

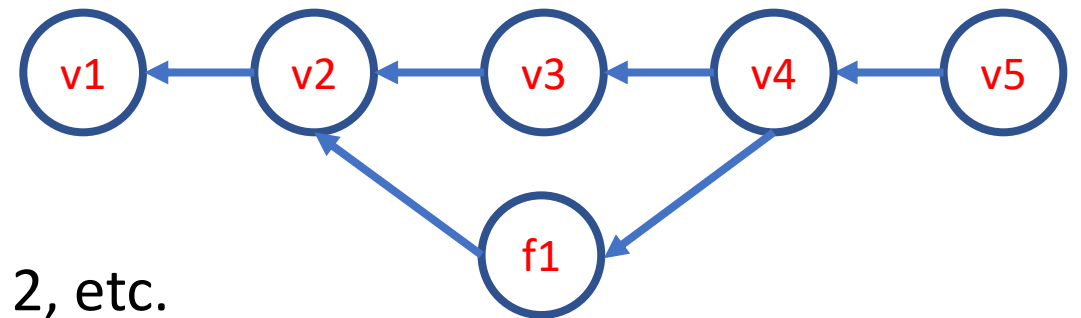
Intro to Git & GitHub

Jay / [TDMDAL](#)

Website for this workshop: <https://tdmdal.github.io/git-workshop-2023-rccl>

What's Git git

- A version control system
 - manage the evolution of a set of files (repository / repo)
 - mainly for source code (or text files)
 - **NOT** for large datasets, but see [git lfs](#) and [github lfs](#)
 - NOT really for binary files (.xlsx, .docx, .pdf, etc.): hard to track content changes, but OK to use as file backups

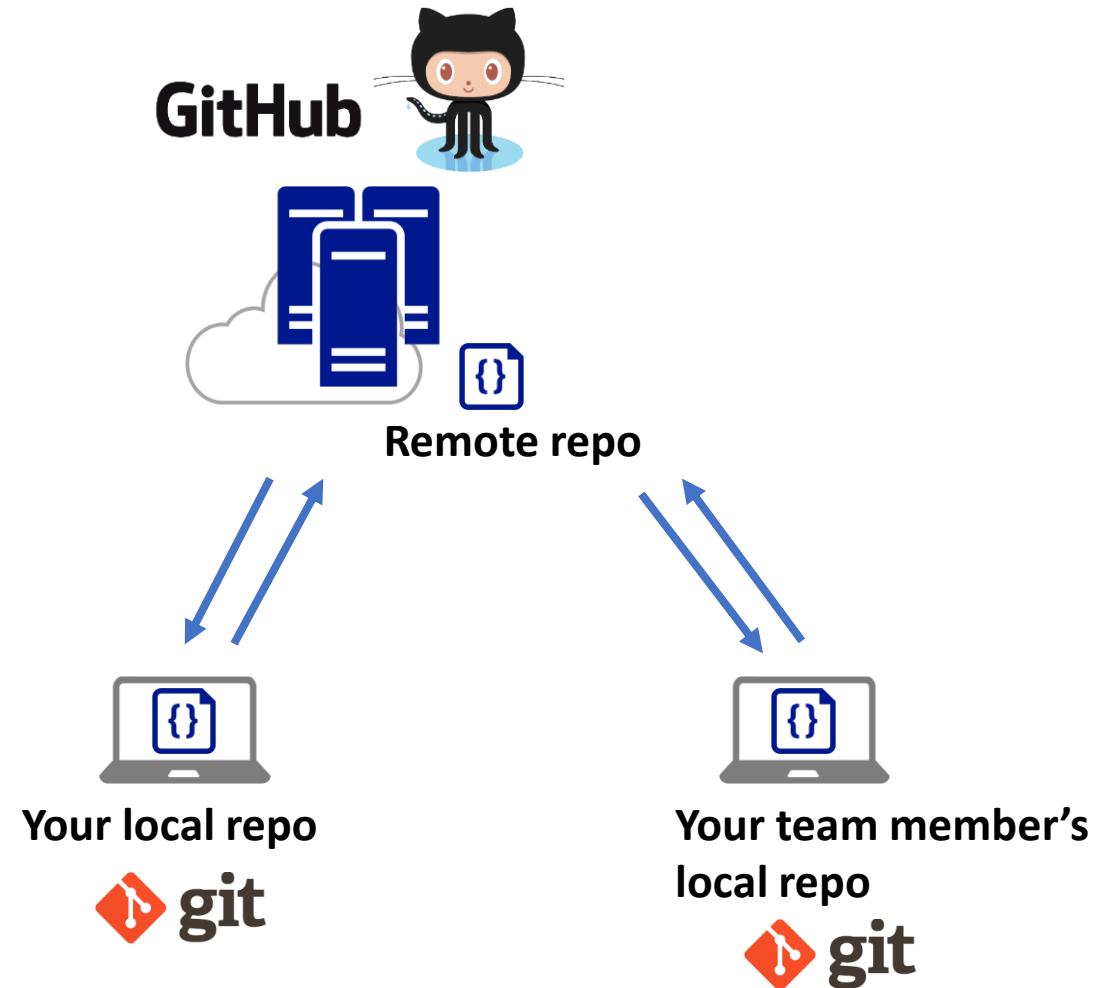


- Version control?
 - keep track of changes: version 1, version 2, etc.
 - like “Track Changes” in MS Word, or “save progress” in game play


What's GitHub

- A git-aware online repo host
- Enable repo sharing and collaboration
 - raise issues, pull request, etc.
- Free public and private repo (*)
- Other repo hosts exist
 - e.g., [bitbucket](https://bitbucket.org/), [gitlab](https://gitlab.com/), etc.

*Ref: <https://github.com/pricing>



What's GitHub (Other than a Git Repo Host)

- [GitHub Pages](#): static web site host
 - The workshop website is hosted on github,
 - <https://tdmdal.github.io/git-workshop-2023-rccl>
 - We will learn how to create a blog site and host it on GitHub in this workshop
 - Like this example, <https://eijoac.github.io/my-blog/>
- [Codespaces](#): online code editor/developer environment
- [Copilot](#): “Don’t fly solo”, and code together with AI! 
- ...

Why Git & GitHub

- **Organize** (record keeping; traceability)
 - Track, compare and undo changes
 - Manage multiple versions/ideas at the same time efficiently
 - Backup your work
- **Share**
 - project code, notes, ideas, etc.
- **Collaborate**
 - Team members (no more emailing code around)
 - open-source community
- Others...
 - e.g., host personal/project website, and blogs on GitHub, i.e., online presence, "[I web, therefore I am a spiderman.](#)"

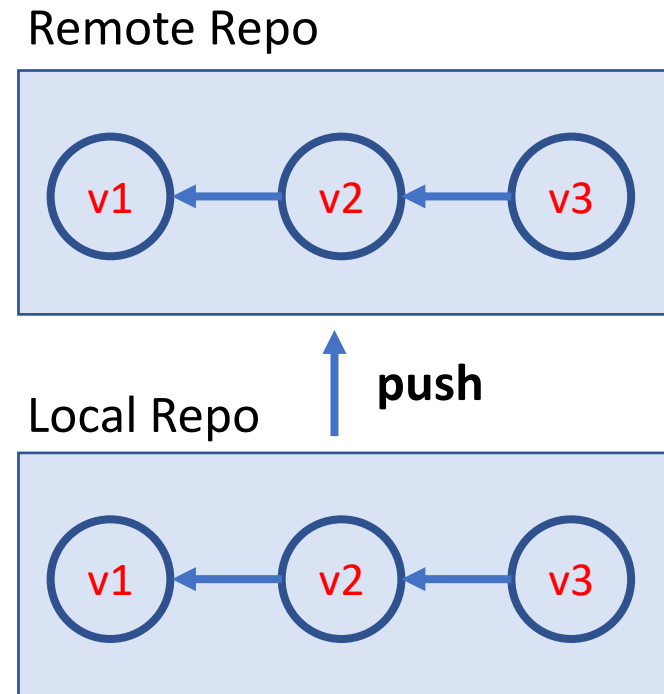


Using Git: GUI Clients vs Command Line

- GUI is easy to get started
 - In this workshop, we will focus on a GUI client, [GitHub Desktop](#)
 - Briefly discuss some underlying concepts & git commands associated with each GUI operation
 - Note that many code editors comes with Git integration too (semi-GUI)
 - e.g., [RStudio](#), [VSCode](#), etc.
- Command line is universal
 - i.e., same commands for Windows, Mac, and Linux
- It's easy to go from command line to a GUI client
 - Not quite vice versa

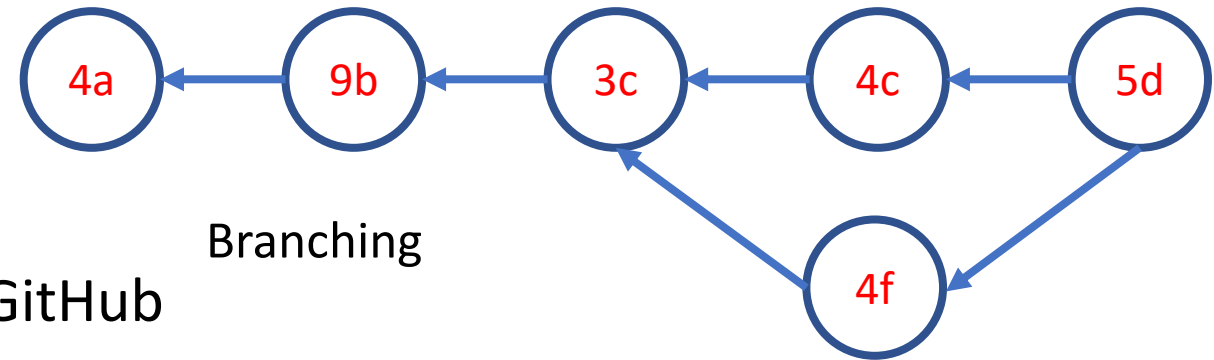
Plan for This Workshop - Today

- Focus on a simple linear workflow
 - manage version history in local repo
 - push local repo to GitHub

