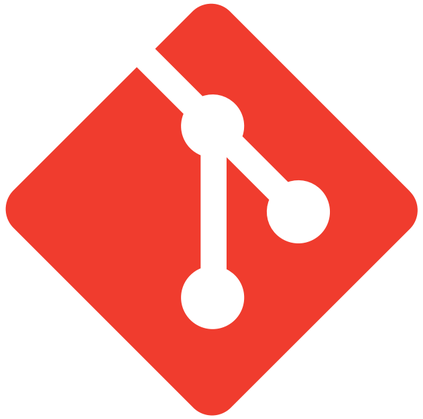
3. Change your directory so that you are inside the 'first_repo' folder

4. Change that folder to a git repo by typing **`git init`**

first_repo — -zsh — 97×24

```
(base) angelicaspratley@Angelicas-MBP ~ % cd Documents          1
(base) angelicaspratley@Angelicas-MBP Documents % mkdir first_repo   2
(base) angelicaspratley@Angelicas-MBP Documents % cd first_repo   3
(base) angelicaspratley@Angelicas-MBP first_repo % git init   4
Initialized empty Git repository in /Users/angelicaspratley/Documents/first_repo/.git/
(base) angelicaspratley@Angelicas-MBP first_repo %
```

“SO WHAT IS GIT INIT?



GIT INIT "INITIALIZES A GIT REPO" AKA TURNS THE FOLDER INTO A GIT REPO THAT WE CAN USE FOR VERSION CONTROL!

CREATE A FILE INSIDE LOCAL REPO

Make sure you are in the 'first_repo' directory

1. Create a README.md file using the touch command
2. Type git status to check to see if the file was created (this is a command you should use A LOT to check the status of your repository)

first_repo — -zsh — 97×24

```
(base) angelicaspratley@Angelicas-MBP first_repo % touch README.md   1
(base) angelicaspratley@Angelicas-MBP first_repo % git status   2
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
(base) angelicaspratley@Angelicas-MBP first_repo %
```

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

GIT & GITHUB SERIES

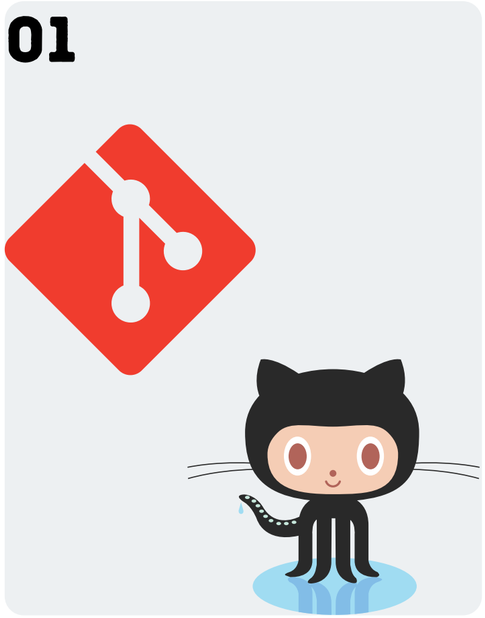
CONNECTING A LOCAL REPO TO GITHUB

@LEARNINGWITHJELLY

VIDEO #3

AGENDA

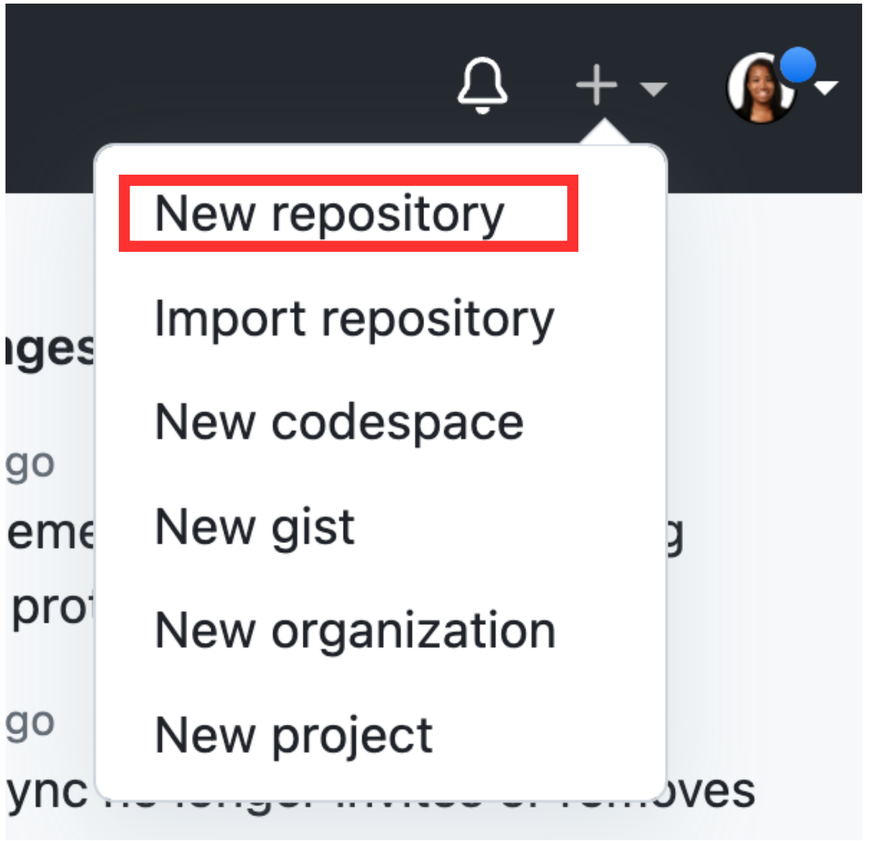
01



CONNECTING LOCAL REPO TO GITHUB

CREATE A NEW REPO ON GITHUB

- LOG INTO YOUR GITHUB ACCOUNT
- HIT THE '+' SIGN NEXT TO YOUR FACE AND CLICK ON "NEW REPOSITORY"
- GIVE THE REPO A NAME -- SOMETIMES I MAKE IT THE SAME NAME AS MY LOCAL VERSION
- MAKE IT PUBLIC (IF YOU WANT ANYONE TO SEE IT)
- CLICK 'CREATE REPOSITORY'



CONNECT OUR 'FIRST_REPO' TO GITHUB

- FOLLOW THE SECOND BOX OF INSTRUCTIONS IN YOUR TERMINAL (EXCEPT FOR LAST LINE)!

- BE SURE YOU ARE IN YOUR LOCAL REPO (AKA THE DIRECTORY IS THE REPO YOU CREATED)

- IF THERE IS AN ERROR, READ WHAT GIT IS TRYING TO SUGGEST FOR YOU TO DO

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

```
git remote add origin https://github.com/aspratle/first_repo.git
git branch -M main
```

GIT COMMANDS

Common Git Commands

- $git config
- $git init
- $git clone <path>
- $git add <file_name>
- $git commit
- $git status
- $git remote
- $git checkout <branch_name>
- $git branch
- $git push
- $git pull
- $git merge <branch_name>
- $git diff
- $git reset
- $git revert
- $git tag
- $git log

SOURCE: DEV COMMUNITY

STEPS FOR STARTING LOCAL THEN REMOTE

1. CREATE A FOLDER ON YOUR COMPUTER FOR THE PROJECT
2. RUN A `GIT INIT` TO CHANGE IT TO A GIT REPO
3. USE THE `TOUCH` COMMAND TO MAKE A FILE
4. CREATE A REPO ON GITHUB
5. FOLLOW THE SECOND BOX OF CODE TO CONNECT THE REMOTE REPO TO YOUR LOCAL ONE
6. ANYTIME YOU CHANGE A FILE:
   A. GIT ADD FILENAME
   B. GIT COMMIT -M "MESSAGE"
   C. GIT PUSH

