Introduction to Git

Version Control Basics

Sam Miyamoto, MPH for Data Umbrella
December 2024

Agenda

3	Hello and Introduction!
4	What is Git?
6	Vocab and Terminology
7	Git Branching
14	Demos
23	Collaborating
30	Continued Learning
32	Q&A, Wrap

Hello and Introduction!

I'm a software engineer based in the Los Angeles, California area. I have experience in clean transportation, data storage, and intersections with public health. Thank you for attending my talk!

Website: https://www.smiyamoto.dev/

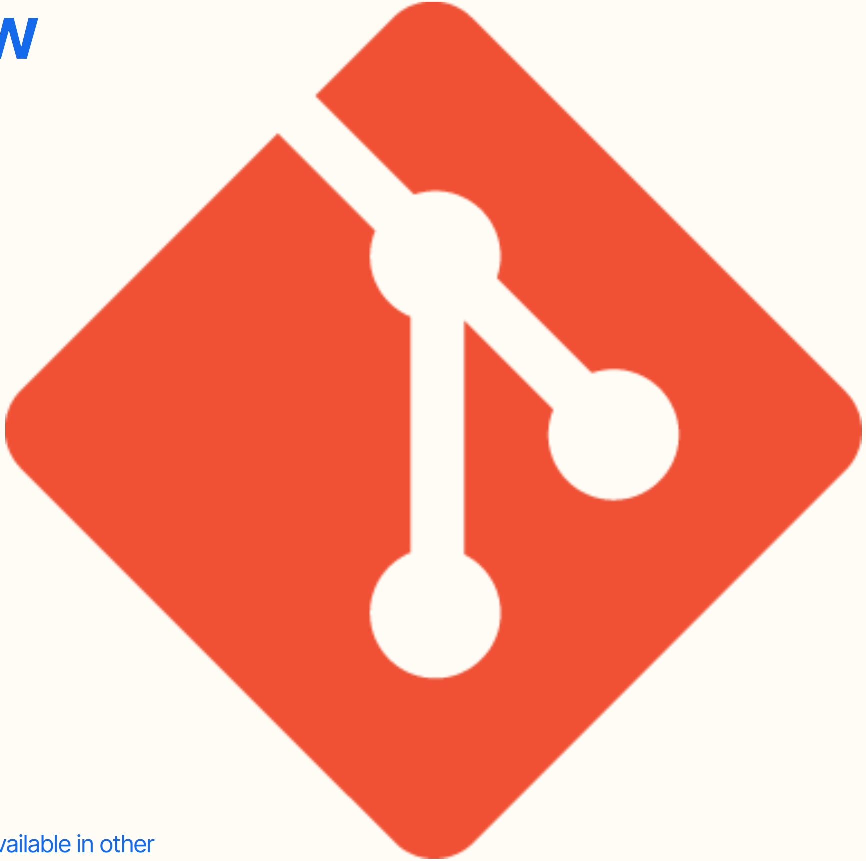
GitHub: https://github.com/samvmdev

LinkedIn: https://www.linkedin.com/in/e-samantha-miyamoto/

What is Git? 30,000-ft view (and some details)

Git is a popular open-source version control system (VCS) / source code management (SCM) tool.