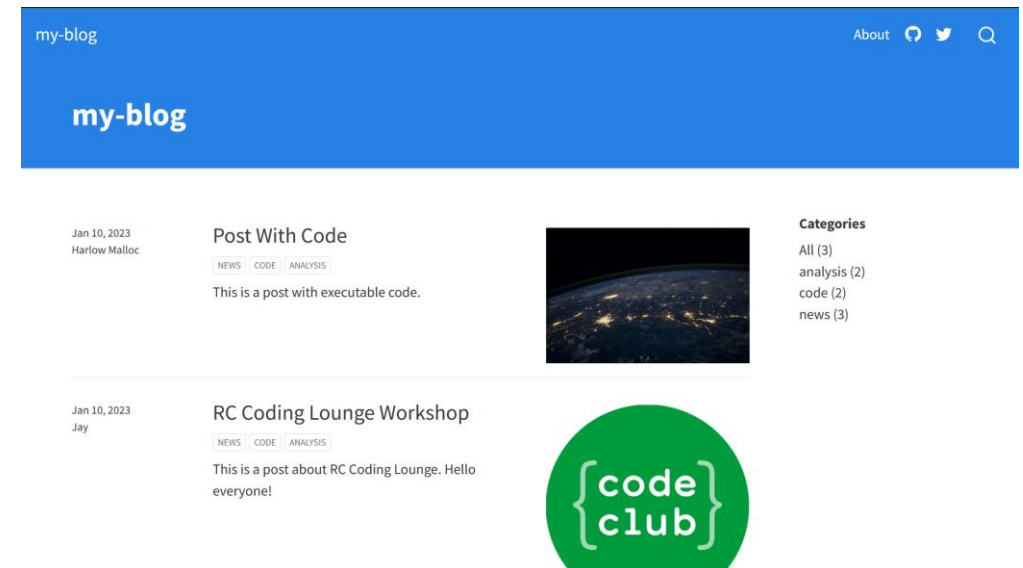


Plan for This Workshop – Next Time

- Intro to
 - a simple branching workflow
 - a simple collaboration workflow via GitHub

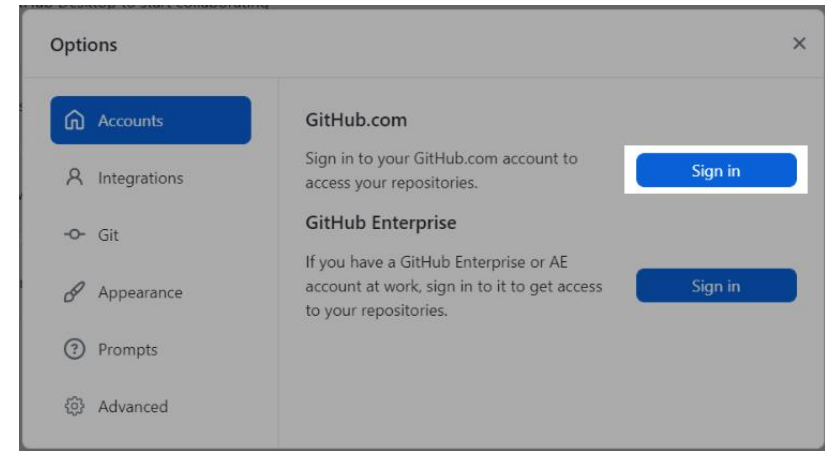


- Host a blog site on GitHub
 - Create a blog site using [Quarto](#)
 - Host it on Github via [GitHub Page](#)



Setup GitHub Desktop

- Step 1: Create a GitHub account, <https://github.com/>
- Step 2: Install GitHub Desktop, <https://desktop.github.com/>
 - Launch GitHub Desktop
 - Sign in GitHub: File → Options... → Accounts
 - Set some global options: File → Options... → Git
 - Configure git for first-time use (`>|:git config`)
- Optional: Install Git (command line): <https://git-scm.com/downloads>



The simplest git workflow (demo)

1. Create a new local git repo
2. Create or make changes to your files/code
3. Snapshot files to prepare versioning (stage the changes)
4. Record version history (commit the changes)
5. repeat (back to 2)...

Check commit history

Compare difference between changes

Create a New Local Git Repo

The image shows the Visual Studio Code interface with the 'Create a new repository' dialog open on the left and the 'No local changes' view on the right. Red numbers 1 through 13 are placed over various UI elements to indicate steps or components.

Left Panel: Create a new repository dialog

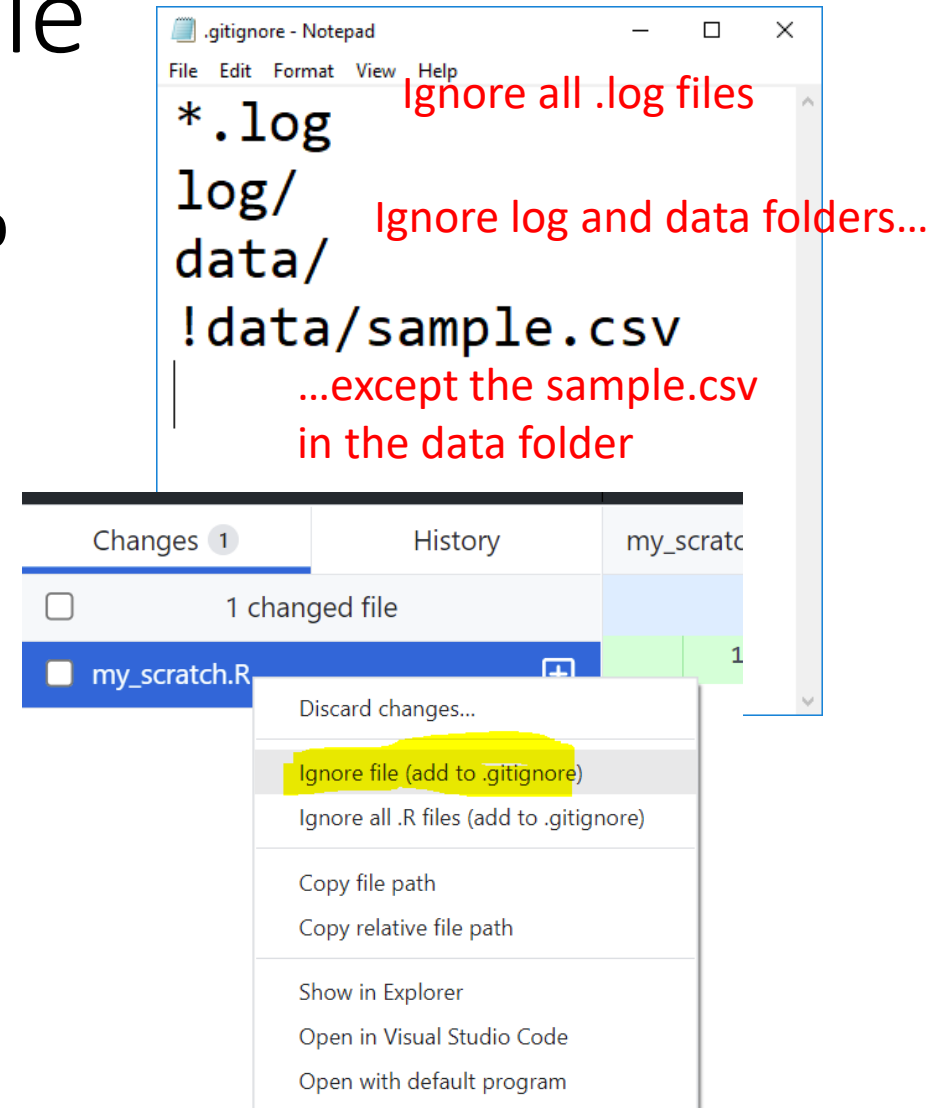
- 1: Name input field (contains 'r-demo-proj')
- 2: Description input field (contains 'a r demo project')
- 3: Local path input field (contains 'C:\Users\jay.cao\Documents')
- 4: ☐ Initialize this repository with a README
- 5: Git ignore dropdown menu (shows 'R')
- 6: License dropdown menu (shows 'None')
- Buttons: 'Create repository' and 'Cancel'

Right Panel: No local changes view

- 7: Current repository dropdown (shows 'r-demo-proj')
- 8: Current branch dropdown (shows 'main')
- 9: Changes tab
- 10: History tab
- 11: Summary (required) input field
- 12: Commit to main button
- 13: Publish repository button (in the top toolbar)
- Buttons: 'Open in Visual Studio Code', 'Show in Explorer', 'Publish repository', 'Commit to main', 'Undo'

Suppress Tracking: .gitignore file

- You may not want to check in and push to GitHub
 - Large or confidential datasets
 - Intermediate or temporary files
 - generated by IDE or a compiler
 - Password/API key files
- a file named .gitignore in your git repo folder
 - e.g., my_proj/.gitignore
- A collection .gitignore templates
 - <https://github.com/github/gitignore>



Stage and Commit

The screenshot displays the Git GUI interface with a dark theme. At the top, the menu bar includes File, Edit, View, Repository, Branch, and Help. Below the menu, the status bar shows the current repository as 'r-demo-proj', the current branch as 'main', and a 'Push origin' button indicating the last fetch was 7 minutes ago with 4 new commits.

The main workspace is divided into three panes. The left pane, titled 'Changes' with a badge for 2 changes, lists '2 changed files': 'test.R' and 'test04.R'. Both files are marked with a checkmark and a yellow highlight. The middle pane shows a diff for 'test.R', with line 7 highlighted in green, indicating a new addition: '+b <- a'. The right pane shows the file content for 'test.R'.

At the bottom, the commit summary section includes a 'Summary (required)' field (highlighted in yellow), a 'Description' text area, and a 'Commit to main' button (highlighted in green). Below this, it shows the commit history: 'Committed 16 minutes ago' and 'Revert "add b variable and test04"', with an 'Undo' button.

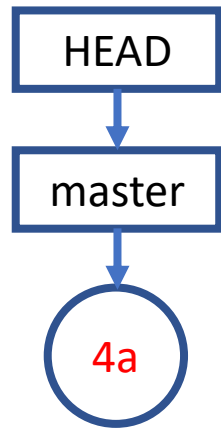
The simplest git workflow (FYR)

1. Create a new local git repo: `git init`
2. Create or make changes to your files/code
3. Snapshot files to prepare versioning (stage the changes): `git add`
4. Record version history (commit the changes): `git commit`
5. repeat (back to 2)...