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

PUSHING CHANGES TO GITHUB REPO

GIT & GITHUB SERIES

VIDEO #4

@LEARNINGWITHJELLY

AGENDA

01

VS CODE INTERFACE

02

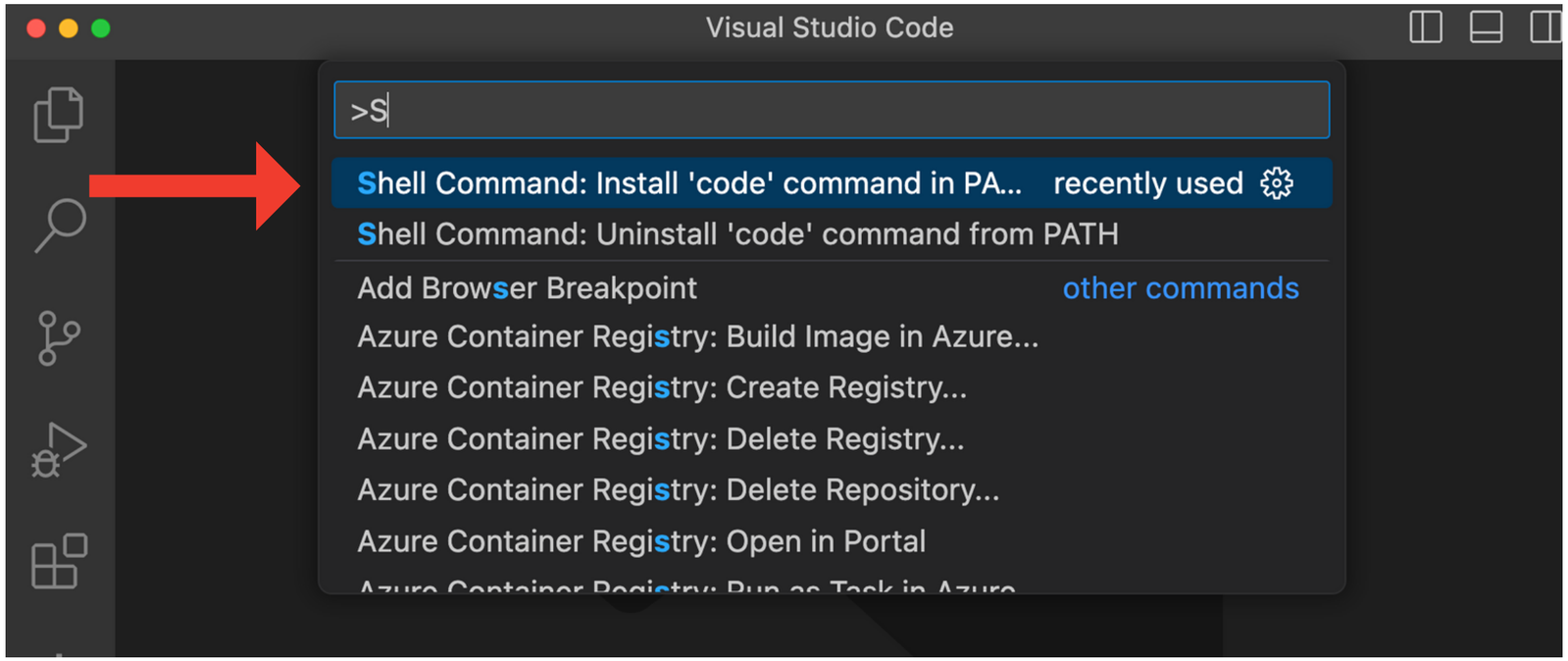
CHANGE A FILE

03

PUSH CHANGES

CONNECT VSCODE TO TERMINAL

- OPEN UP VS CODE
- PRESS 'COMMAND + UP ARROW + P'
- IN THE SEARCH BAR TYPE
  - > SHELL COMMAND
- CLICK THE "INSTALL 'CODE' COMMAND IN PATH
- <u>THIS ALLOWS US TO USE 'KEYWORD' CODE IN OUR TERMINAL TO OPEN UP VS CODE</u>



IF IT DOES NOT WORK: TRY 'UNINSTALLING' THE SHELL COMMAND THEN INSTALLING IT (CLICK THE SECOND CHOICE ABOVE THEN THE FIRST)

CHANGE A FILE

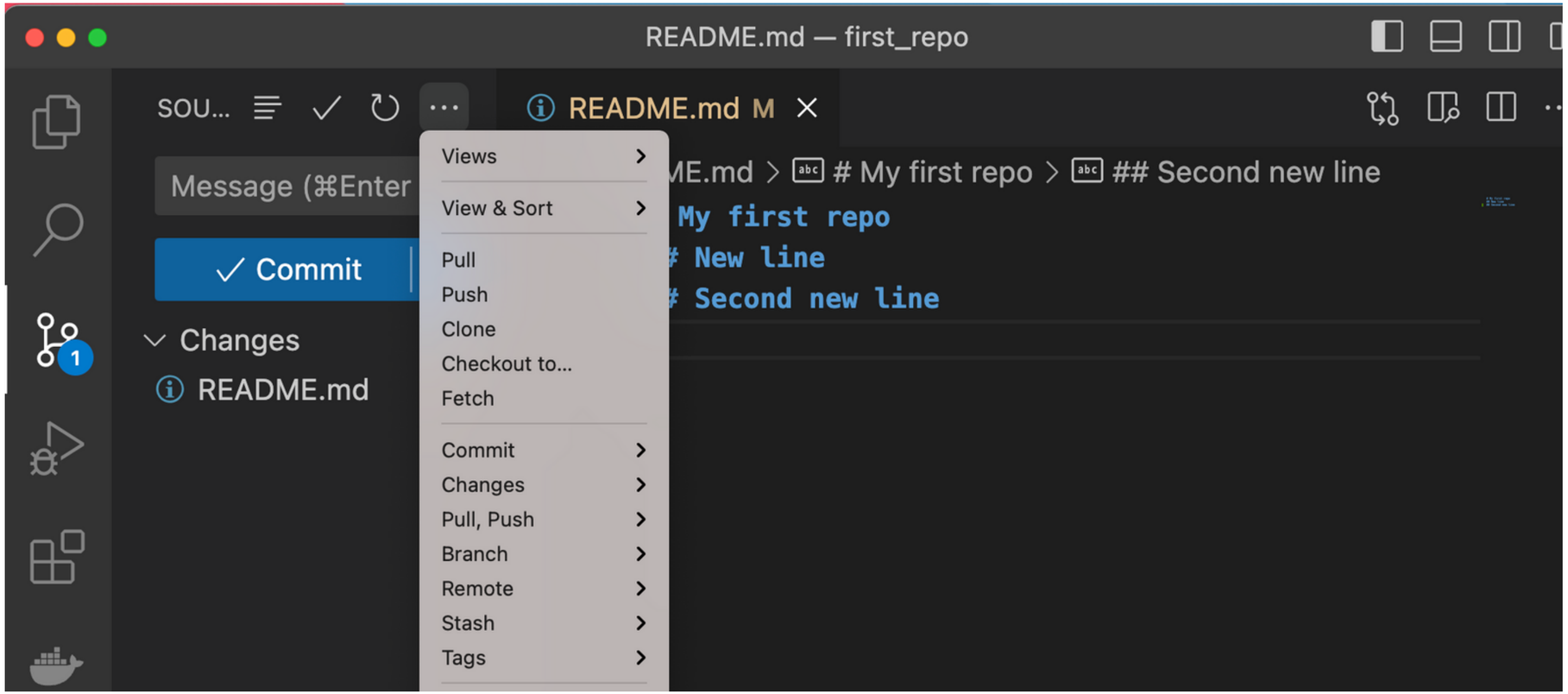
RED COLOR IS WHAT YOU TYPE IN THE TERMINAL!