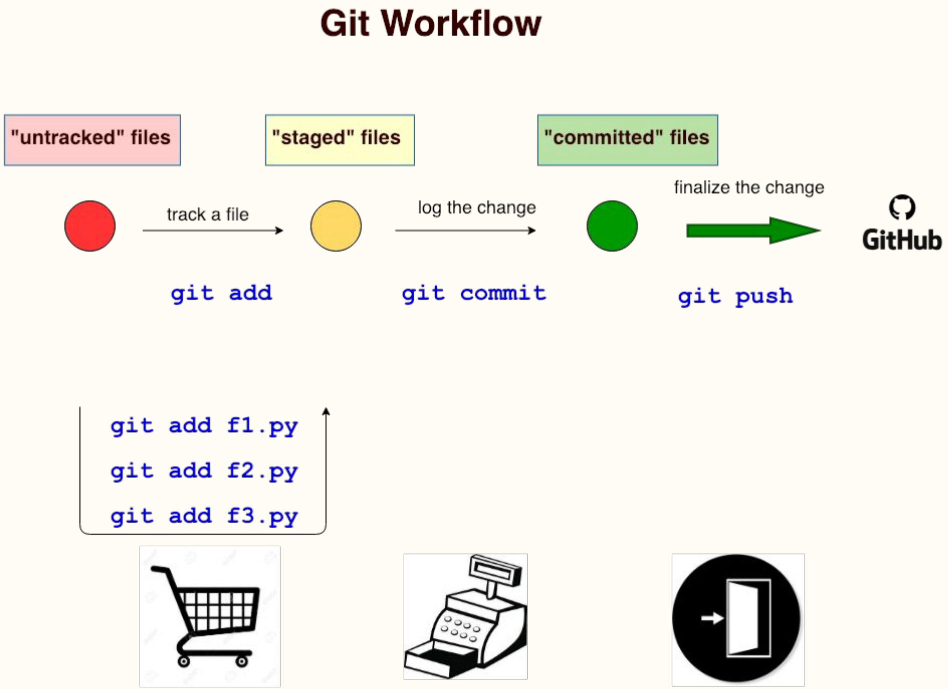
At a high level, it is used to manage code, keep track of code changes, and collaborate on software / data science development.



Link and logo: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git (available in other languages)

The Three Stages of a File in Git

1) Untracked or Modified
2) Staged
3) Committed



```
git add
git commit
git push
```

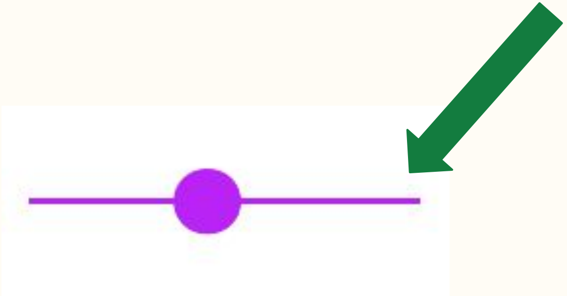
```
git add f1.py
git add f2.py
git add f3.py
```

Image courtesy of Reshama Shaikh, Data Umbrella

Vocab / Terminology

Term	Brief Description
Branch	Line of development
Commit	Snapshot
Push	Share / send code
Pull	Get / fetch and merge code
CLI	Command line interface
GUI	Graphical user interface