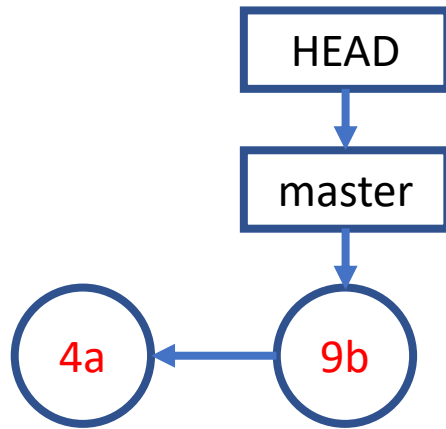
Check commit history: `git log`; `git show`

Compare difference between changes: `git diff`

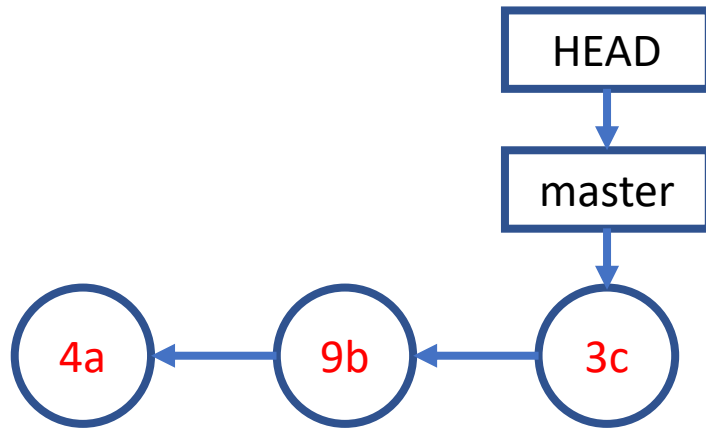
Git Concepts – First commit



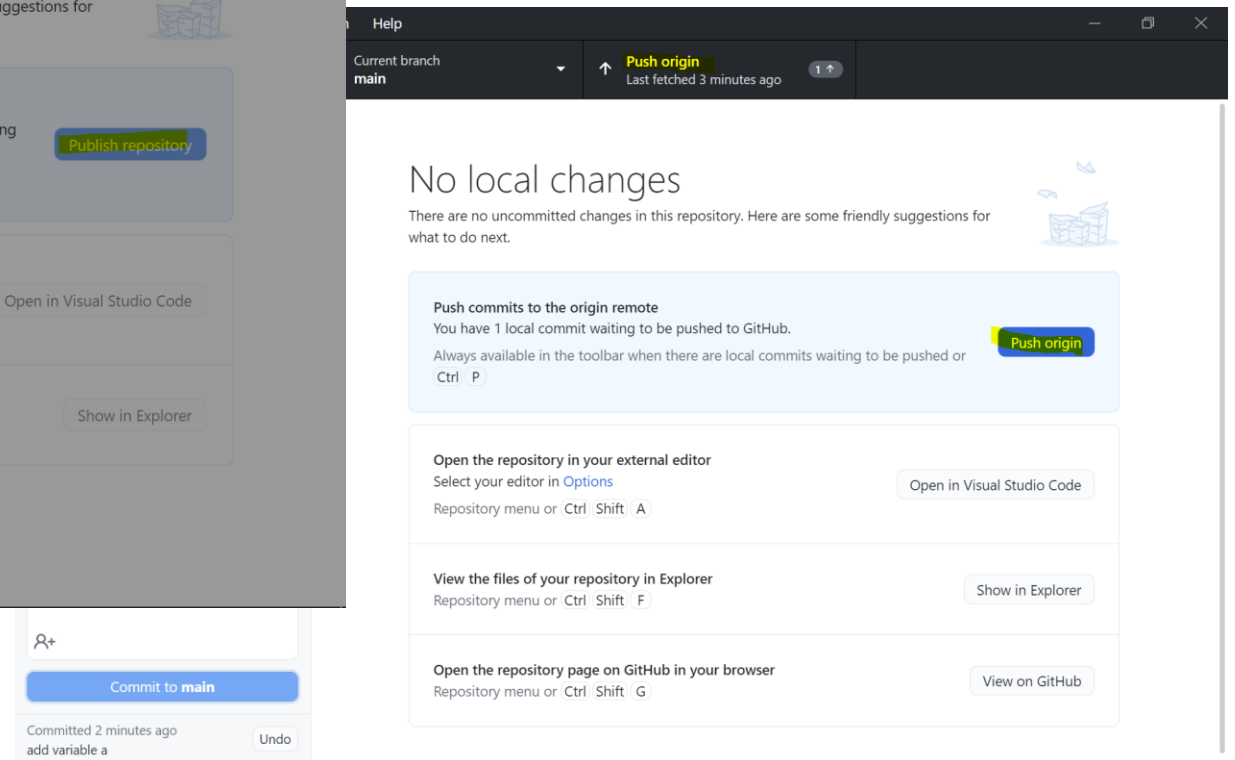
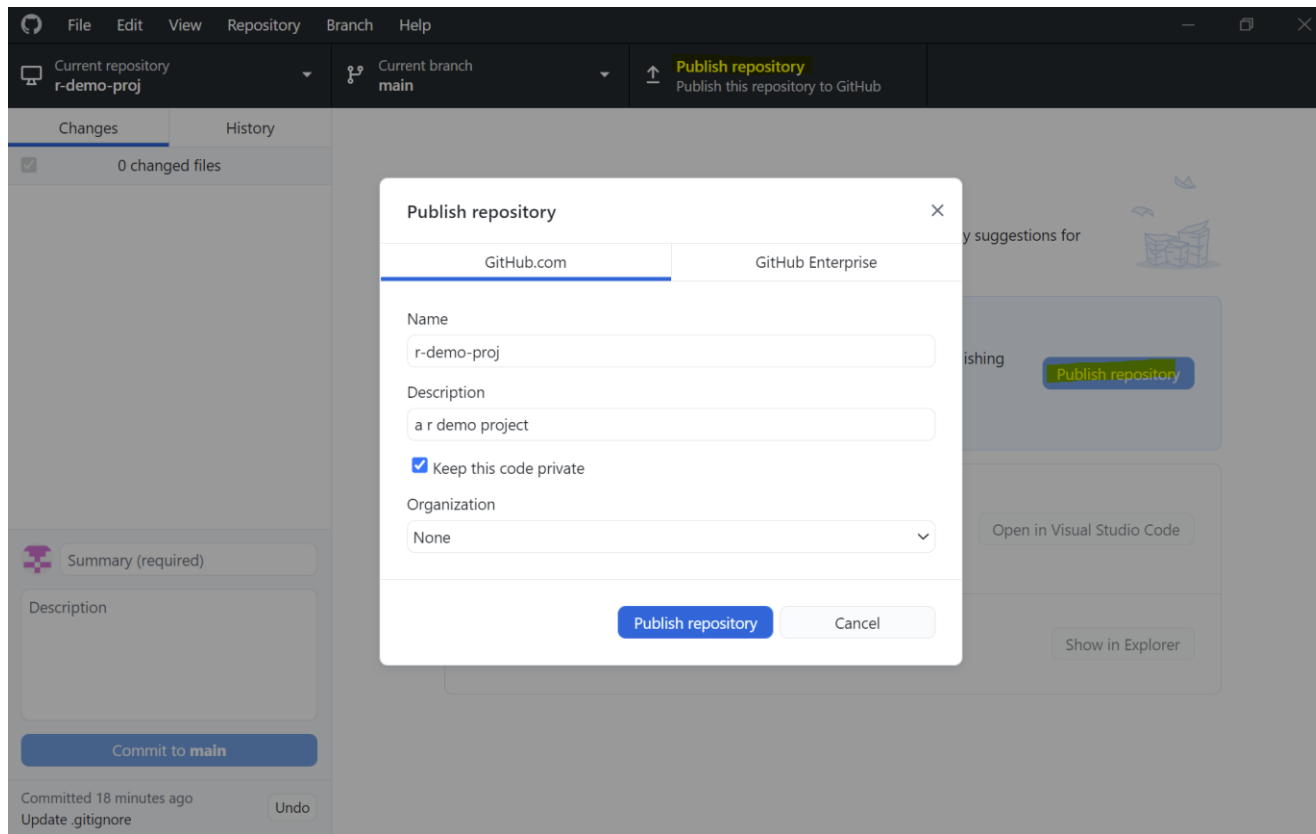
Git Concepts – Second commit



Git Concepts – Third commit and so on...



Publish/Push Local Repo to GitHub (demo)



Publish/Push Local Repo to GitHub (FYR)

- Create a GitHub project repo
- Push your code there
 - backup
 - collaborate with your co-authors
 - collaborate with open-source community