1. NAVIGATE TO YOUR LOCAL REPO 'FIRST_REPO'
2. OPEN UP README IN VSCODE (CODE README.MD)
3. CHANGE THE FILE
4. SAVE CHANGES (COMMAND+S OR CTRL + S)
5. ADD THE CHANGED FILE TO STAGING (GIT ADD)
6. WRITE A COMMIT MESSAGE ABOUT THE CHANGE (GIT COMMIT -M "MESSAGE")
7. GIT PUSH TO PUSH CHANGES UP

FOLLOW VIDEO FOR DEMONSTRATION!

VS CODE AS GIT GUI

ONCE YOU SAVE FILES IN VSCODE YOU ARE ABLE TO STAGE, COMMIT, AND PUSH THOSE CHANGES RIGHT WITHIN VSCODE!



THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

CREATE A REMOTE GITHUB REPO FIRST

GIT & GITHUB SERIES

VIDEO #5

@LEARNINGWITHJELLY

AGENDA

01

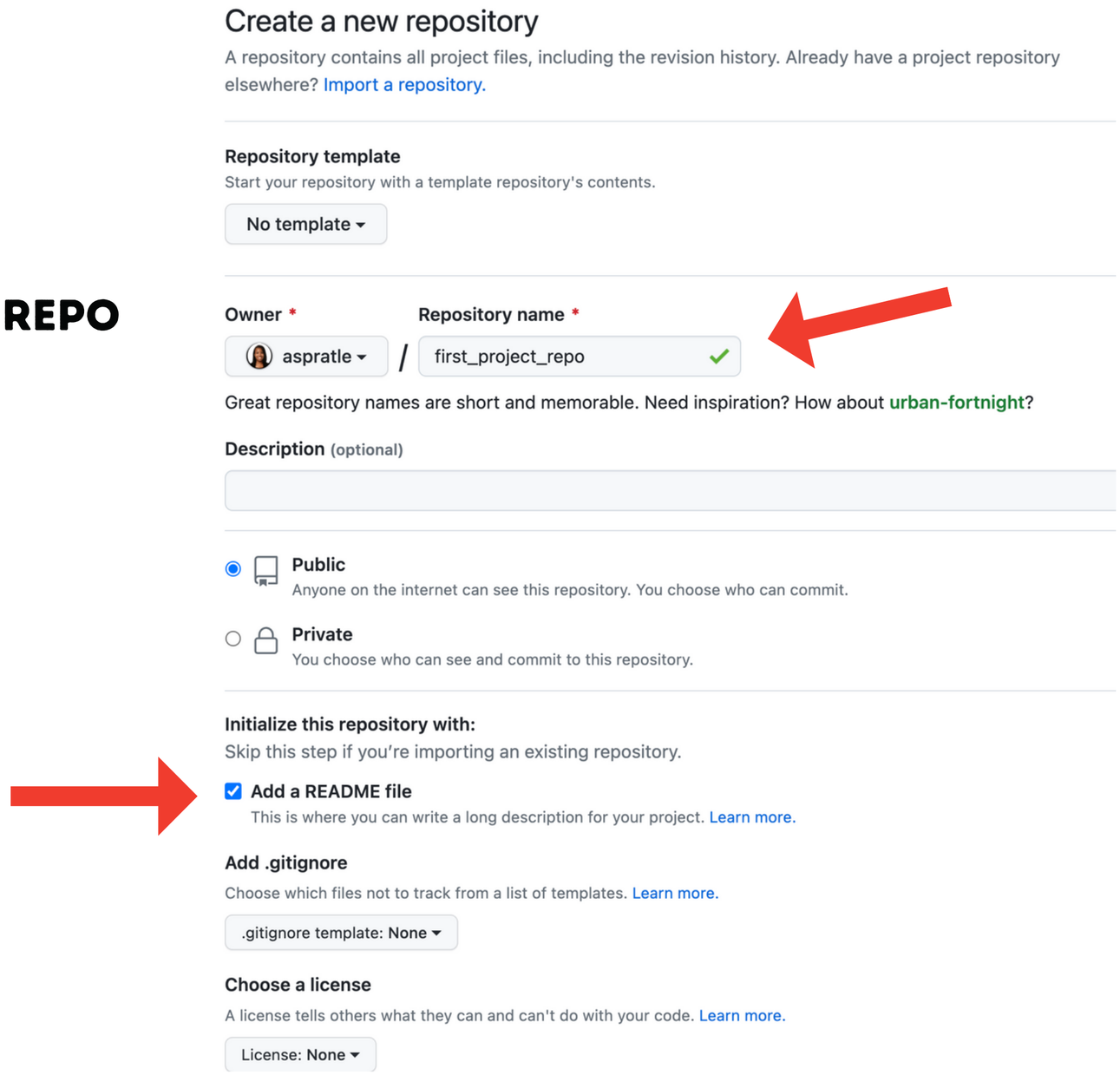
CREATE A REPO IN GITHUB

02

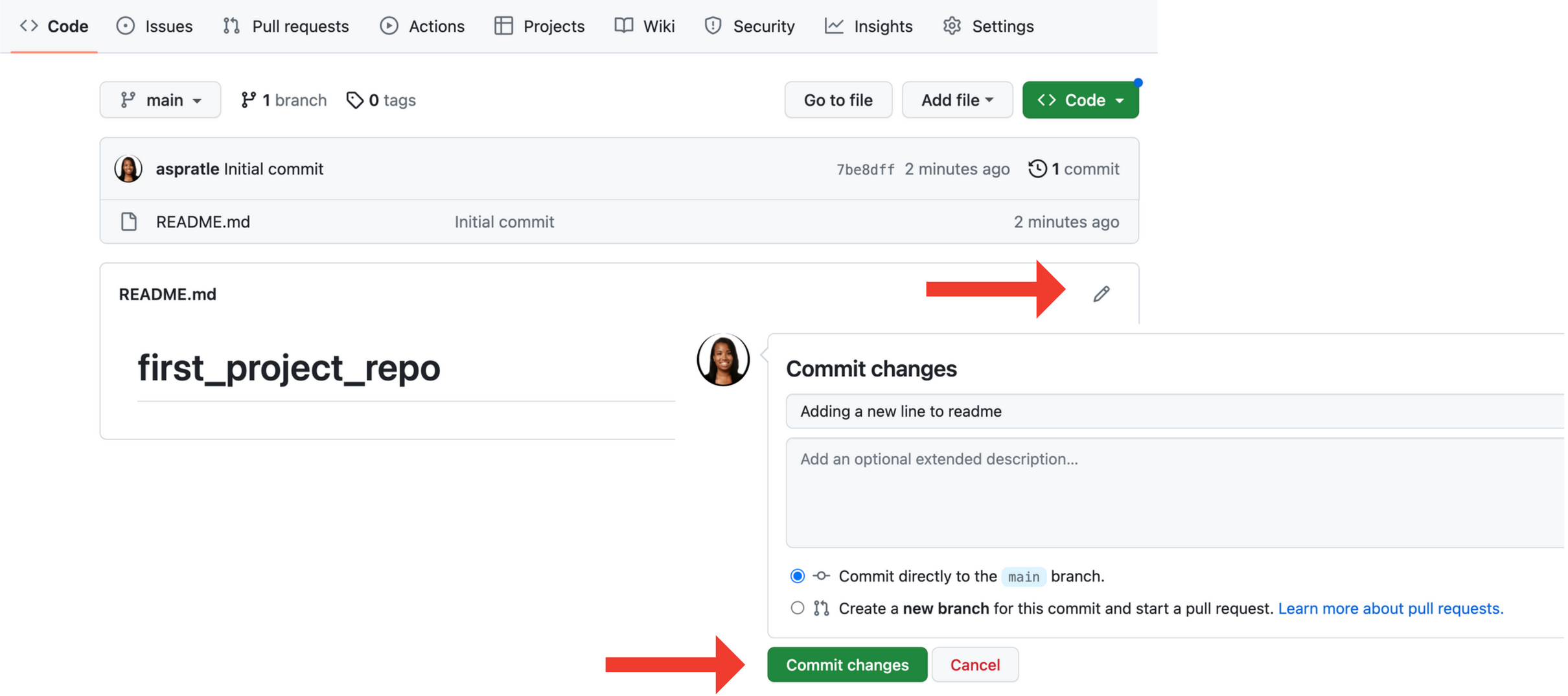
MAKE CHANGES RIGHT IN GITHUB

CREATE A GITHUB REPO

- HIT THE '+' SIGN NEXT TO YOUR FACE
- ADD AN OWNER (YOUR ACCOUNT) AND REPO NAME
- CLICK PUBLIC
- CHECK 'ADD A README FILE'
- "CREATE REPOSITORY"

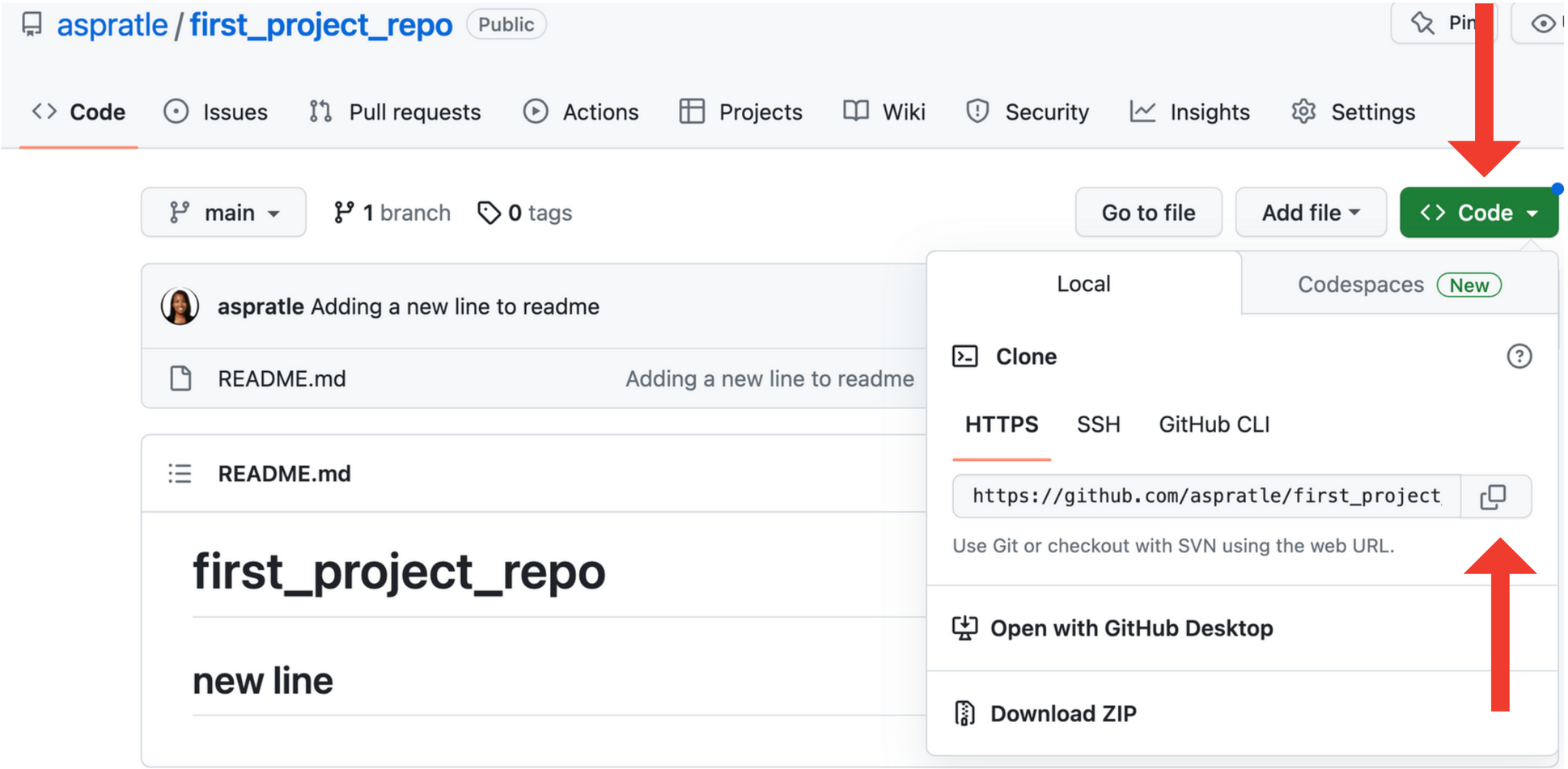


EDIT THE README RIGHT WITHIN GITHUB



CLONE THE REPO DOWN TO YOUR LOCAL COMPUTER

- **COPY THE URL OF THE REPO**



CLONE THE REPO DOWN TO YOUR LOCAL COMPUTER

first_project_repo — -zsh — 117×24

```
(base) angelicaspratley@Angelicas-MBP ~ % cd Documents
(base) angelicaspratley@Angelicas-MBP Documents % mkdir project_repos