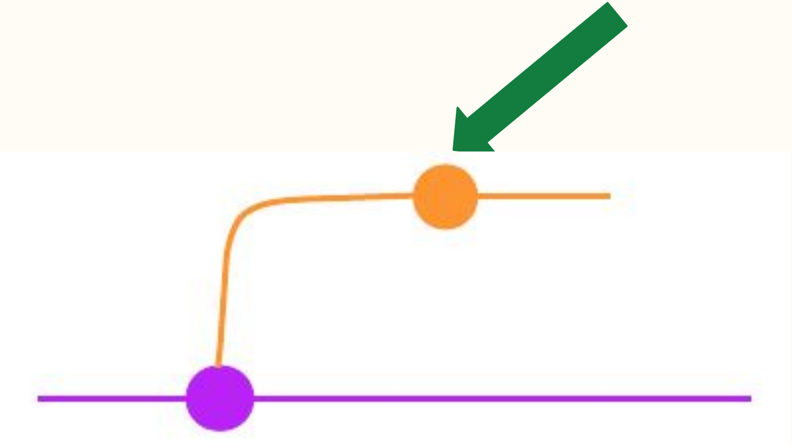
Git Branching



main branch

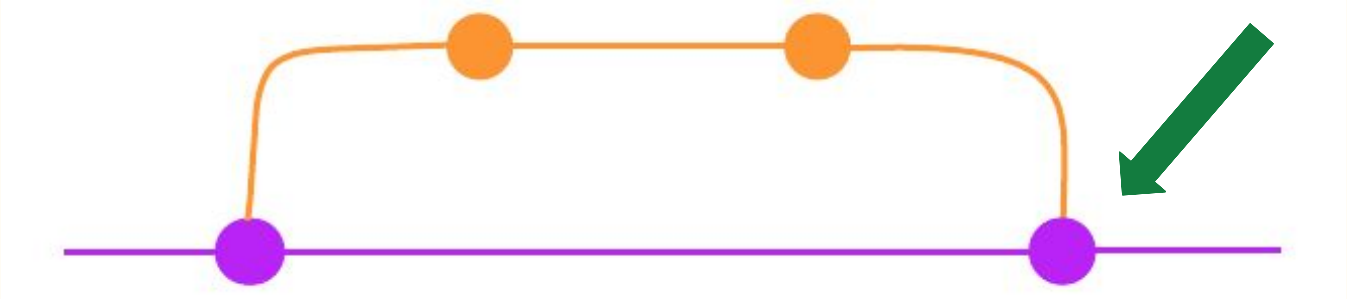
Git Branching

feature/orange branching off main with one commit

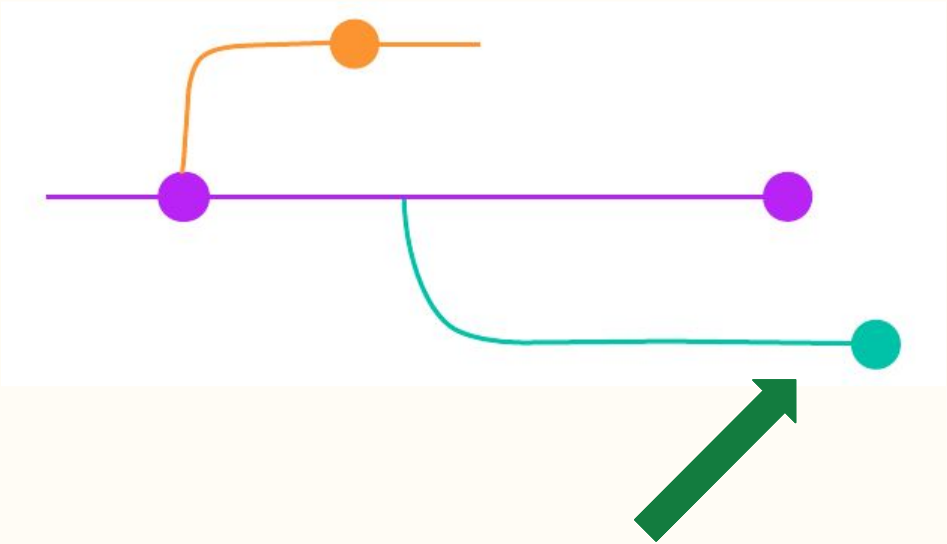


Git Branching

feature/orange branch with two commits merged back into main

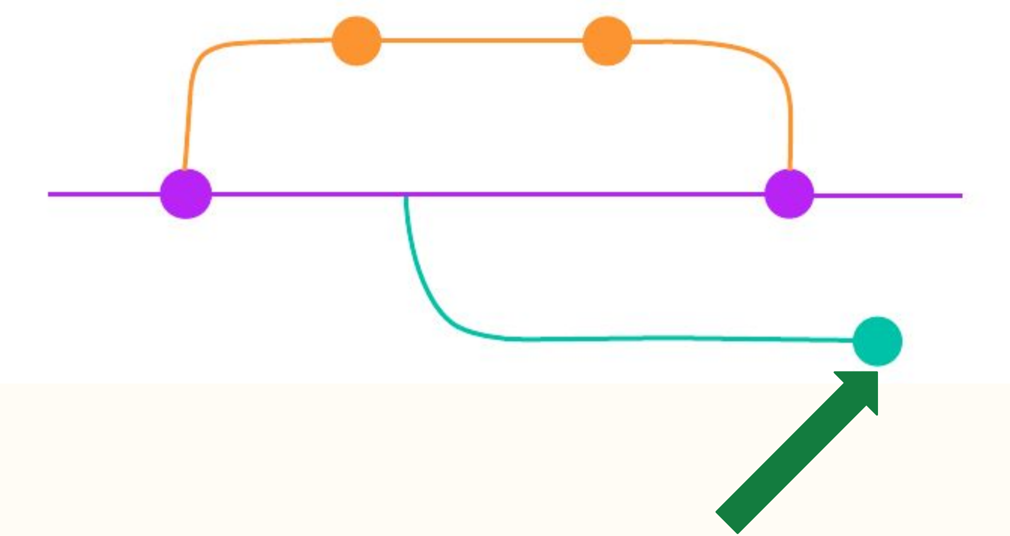