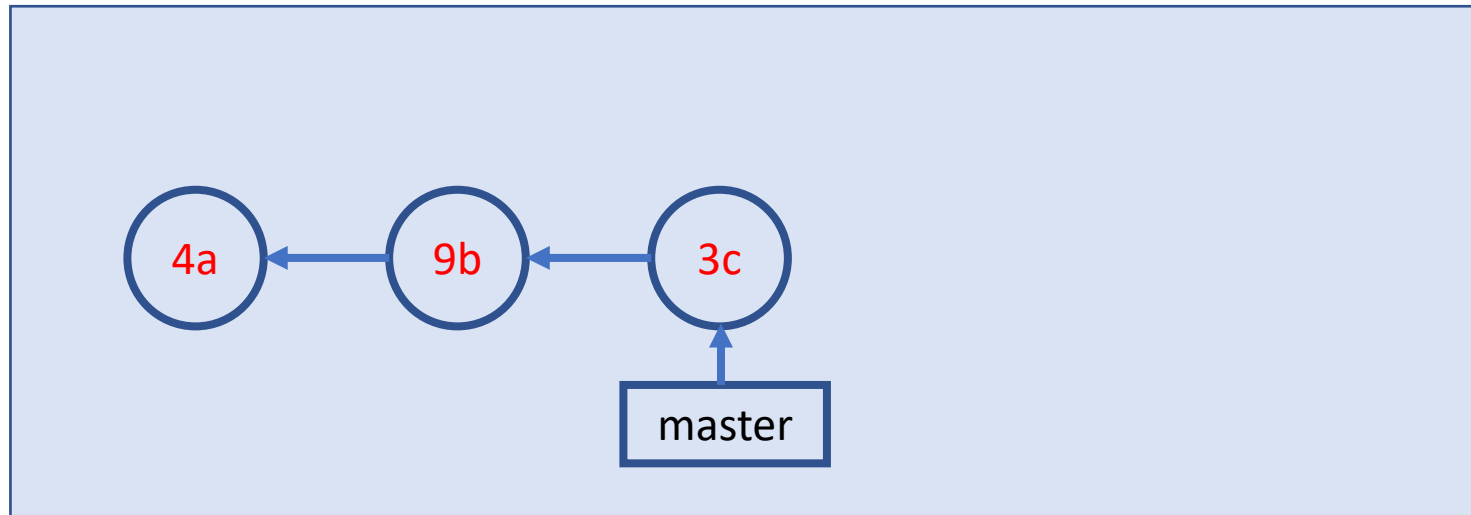
```
git remote add  
git push
```

A Simple Remote Repo Workflow

Remote Repo

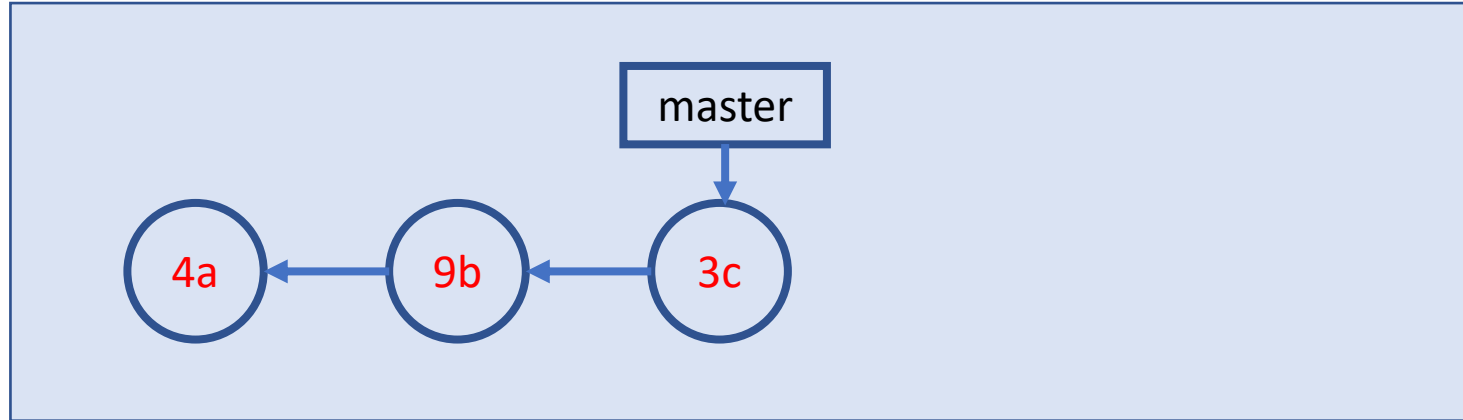


Local Repo

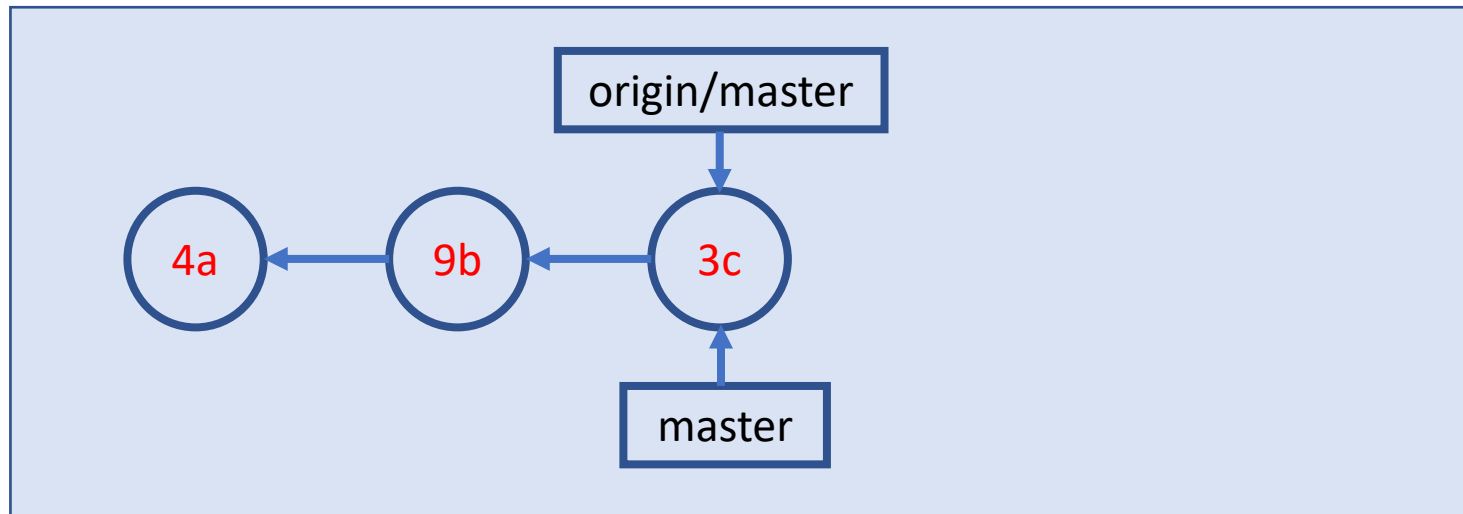


A Simple Remote Repo Workflow `git push`

Remote Repo

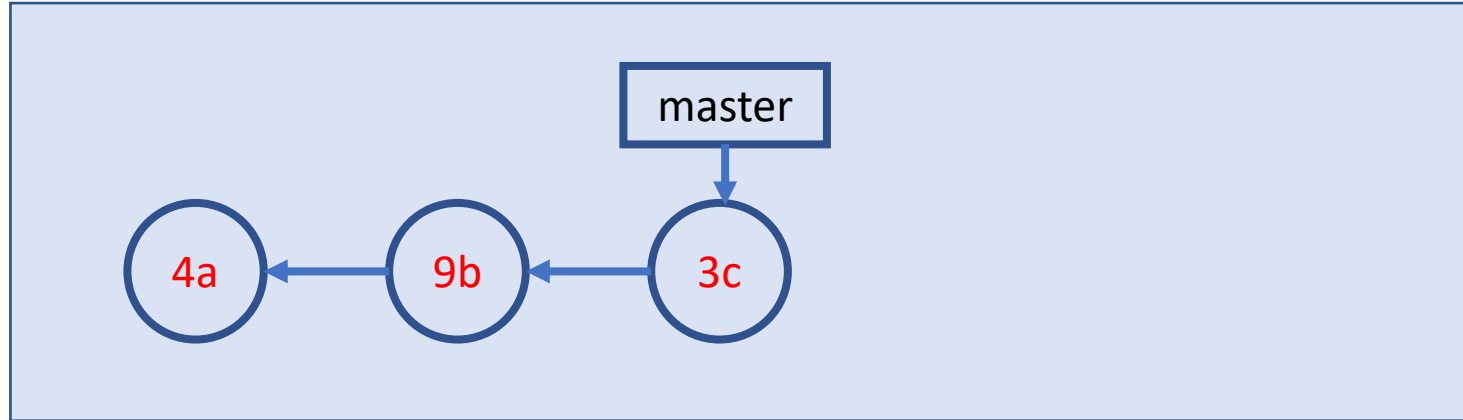


Local Repo

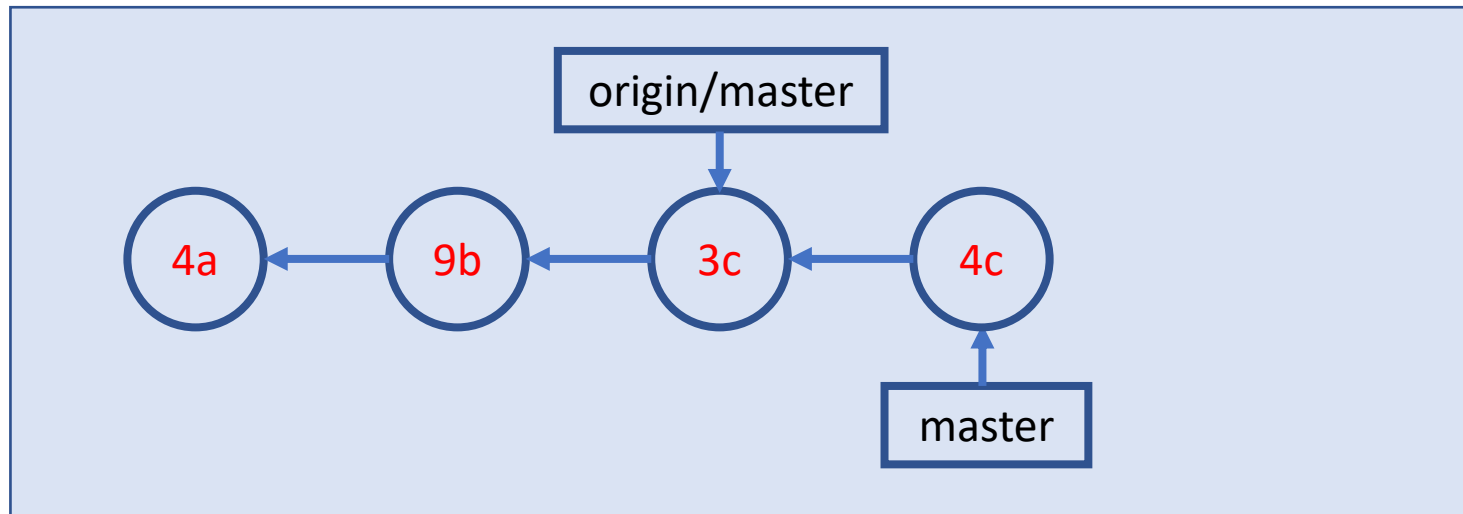


A Simple Remote Repo Workflow

Remote Repo

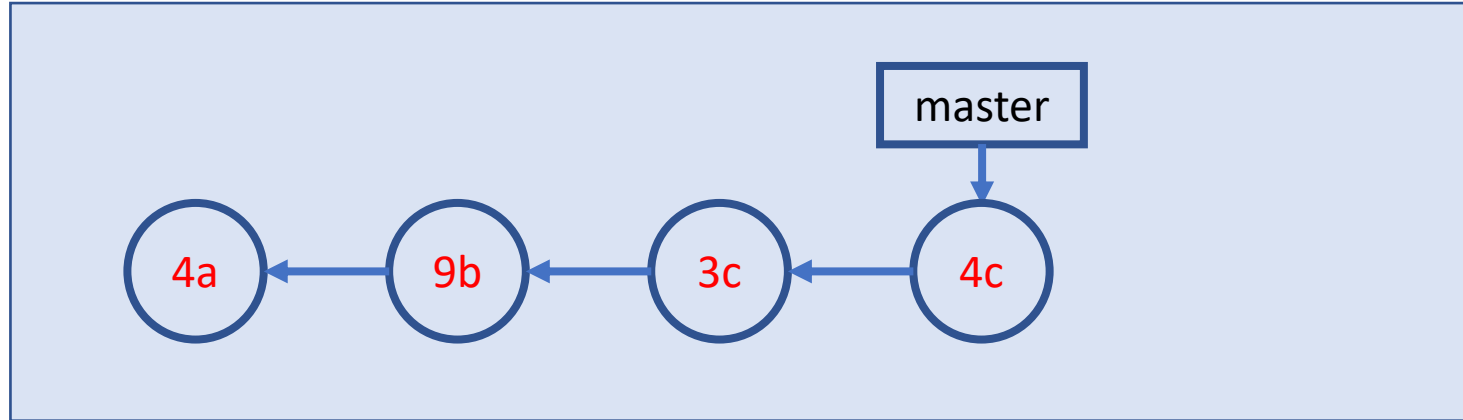


Local Repo

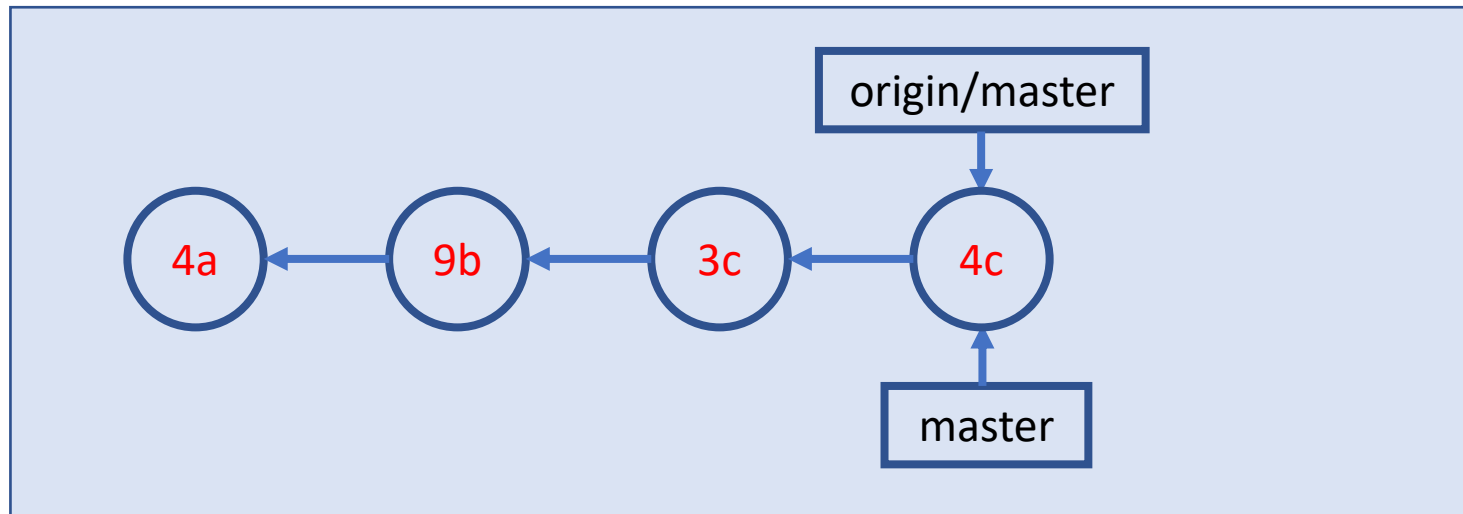


A Simple Remote Repo Workflow `git push`

Remote Repo

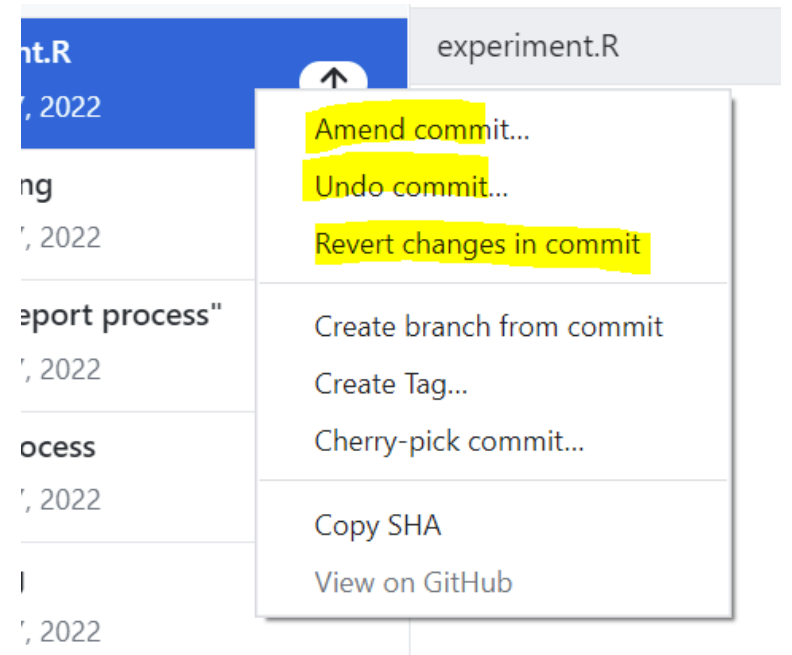


Local Repo



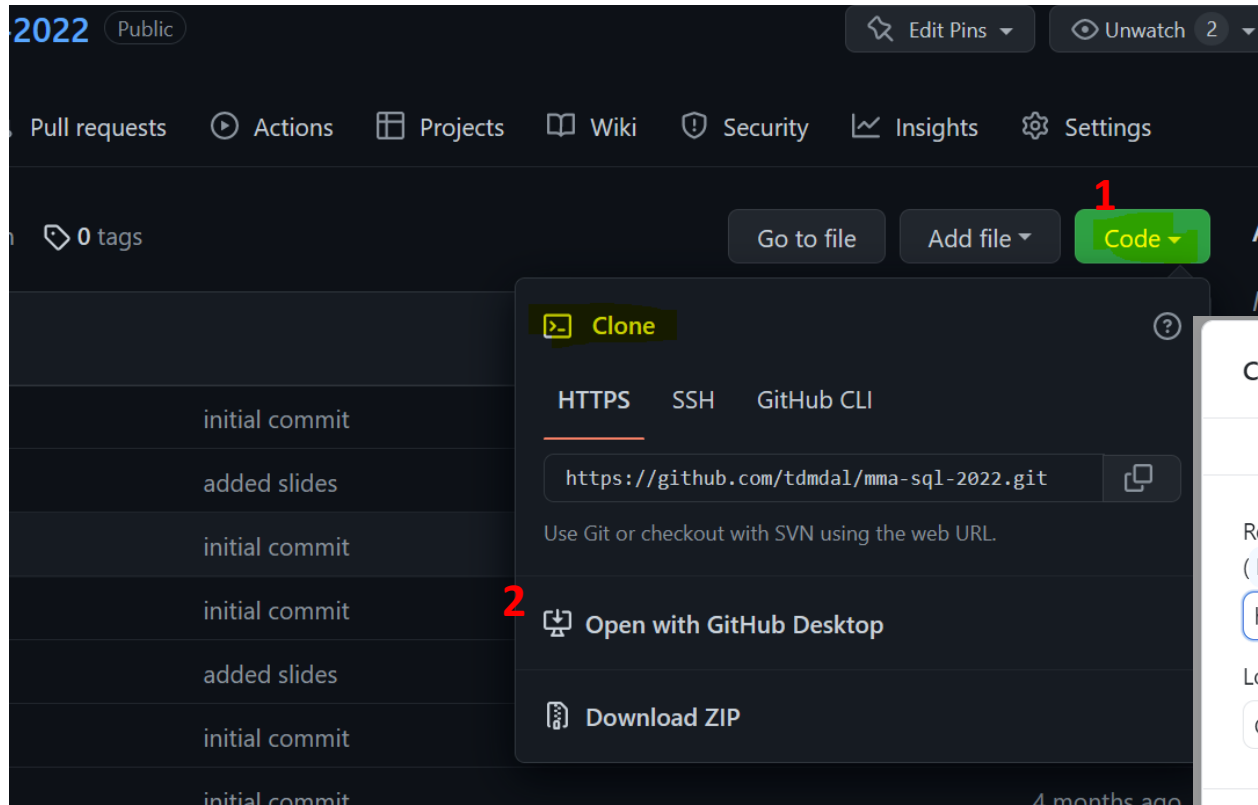
Amend, Undo, Revert, Remove & Rename

- Amend the last commit: change commit message or add new files to commit
 - In principle, don't do it if the commit is already pushed