(base) angelicaspratley@Angelicas-MBP Documents % cd project_repos
(base) angelicaspratley@Angelicas-MBP project_repos % git clone https://github.com/aspratle/first_project_repo.git
Cloning into 'first_project_repo'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
(base) angelicaspratley@Angelicas-MBP project_repos % cd first_project_repo
(base) angelicaspratley@Angelicas-MBP first_project_repo % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(base) angelicaspratley@Angelicas-MBP first_project_repo %
```

MAKE NEW FILES OR MOVE FILES INSIDE THAT REPO

```
(base) angelicaspratley@Angelicas-MBP first_project_repo % touch code.ipynb
(base) angelicaspratley@Angelicas-MBP first_project_repo % code code.ipynb
(base) angelicaspratley@Angelicas-MBP first_project_repo % git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        code.ipynb

nothing added to commit but untracked files present (use "git add" to track)
(base) angelicaspratley@Angelicas-MBP first_project_repo % git add code.ipynb
(base) angelicaspratley@Angelicas-MBP first_project_repo % git commit -m "Adding code file for addition"
[main f88cfa4] Adding code file for addition
 1 file changed, 49 insertions(+)
 create mode 100644 code.ipynb
(base) angelicaspratley@Angelicas-MBP first_project_repo % git push
```