Git Branching - Parallel Development



feature/teal branch and feature/orange are being developed simultaneously

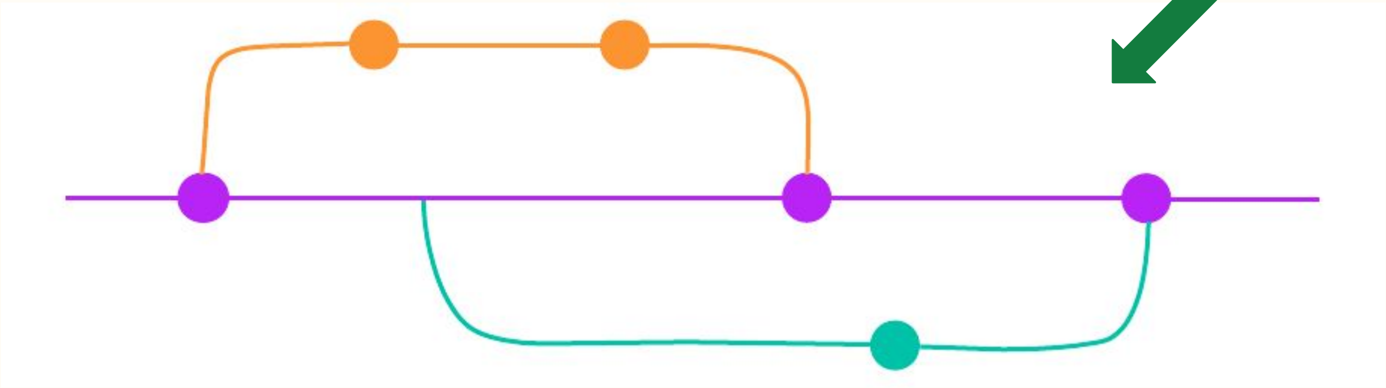
Git Branching - Parallel Development



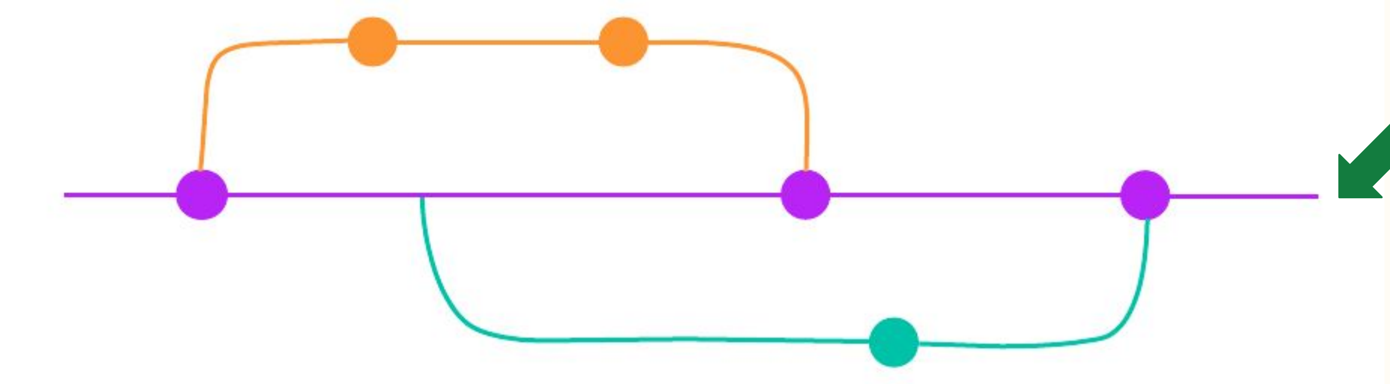
feature/teal branch is still under development after feature/orange was merged in