- Undo the last commit: “uncommit” the last commit
 - Disabled by GitHub Desktop if the commit is already pushed
 - In general, don't change history
- Revert a previous commit: revert a previous code change, and commit it
 - May need to resolve conflict
- Remove or Rename a file

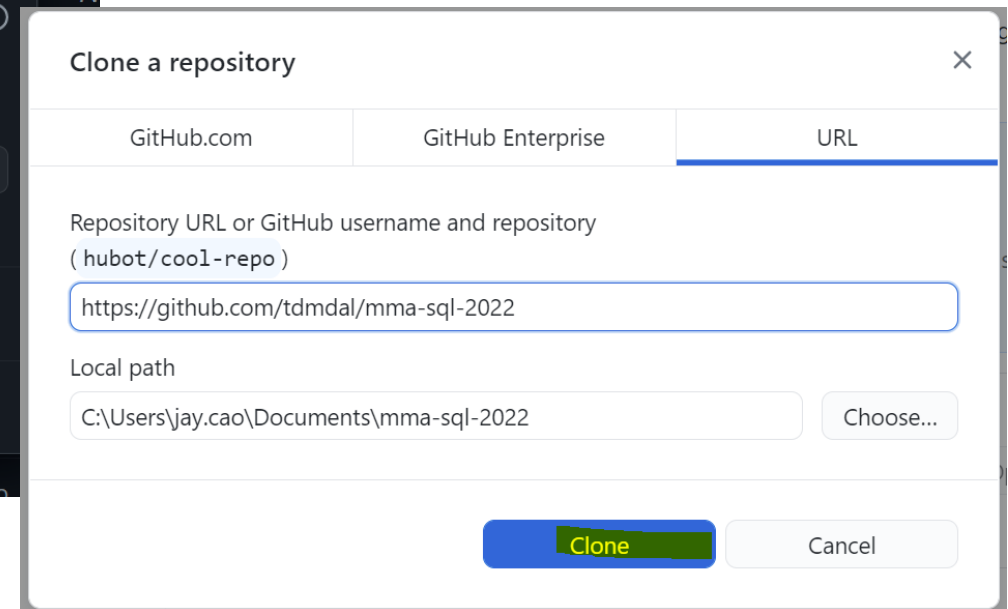


Note: Many other “undo” type of operations can be done in command line.

Clone a GitHub Repo



A repo on GitHub



Clone a GitHub Repo (FYR)

- Clone a GitHub Repo `git clone`
 - Clone your co-author's code (which you have granted access to)
 - Use a public repo as your project starting point
- What is Fork?

Many more to explore... (when needed)

- Git concept / command
 - branch & remote branch
 - merge conflict
 - git reset
 - git stash, rebase, bisect
 - ...
- Git best practice
 - workflows
 - commit size / message
 - ...