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

FORKING AND CLONING GITHUB REPOS

GIT & GITHUB SERIES

VIDEO #6

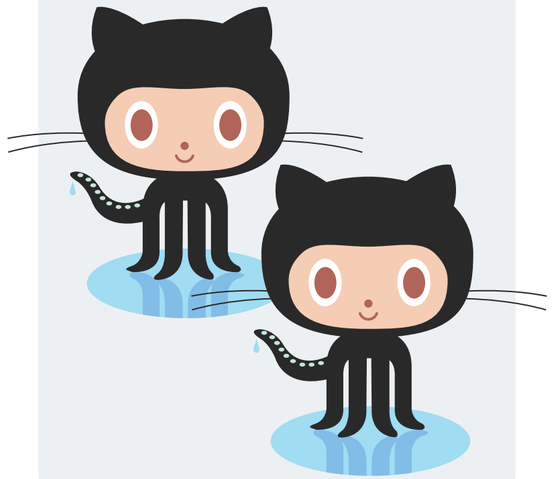
@LEARNINGWITHJELLY

AGENDA

01

FORKING

02

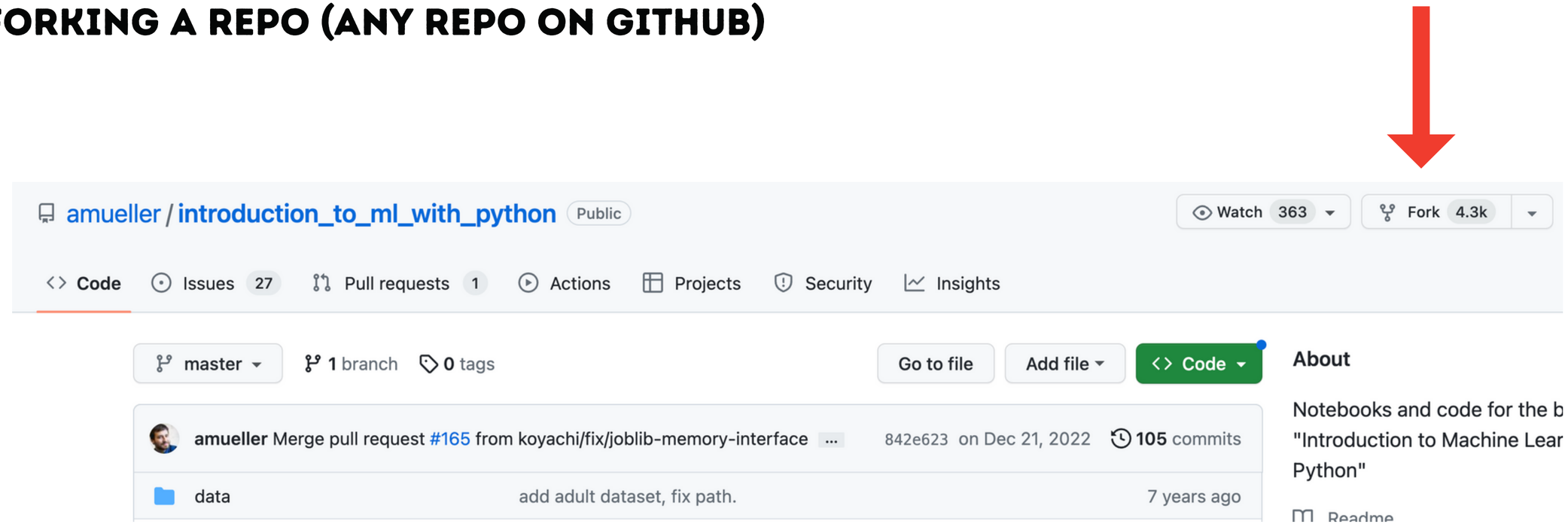


CLONING

FORKING A REPO

- FORKING A REPO IS GOOD WHEN YOU WANT TO ADD A COPY OF SOMEONE ELSE'S REPO TO YOUR ACCOUNT (FOR YOUR OWN PROJECT OR USE)

- TRY FORKING A REPO (ANY REPO ON GITHUB)



FORKING A REPO

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Owner * **Repository name** *

aspratle / introduction_to_ml_with_pythor

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Notebooks and code for the book "Introduction to Machine Learning with Python"

☑ **Copy the** `master` **branch only**

Contribute back to amueller/introduction_to_ml_with_python by adding your own branch. Learn more.

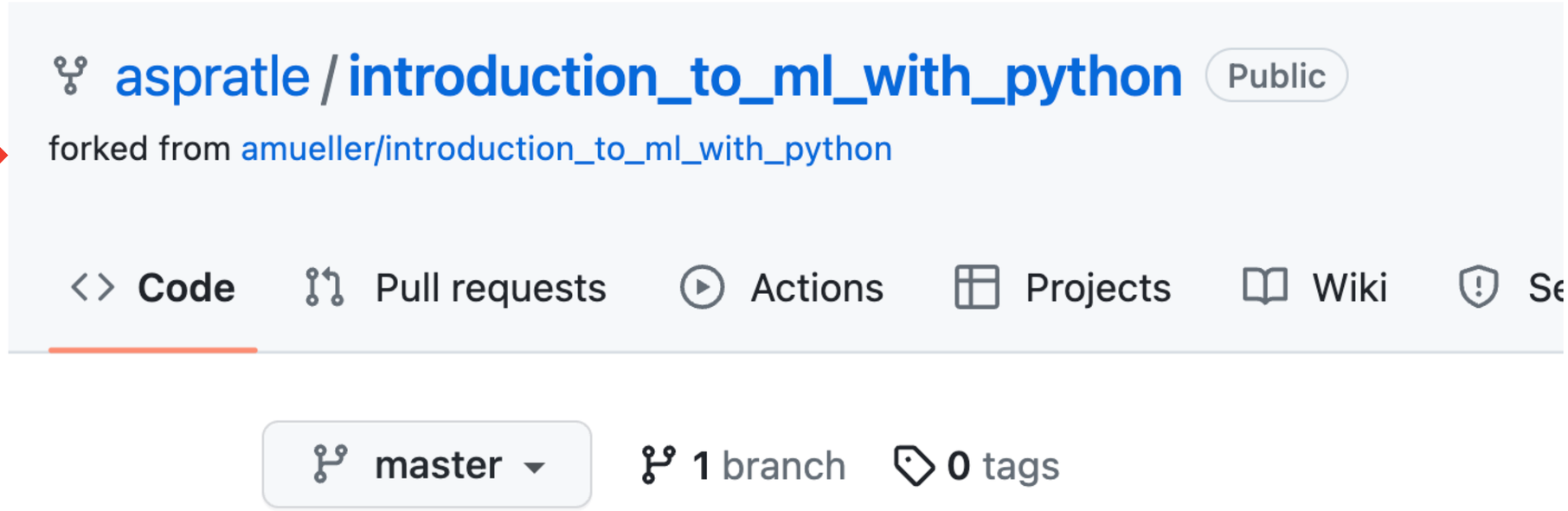
ⓘ You are creating a fork in your personal account.