Git Branching - Parallel Development

feature/teal branch with one commit is merged back into main



Git Branching - Parallel Development



main now has both features merged in

Demo Time

Pre-Requisites

Download and install Git and Visual Studio Code (VSCode)

Git: https://git-scm.com/downloads

VSCode: https://code.visualstudio.com/download

Important - Configure Your Git Identity

```
git config --global user.email “<email>”
git config --global user.name “<name>”
```

Note: If this is not configured upon first Git install, Git will use information from your local machine.

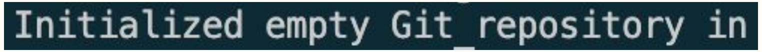
You can override these settings for specific projects without the --*global* option.

Create a new local Git repository - CLI

`cd ~` (Navigates to home directory)

`mkdir example_cli` (creates new local repository)

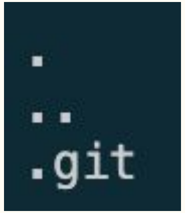
`git init` (initiates Git repository)



```
Initialized empty Git repository in
```

<path>

`ls -la` (to see presence of Git files)

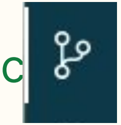


```
.
..
.git
```

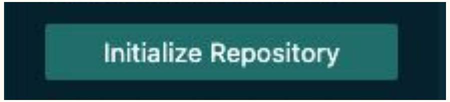
Create a new local Git repository - GUI

Step 1: Create *example_gui* directory

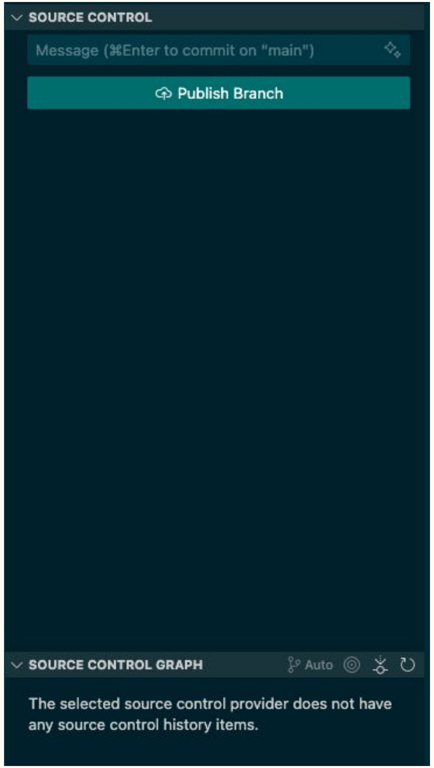
Step 2: Navigate to the Side Bar on left side → Clic Source Control icon



Step 3: Click



Step 4: Confirm

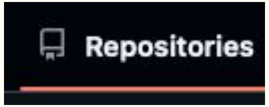


The selected source control provider does not have any source control history items.

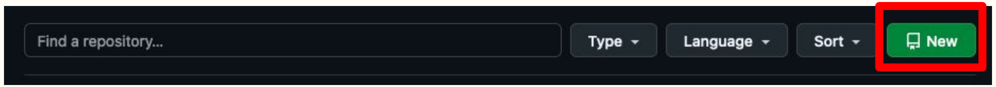
Create a GitHub repository

Step 1: Navigate to GitHub (requires that you have an account)

Step 2: Navigate to Repositories tab



Step 3: Click New



Step 4: Populate information

Step 5: Click Create Repository

Make a commit - CLI

`touch example_file.txt`

Make edits to the file, save

`git add example_file.txt` *(to stage file)*

`git commit -m “<Commit message>”` *(commits files)*

Note: Run *git status* at various steps to see what state your file is in (untracked, staged). It can provide color-coded output.