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

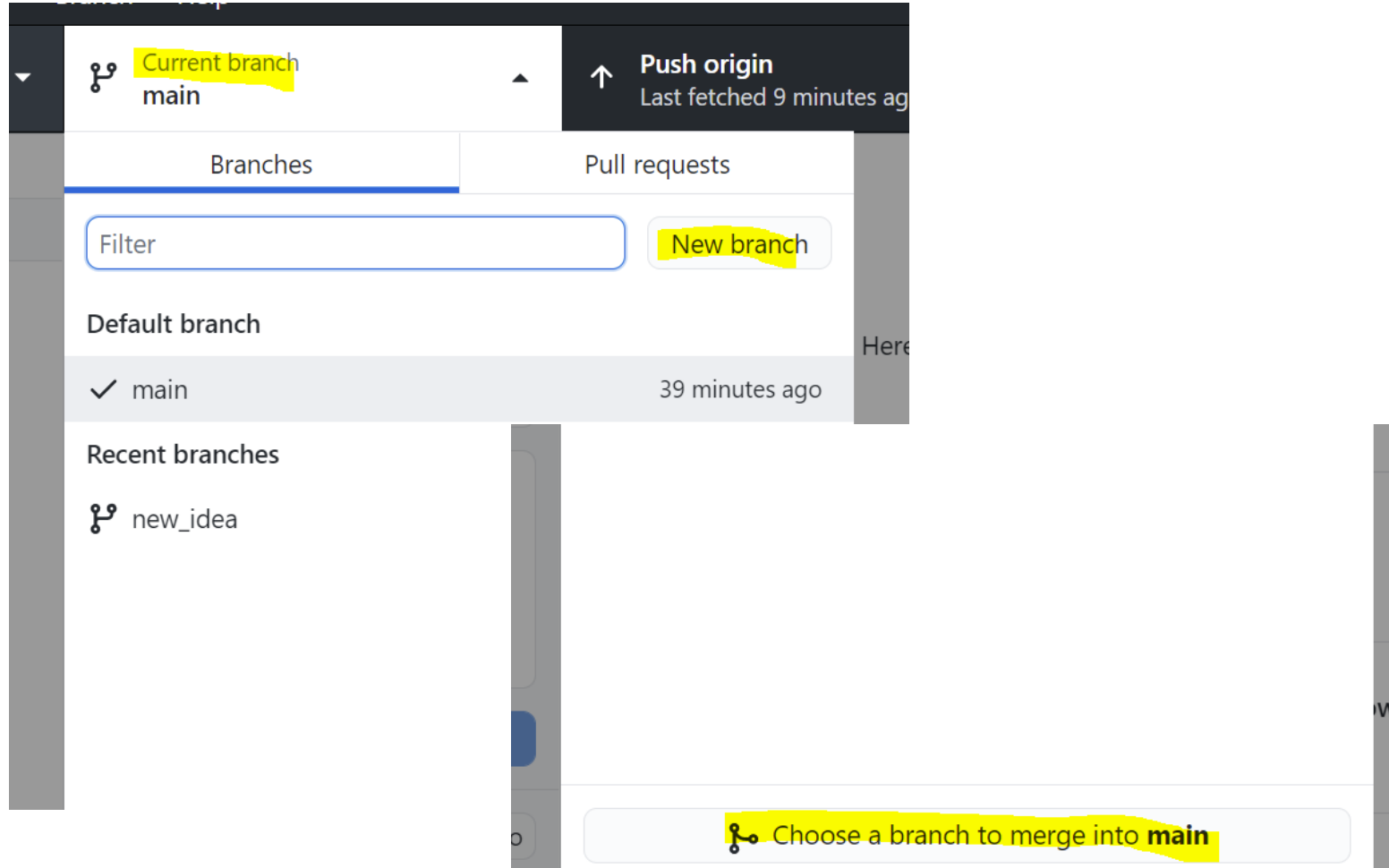
Source: <https://xkcd.com/1296/>

Resources

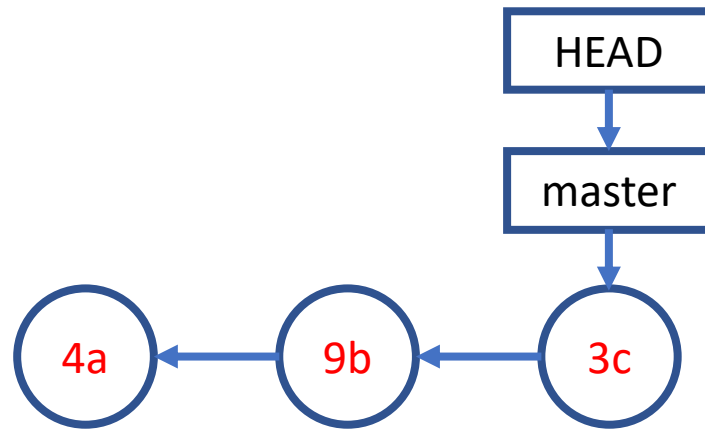
- Git/GitHub with GitHub Desktop
 - [Youtube Video](#) by Coder Coder (22mins; great review for today's workshop)
- Git Command Line Tutorials
 - [Version Control with Git](#) by Software Carpentry
 - [Git Essential Training](#) by Kevin Skoglund at LinkedIn Learning
 - Faculty and staff login from [here](#) for UofT free access
 - Toronto Public Library free access [here](#) for everyone with a library card
 - [Get Started Tutorials](#) from Bitbucket Atlassian
 - [Getting Started with Git](#) from GitHub
- Git Ref Book: <https://git-scm.com/book/en/v2>

Two More Git Workflows

Branch and Merge (demo)

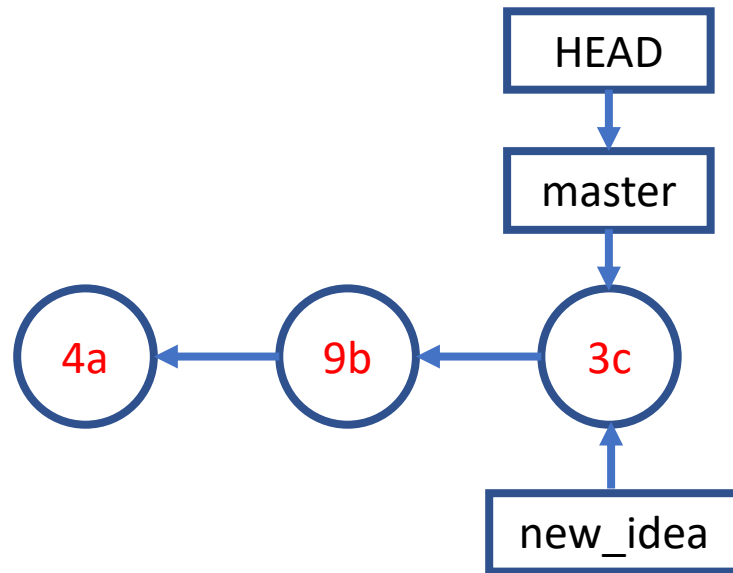


A Simple Branching Workflow



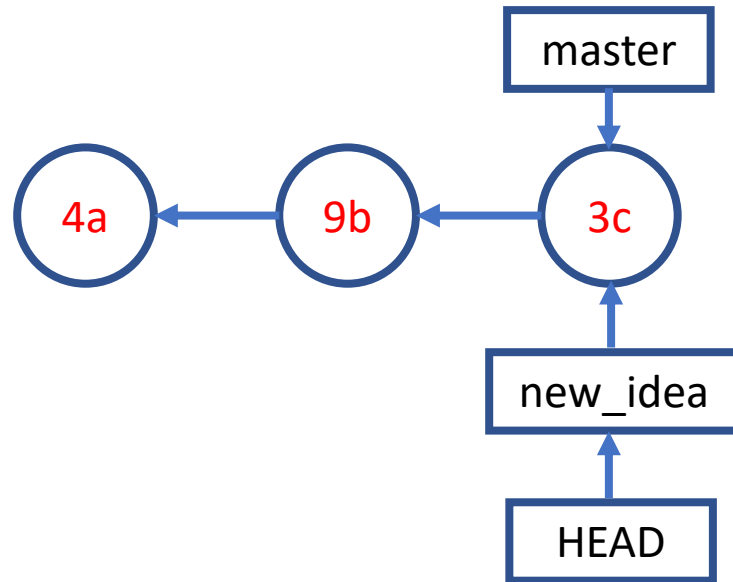
A Simple Branching Workflow

```
git branch new_idea
```



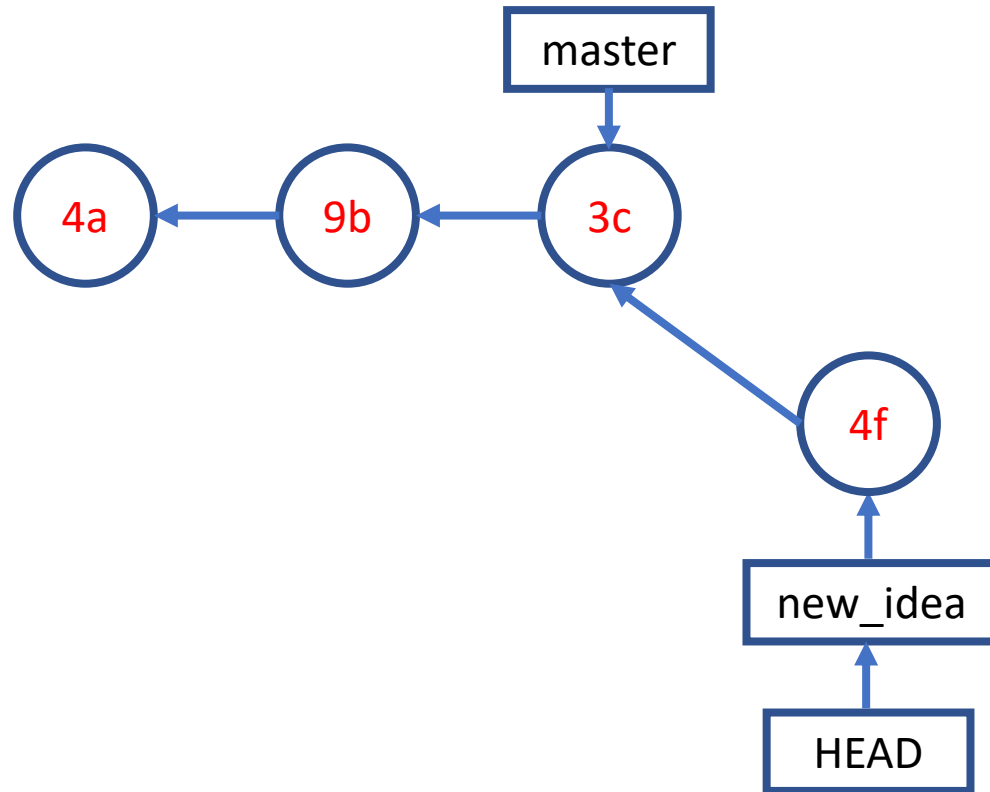
A Simple Branching Workflow

```
git checkout new_idea
```



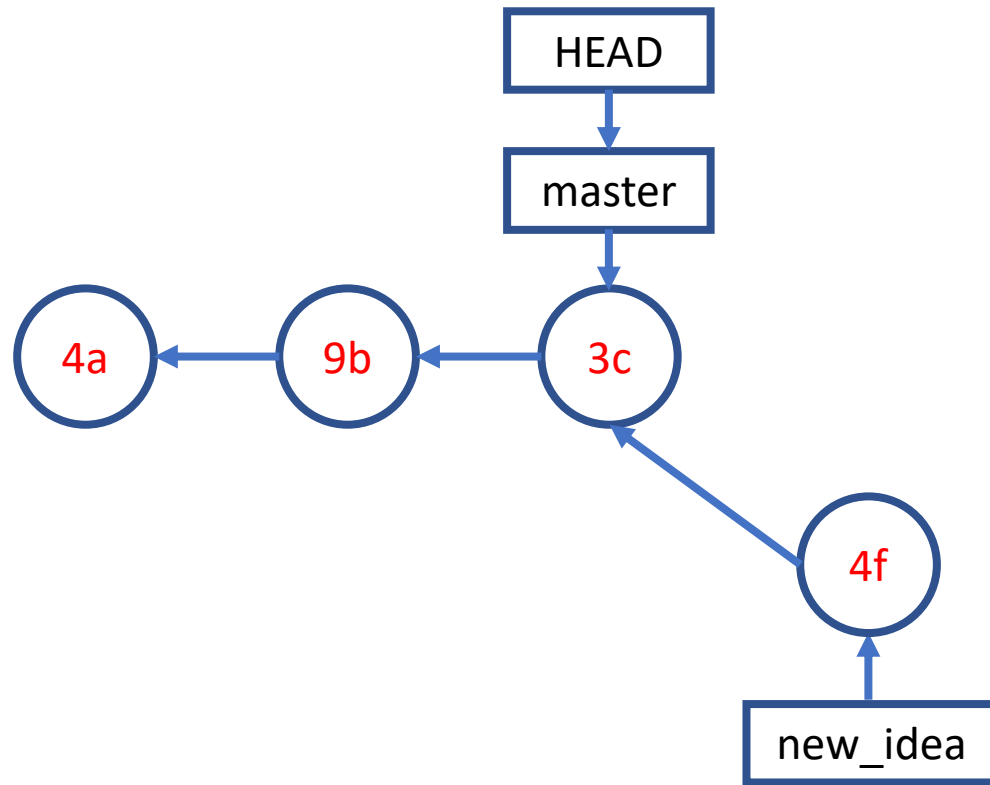
A Simple Branching Workflow

```
git add; git commit;
```