Create fork

FORKING A REPO

YOUR GITHUB ACCOUNT USERNAME

ORIGINAL GITHUB REPO CREATOR



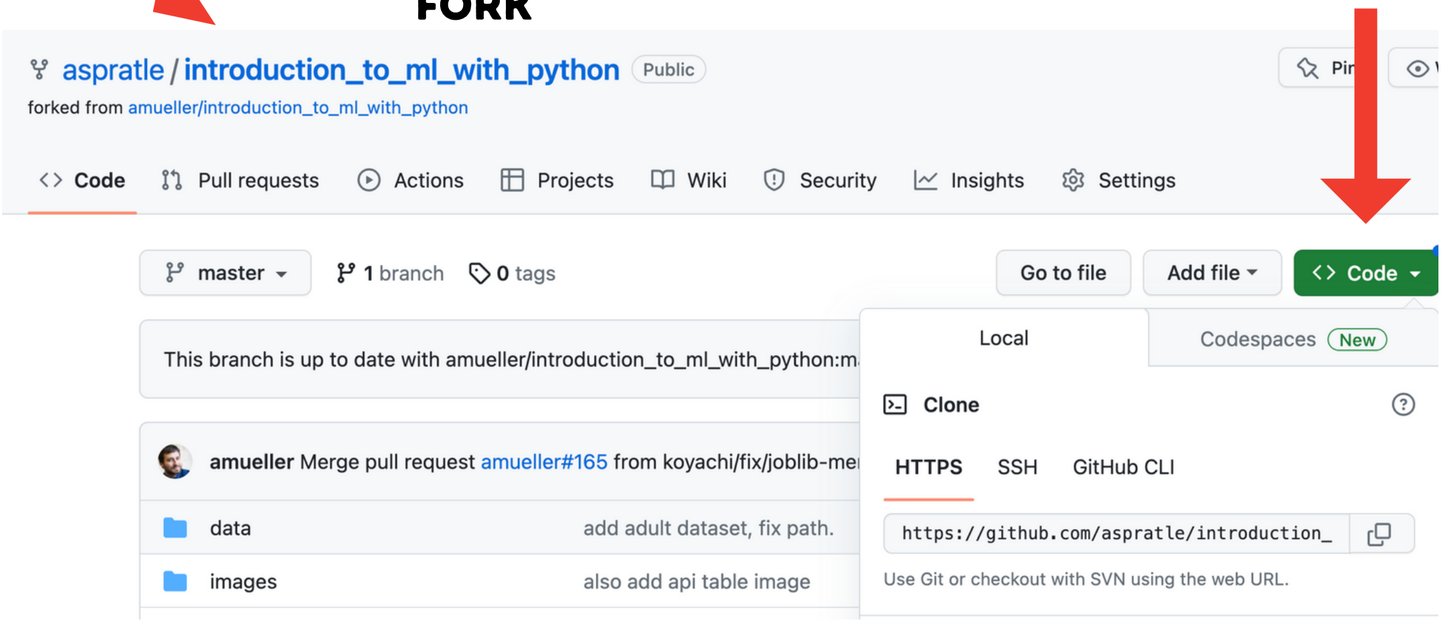
CLONING THE FORK

- USE `GIT CLONE FORKURL` TO CLONE DOWN YOUR FORK LOCALLY

- YOU CAN NOW MAKE NEW FILES, CHANGE FILES, ETC. TO YOUR FORK AND PUSH THEM UP AS ALWAYS IT IS
  - GIT ADD
  - GIT COMMIT -M "MESSAGE"
  - GIT PUSH

MAKE SURE IT IS YOUR FORK

COPY THE URL



CLONING THE FORK

```
(base) angelicaspratley@Angelicas-MBP Documents %
(base) angelicaspratley@Angelicas-MBP Documents % cd project_repos
(base) angelicaspratley@Angelicas-MBP project_repos % git clone https://github.com/aspratle/introduction_to_ml_with_python.git
Cloning into 'introduction_to_ml_with_python'...
remote: Enumerating objects: 453, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 453 (delta 4), reused 10 (delta 3), pack-reused 440
Receiving objects: 100% (453/453), 182.02 MiB | 19.86 MiB/s, done.
Resolving deltas: 100% (193/193), done.
(base) angelicaspratley@Angelicas-MBP project_repos % cd introduction_to_ml_with_python
(base) angelicaspratley@Angelicas-MBP introduction_to_ml_with_python % git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
(base) angelicaspratley@Angelicas-MBP introduction_to_ml_with_python % code
(base) angelicaspratley@Angelicas-MBP introduction_to_ml_with_python % git add README.md
(base) angelicaspratley@Angelicas-MBP introduction_to_ml_with_python % git commit -m "Making a change"
[master 2c648af] Making a change
 1 file changed, 2 insertions(+)
(base) angelicaspratley@Angelicas-MBP introduction_to_ml_with_python % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
```

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

GIT & GITHUB SERIES

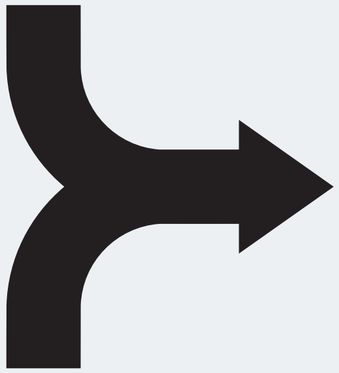
USING GIT FETCH AND GIT MERGE

@LEARNINGWITHJELLY

VIDEO #7

AGENDA

01



GIT FETCH
AND GIT
MERGE

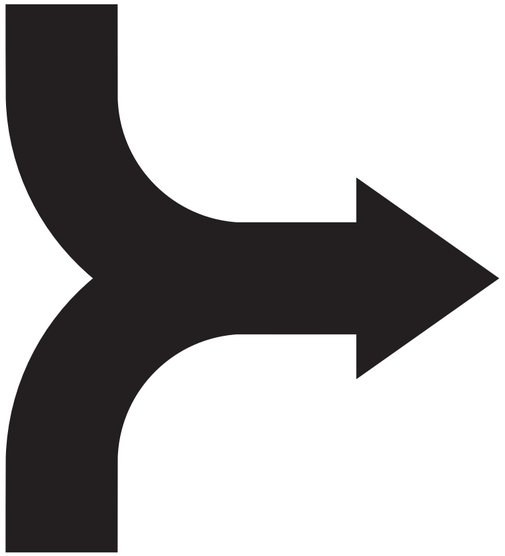
GIT FETCH/MERGE

- SOMETIMES YOU WANT TO MERGE THE ORIGINAL OWNER'S CONTENT INTO YOUR FORK

- THIS IS SO YOU HAVE THE MOST RECENT CHANGES ON YOUR REPO THAT THE OWNER HAS MADE TO THE ORIGINAL REPO

- FIRST: SET THE ORIGINAL OWNERS REPO URL AS THE UPSTREAM (GIT REMOTE ADD UPSTREAM URL)
- GIT FETCH UPSTREAM
- GIT MERGE UPSTREAM/MAIN -M "YOUR MESSAGE"

NOTE MAY BE GIT MERGE UPSTREAM/MASTER -M "YOUR MESSAGE FOR OLDER REPOS!



THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

GIT & GITHUB SERIES

ADDING A .GITIGNORE FILE

@LEARNINGWITHJELLY

VIDEO #8

AGENDA

01

CREATING A
.GITIGNORE FILE

.GITIGNORE

- A .GITIGNORE FILE IS USED WHEN YOU <u>DON'T WANT GIT TO TRACK CERTAIN FILES</u>

- IT IS COMMON TO ADD CODING NOTEBOOK 'CHECKPOINTS' SOMETIMES SEEN AS ".IPYNB_CHECKPOINTS" TO THE .GITIGNORE FILE IF CREATING JUPYTER NOTEBOOKS

- IT IS IMPORTANT TO ADD THIS FILE IN THE BEGINNING (IF ADDED AFTER YOU PUSH A CODE NOTEBOOK, IT WON'T TRACK FUTURE CHECKPOINTS BUT WILL STILL KEEP THE OLD ONES)

.GITIGNORE

- NAVIGATE INSIDE YOUR REPO
- IN TERMINAL TYPE THE FOLLOWING:
  - TOUCH .GITIGNORE (DON'T FORGET THE '.')
  - CODE (OPENS IN VSCODE)
    - EDIT THE FILE BY COPYING AND PASTING:
    - .IPYNB_CHECKPOINTS
    - */.IPYNB_CHECKPOINTS/*
  - GIT ADD .
  - GIT COMMIT -M "COMMIT MESSAGE"
  - GIT PUSH

.gitignore (Untracked)

.gitignore

```
1    1  .ipynb_checkpoints
2    2  */.ipynb_checkpoints/*
```

```
nothing to commit, working tree clean
(base) angelicaspratley@Angelicas-MBP Python_DataAnalyst % touch .gitignore
(base) angelicaspratley@Angelicas-MBP Python_DataAnalyst % code
```