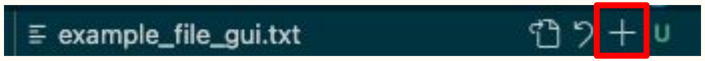
Make a commit - GUI

Step 1: Right click, create *example_file_gui.txt*

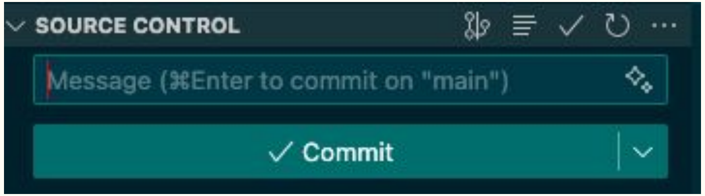
Make edits to the file, save w/ Ctrl + S

Step 2: Navigate to Source Control

Step 3: Press the Plus (+) sign to stage changes



Step 4: Write your commit message and press Commit



Note: The U stands for "Untracked".

Create a branch - CLI

```
git checkout main
git checkout -b <new branch name>
```

Note: Make sure you're branching off the desired branch!

To see what branch you're currently on, run *git branch*.

Create a branch - GUI

Option 1 - Source Control Pane

On Source Control, Click the Ellipsis

Hover over “Branch”

Click “Create Branch” and follow prompts

Note: Make sure you’re branching off the desired branch.

Option 2 - Command Palette

Command Shift P (or Ctrl + Shift + P) to open the Command Palette

Type “Git: Create Branch” in the Input bar and follow the prompts

Collaborating in the Cloud

Connecting to a Remote Repository (GitHub) - Push Code

Step 1: Create the repository on GitHub

Step 2: Get the URL (SSH or HTTPS):

- Step 1a: Both require setup of either an SSH key or a personal access token (PAT)

Step 3: **`git remote add origin <url>`** (SSH example: git@github.com:username/reponame.git)

Step 4: **`git push origin <branch name>`**

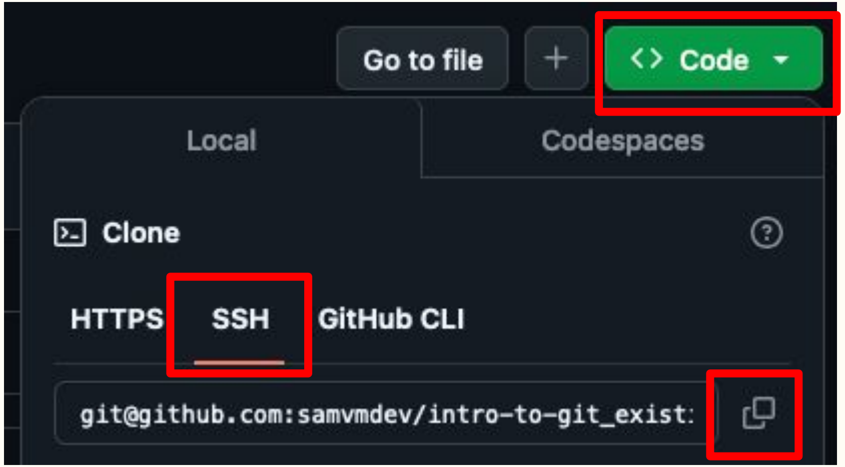
Note: To set up an SSH key, follow this documentation from GitHub: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Connecting to a Remote Repository - Pull Code

Step 1: Get the repo URL (SSH or HTTPS):

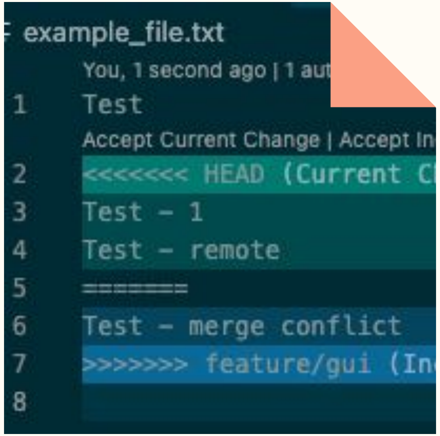
- Step 1a: Both require setup of either an SSH key or a personal access token (PAT)