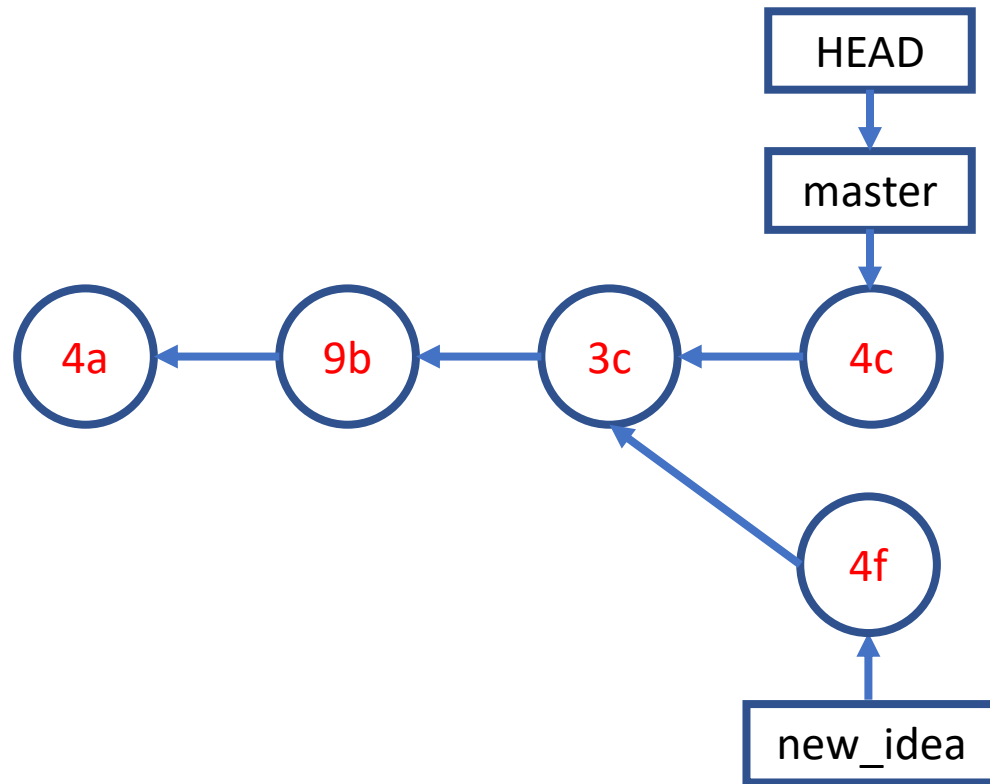
A Simple Branching Workflow

`git checkout master`



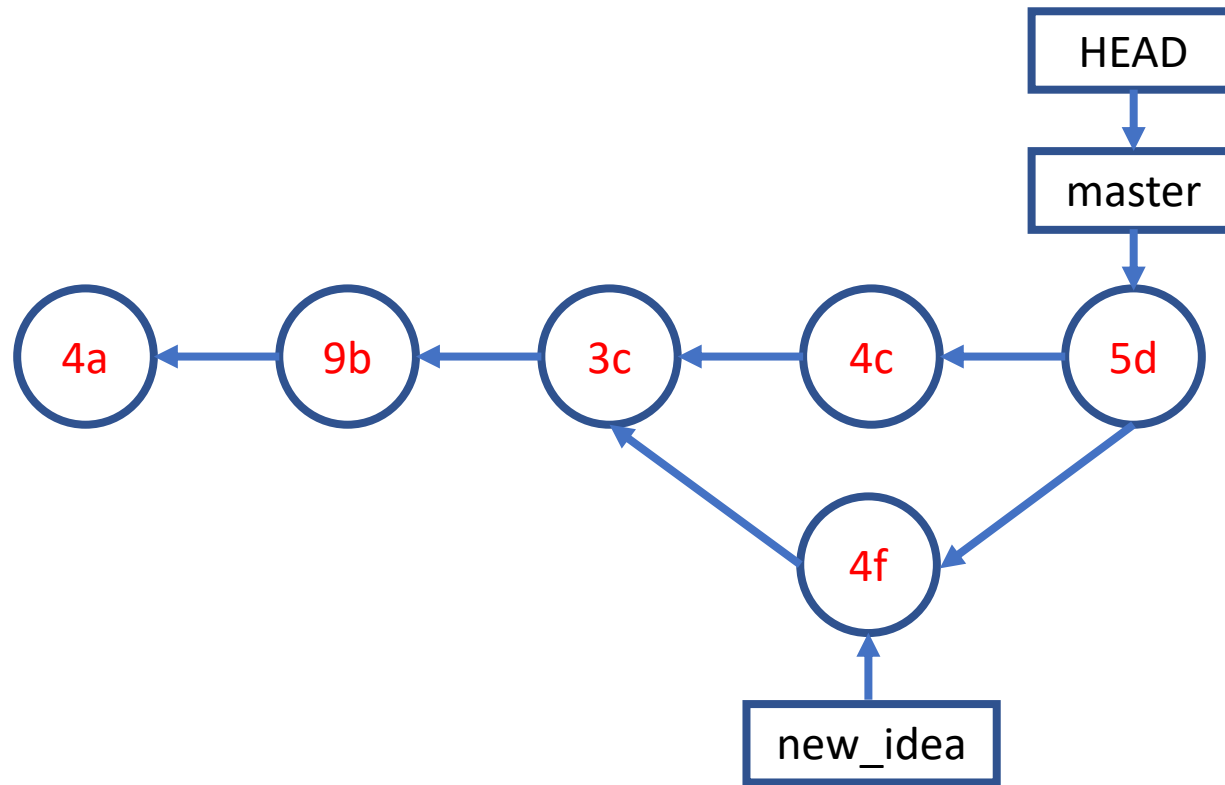
A Simple Branching Workflow

```
git add; git commit;
```



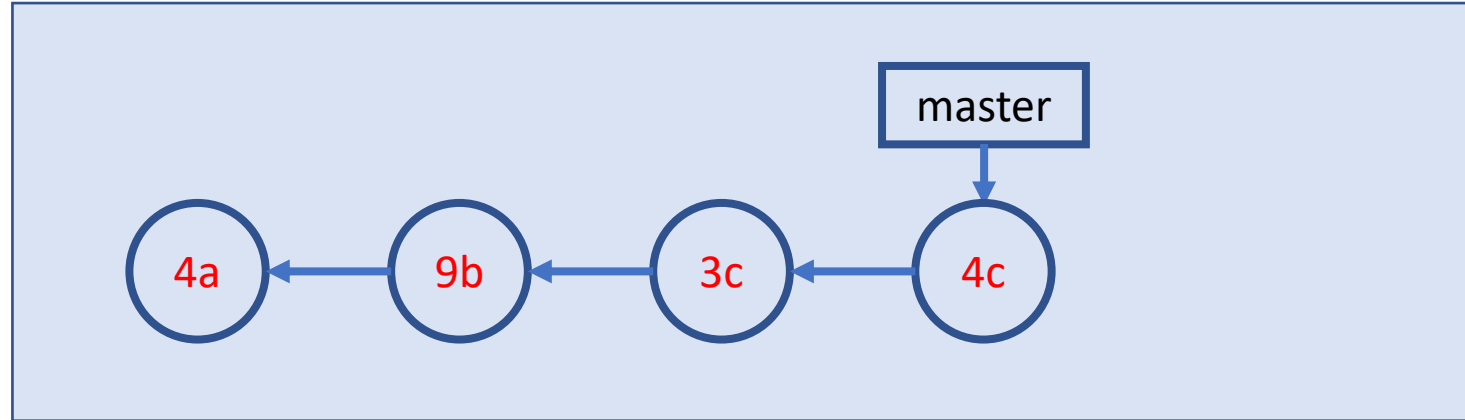
A Simple Branching Workflow

```
git merge new_idea
```