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

REVERTING BACK TO A DIFFERENT VERSION

@LEARNINGWITHJELLY

GIT & GITHUB SERIES

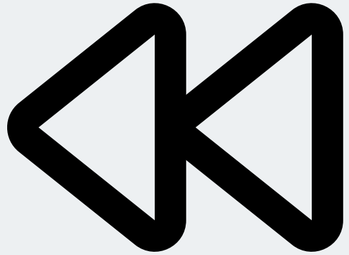
VIDEO #9

AGENDA

01

GIT LOG

02



GOING BACK TO AN OLD VERSION

GIT LOG

- **GIT LOG** WILL PRINT OUT A LOG OF ALL COMMITS INCLUDING THE COMMIT "ID"

- YOU CAN TYPE **GIT LOG --ONELINE** TO PRINT OFF A MORE READABLE SUMMARY OF EACH COMMIT

- <u>TO EXIT OUT OF THE LOG PRESS 'Q'</u>

3 commits

first_project_repo — -zsh — 113×24

```
(base) angelicaspratley@Angelicas-MBP project_repos % cd first_project_repo
(base) angelicaspratley@Angelicas-MBP first_project_repo % git log --oneline
f88cfa4 (HEAD -> main, origin/main, origin/HEAD) Adding code file for addition
c64dc04 Adding a new line to readme
7be8dff Initial commit
(base) angelicaspratley@Angelicas-MBP first_project_repo %
```

GIT REVERT

- GIT REVERT COMMITID

- IT'LL TAKE YOU INTO VIM MODE (PRESS :WQ TO QUIT OUT OF IT ONCE YOU MAKE CHANGES OR :Q IF YOU MAKE NO CHANGES)

- TYPE GIT LOG --ONELINE AND YOU WILL SEE THAT YOU REVERT BACK

first_project_repo — -zsh — 113×24

```
(base) angelicaspratley@Angelicas-MBP project_repos % cd first_project_repo
(base) angelicaspratley@Angelicas-MBP first_project_repo % git log --oneline
f88cfa4 (HEAD -> main, origin/main, origin/HEAD) Adding code file for addition
c64dc04 Adding a new line to readme
7be8dff Initial commit
(base) angelicaspratley@Angelicas-MBP first_project_repo % git revert c64dc0
[main 4a175ad] Revert "Adding a new line to readme"
 1 file changed, 1 insertion(+), 2 deletions(-)
(base) angelicaspratley@Angelicas-MBP first_project_repo % git log --oneline
4a175ad (HEAD -> main) Revert "Adding a new line to readme"
f88cfa4 (origin/main, origin/HEAD) Adding code file for addition
c64dc04 Adding a new line to readme
7be8dff Initial commit
(base) angelicaspratley@Angelicas-MBP first_project_repo %
```

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

CREATING BRANCHES IN GIT/GITHUB

GIT & GITHUB SERIES

@LEARNINGWITHJELLY

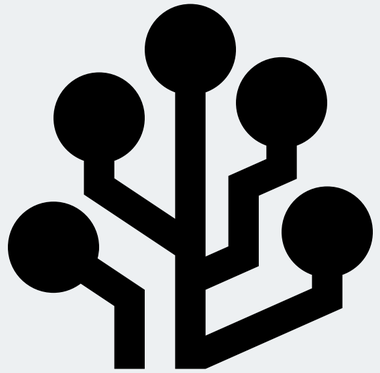
VIDEO #10

AGENDA

01

DEFINE BRANCHES

02



CREATE A BRANCH

BRANCHES

- **BRANCHES** ARE USED WHEN COLLABORATION IS INVOLVED (NOT NECESSARY FOR INDIVIDUAL PROJECTS)

- MOST OF THE TIME, BRANCHES ARE FOR A SPECIFIC SPRINT OR FEATURE

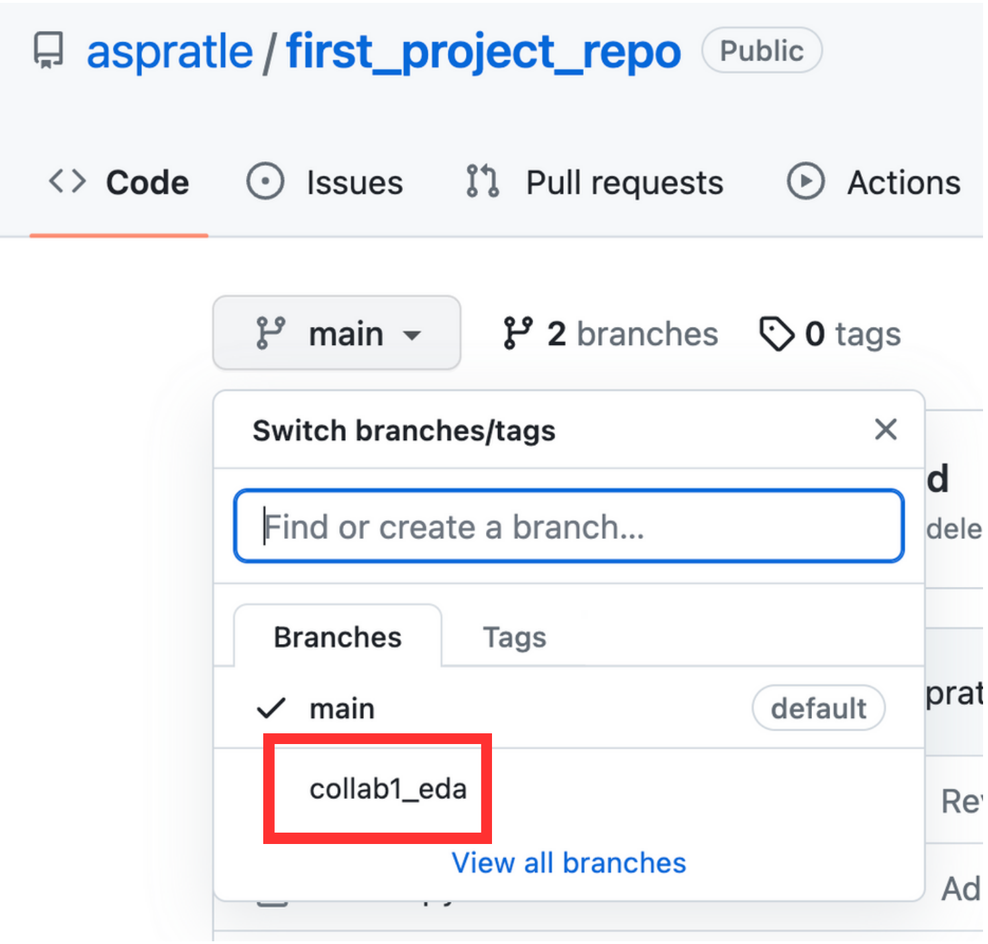
- BRANCHES CAN BE MERGED INTO THE MAIN BRANCH VIA PULL REQUESTS

- ALWAYS MAKE SURE YOU ARE PUSHING TO THE RIGHT BRANCH USING **GIT BRANCH**!

```
(base) angelicaspratley@Angelicas-MBP first_project_repo % git branch
* main
(base) angelicaspratley@Angelicas-MBP first_project_repo %
```

CREATE A BRANCH

- USE **GIT BRANCH NAMEOFBRANCH** TO CREATE A NEW BRANCH
- USE **GIT CHECKOUT NAMEOFBRANCH** TO SWITCH TO THAT BRANCH
- ONCE ON THAT BRANCH YOU CAN CREATE YOUR OWN FILES AND PUSH THEM TO THAT BRANCH (<u>BRANCH WILL NOT SHOW UNTIL YOU ADD A FILE TO IT</u>)




```
(base) angelicaspratley@Angelicas-MBP first_project_repo % git branch collab1_eda
(base) angelicaspratley@Angelicas-MBP first_project_repo % git checkout collab1_eda
Switched to branch 'collab1_eda'
(base) angelicaspratley@Angelicas-MBP first_project_repo % git status
On branch collab1_eda
```

THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE

RESOLVING MERGE CONFLICTS IN GIT/GITHUB

@LEARNINGWITHJELLY

GIT & GITHUB SERIES

VIDEO #11

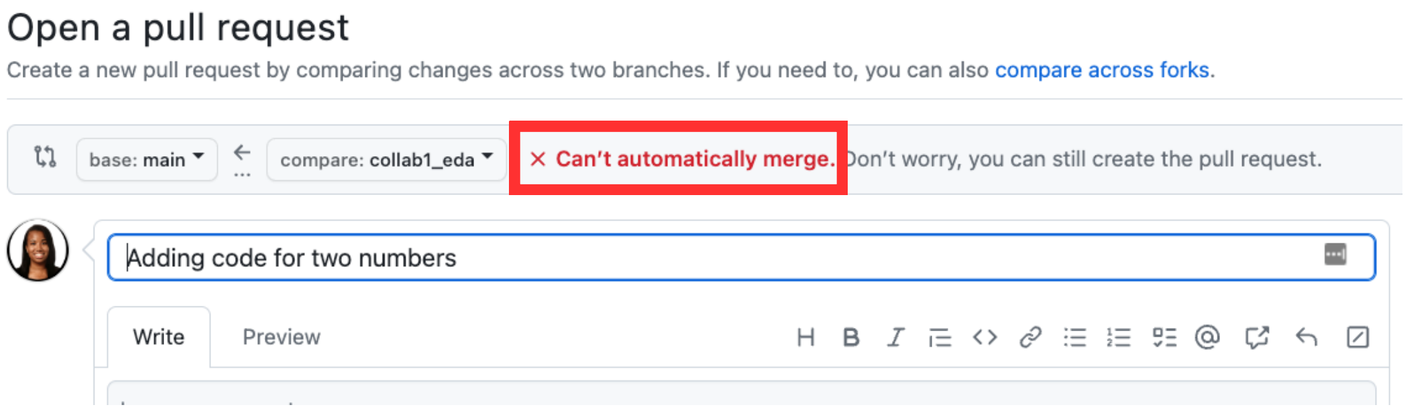
AGENDA

01

MERGE
CONFLICTS

MERGE CONFLICTS

- DUE TO THE COLLABORATIVE NATURE OF GIT, THERE COULD BE MULTIPLE PEOPLE CHANGING THE SAME THINGS AT THE SAME TIME AND CAUSE CONFLICT
- GIT TRIES TO DO AN AUTO-MERGE BY DEFAULT
- IF IT CAN'T AUTO-MERGE, IT WILL RAISE A CONFLICT
- YOU THEN HAVE TO GO IN AND DECIDE WHAT CHANGES TO ACCEPT
- MERGE CONFLICTS CAN BE VERY COMPLEX



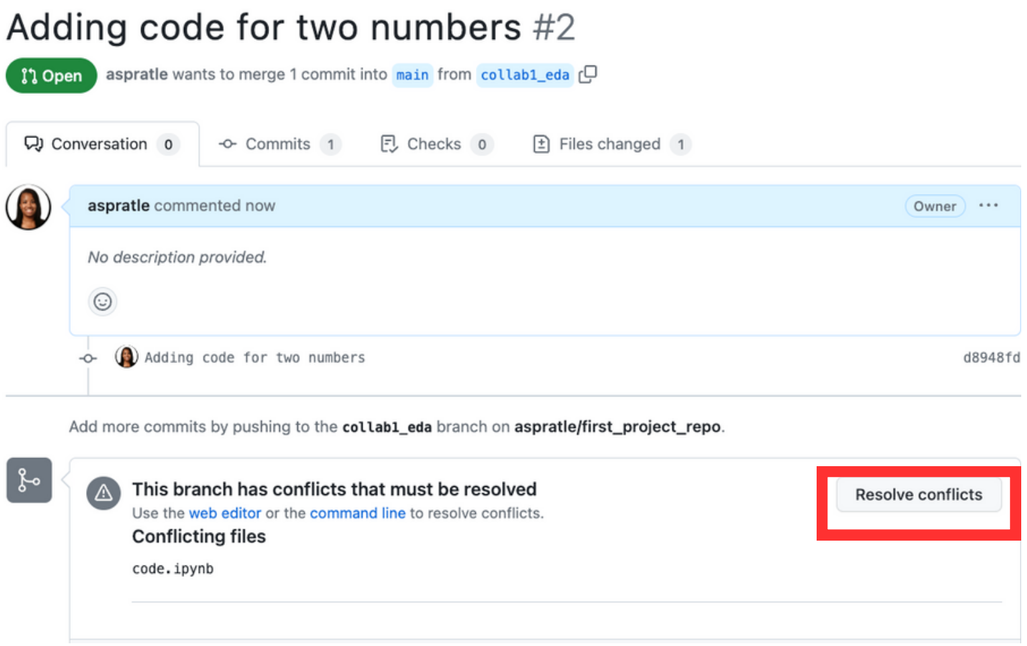
MERGE CONFLICTS

- ONCE YOU CLICK 'CREATE PULL REQUEST' YOU WILL BE TAKEN TO A SCREEN WHERE YOU CAN RESOLVE CONFLICTS

- FOR CODE CONFLICTS IT IS BEST TO LOOK AT EACH BRANCH AND UPDATE ONE OF THE BRANCHES WITH THE RIGHT CODE

- MERGE CONFLICTS CAN EXIST WHEN YOUR LOCAL REPO IS DIFFERENT THAN THE REMOTE OR

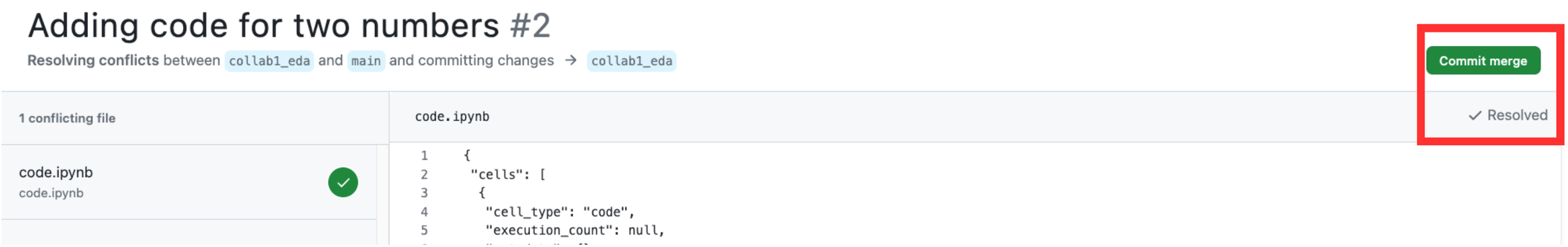
- WHEN SOMEONE CHANGED THE SAME FILE ON A BRANCH, THAT IS DIFFERENT THAN THE MAIN AND WANT TO MERGE IN THEIR FILE WITH THE MAIN BRANCH



MERGE CONFLICTS

- ONCE YOU RESOLVE THE CONFLICT YOU CAN MARK RESOLVED AND COMMIT TO MERGE

- RESOLVING CONFLICTS MAY BE NOT MERGING THE REQUEST, UPDATING ONE/BOTH BRANCHES, ETC..



THANK YOU!

@LEARNINGWITHJELLY

SUBSCRIBE