Step 2: ***`git clone <url>`***
(SSH example:
git@github.com:username/reponame.git)



```
git@github.com:samvmdev/intro-to-git_exist:
```

Troubleshooting Some Common Git Issues



```
example_file.txt
You, 1 second ago | 1 aut
Test
Accept Current Change | Accept In
<<<<<<< HEAD (Current C
Test - 1
Test - remote
=======
Test - merge conflict
>>>>>>> feature/gui (In
```

Merge Conflicts*

Reconciling Divergent Branches

Detached HEAD

Sunflower photo by Tatiana Reusche on Unsplash

Merge Conflicts

How this may arise:

Changing the same part of the same file differently in the two branches you're merging.

One solution:

Re-open your file(s), resolve the conflicts, and commit the results.

To prevent:

They can happen during the course of development. Staying vigilant, committing early and often, and having shorter pull requests can help reduce these instances.

Git Rebase

This is a command (similar to git merge) that is used to move all of the commits from a feature branch onto main.

```
git checkout feature

git rebase main
```

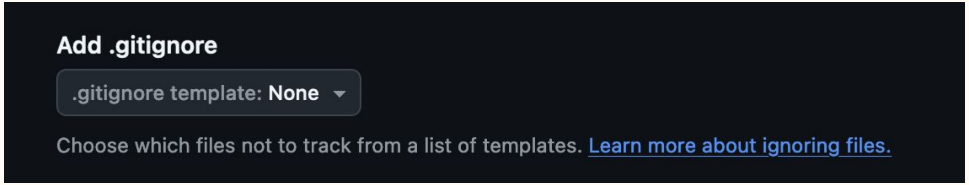
While this command can be helpful for cleaning up commit history, it can be disruptive to any team-based work. Be careful when using this command.

Last, but Not Least - .gitignore

Add a .gitignore file so you keep files you don’t want committed out of a shared repository.

Files to add for exclusion in the .gitignore include logs and files produced by the build system

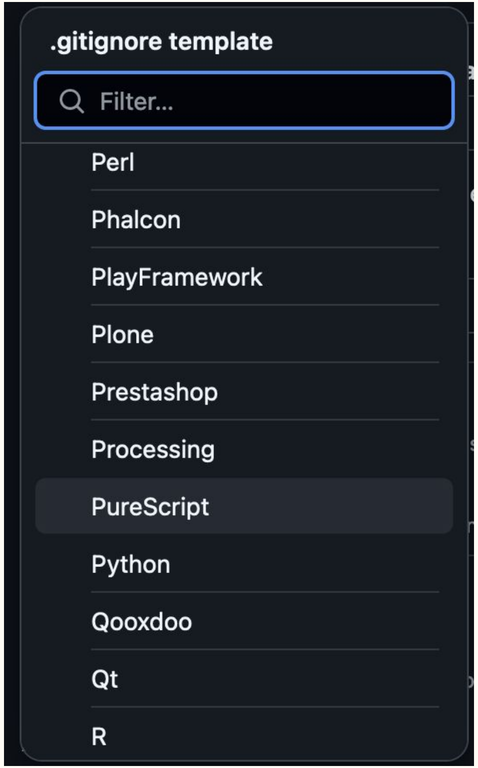
GitHub has many .gitignore templates for various languages!



Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.



Continued Learning

git stash

git reflog

git cherry-pick

Aliases

Using Git tags

Q&A

Additional Resources

Git documentation: https://git-scm.com/docs

Free Pro Git book (Scott Chacon and Ben Straub): https://git-scm.com/book/en/v2

Atlassian tutorials: https://www.atlassian.com/git/tutorials

GitLens extension (includes AI Explain):
https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens

Additional history:
https://stackoverflow.blog/2023/01/09/beyond-git-the-other-version-control-systems-developers-use/

Example workflows:
https://github.com/reshamas/git-intro-workshop/tree/master/workflows

Thank you!