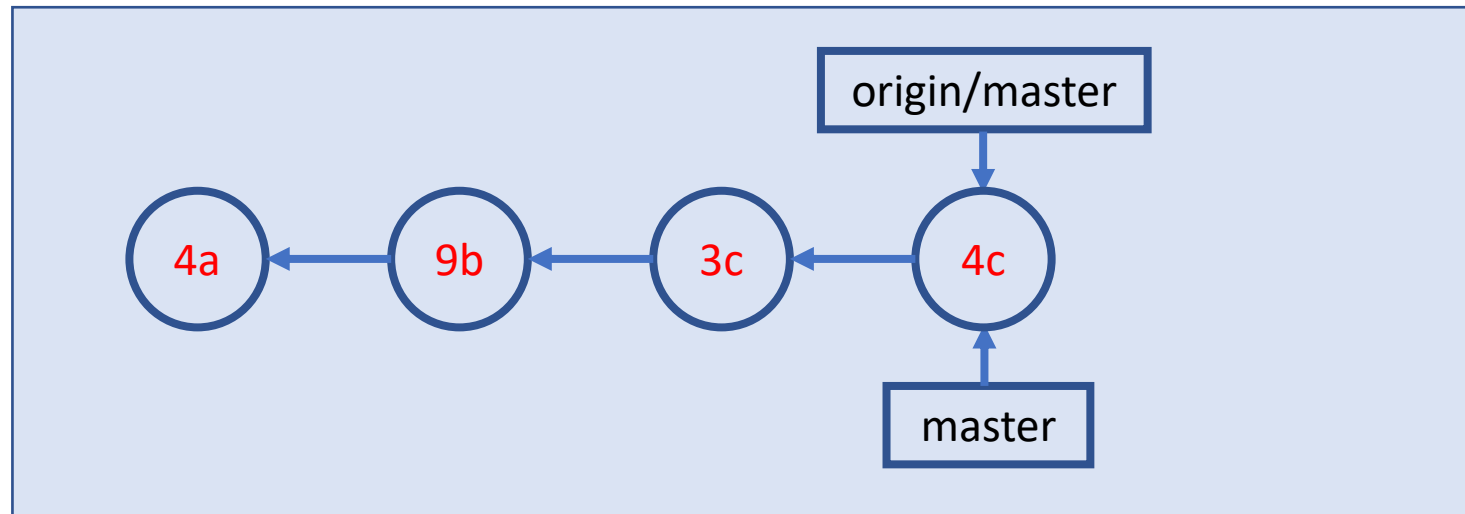


A Simple Collaboration Workflow

Remote Repo

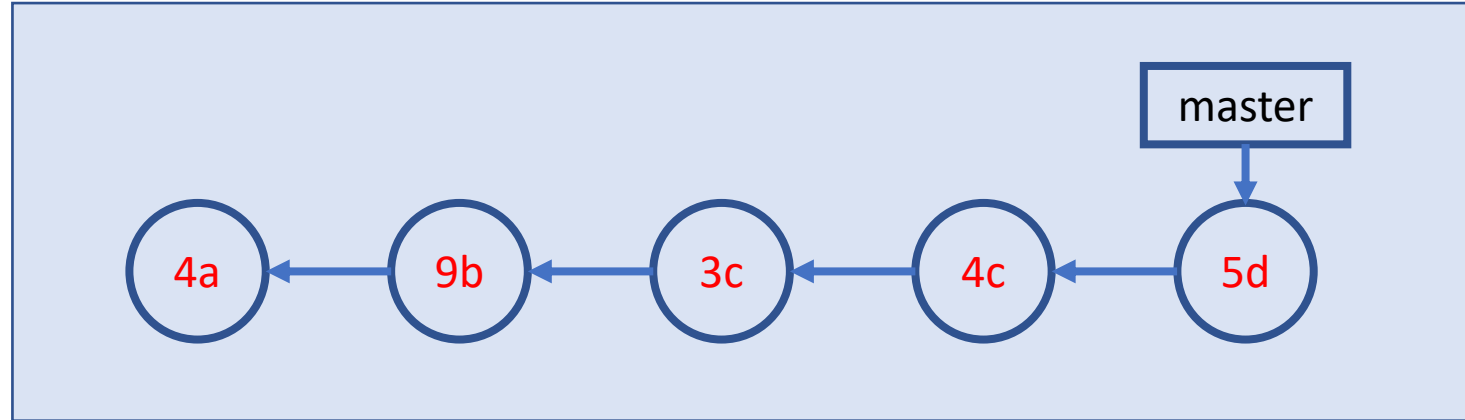


Local Repo

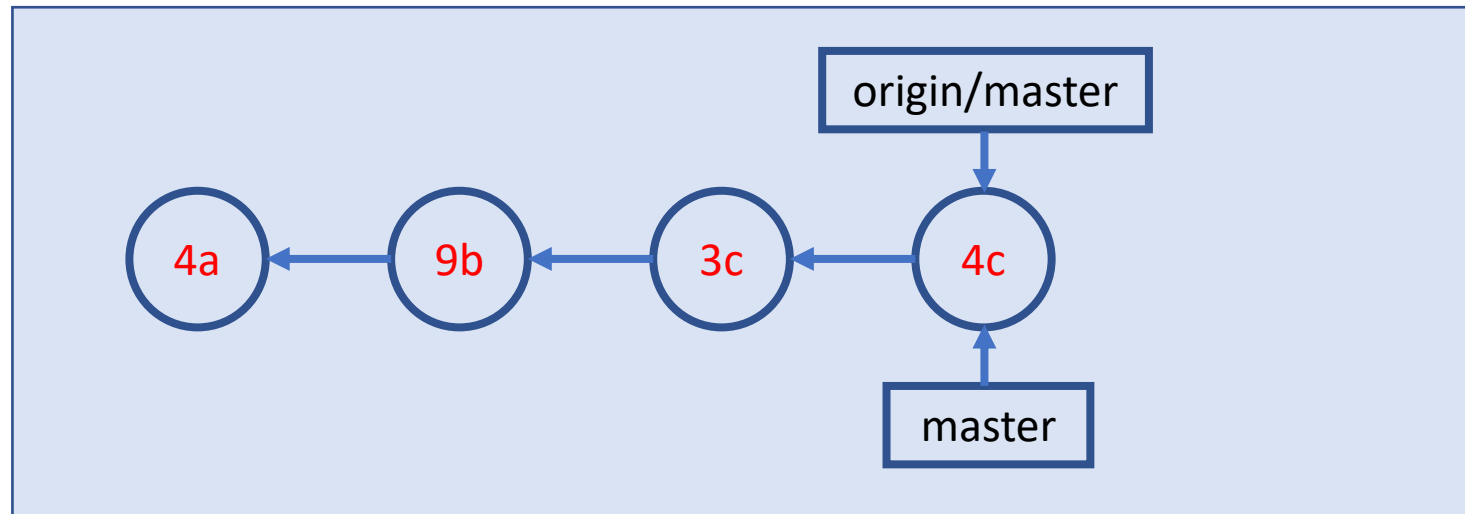


A Simple Collaboration Workflow

Remote Repo

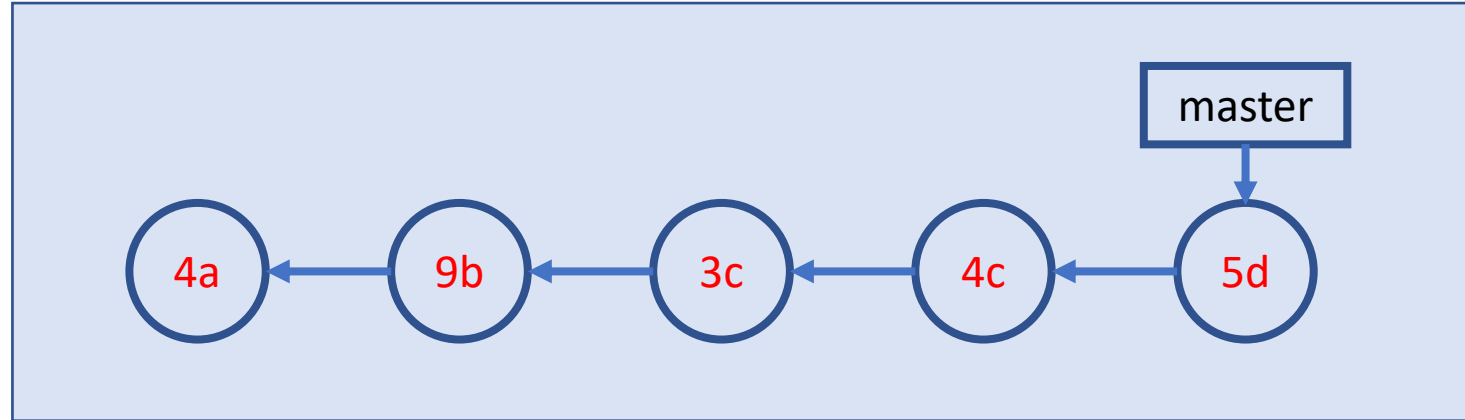


Local Repo

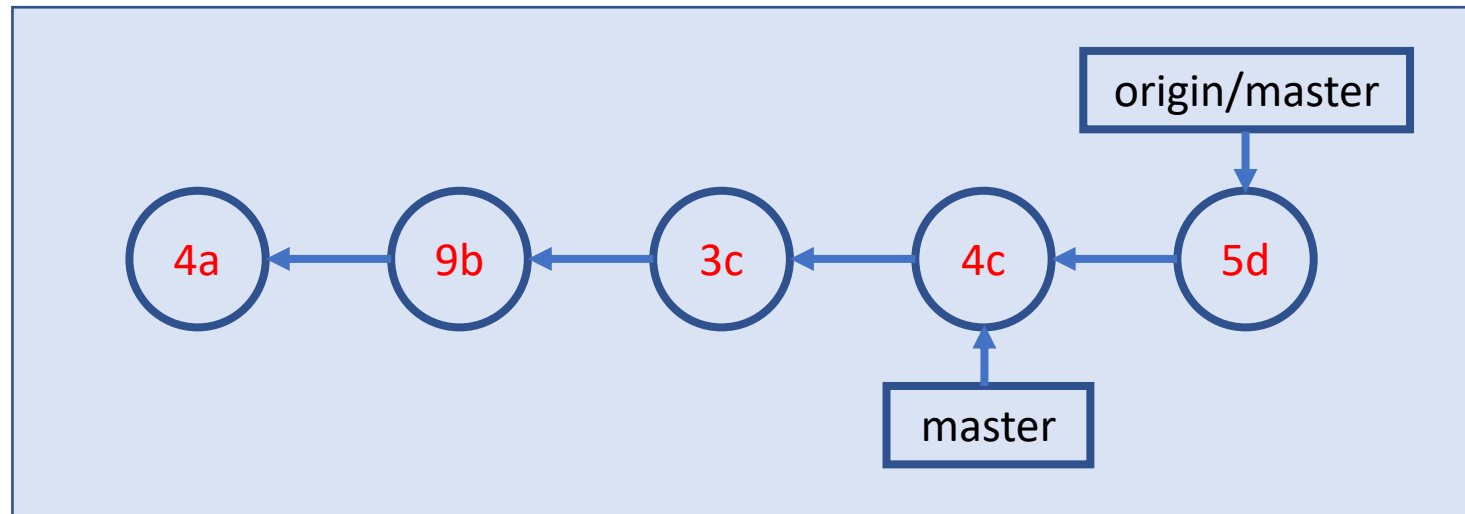


A Simple Collaboration Workflow `git fetch`

Remote Repo

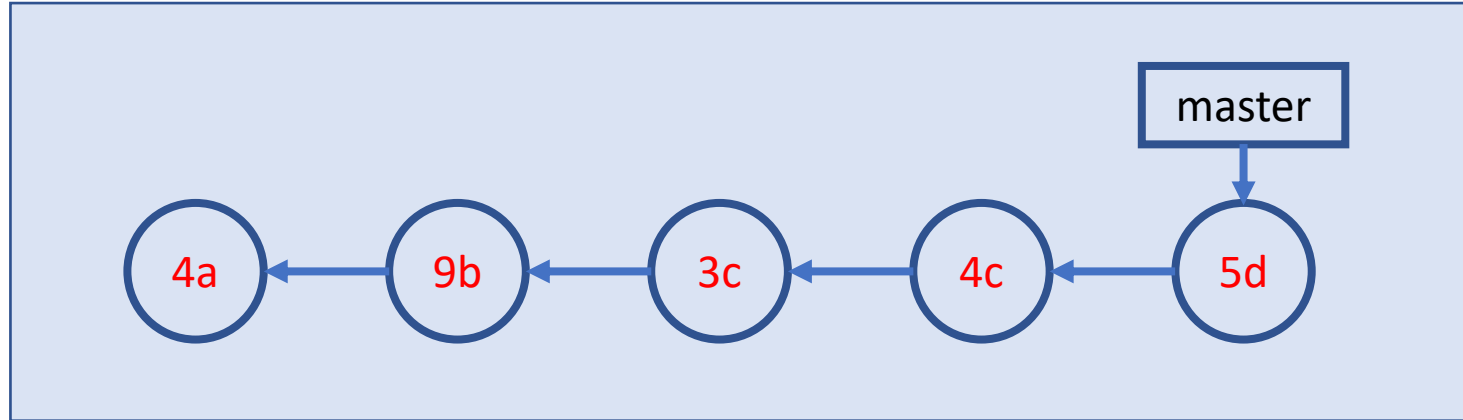


Local Repo

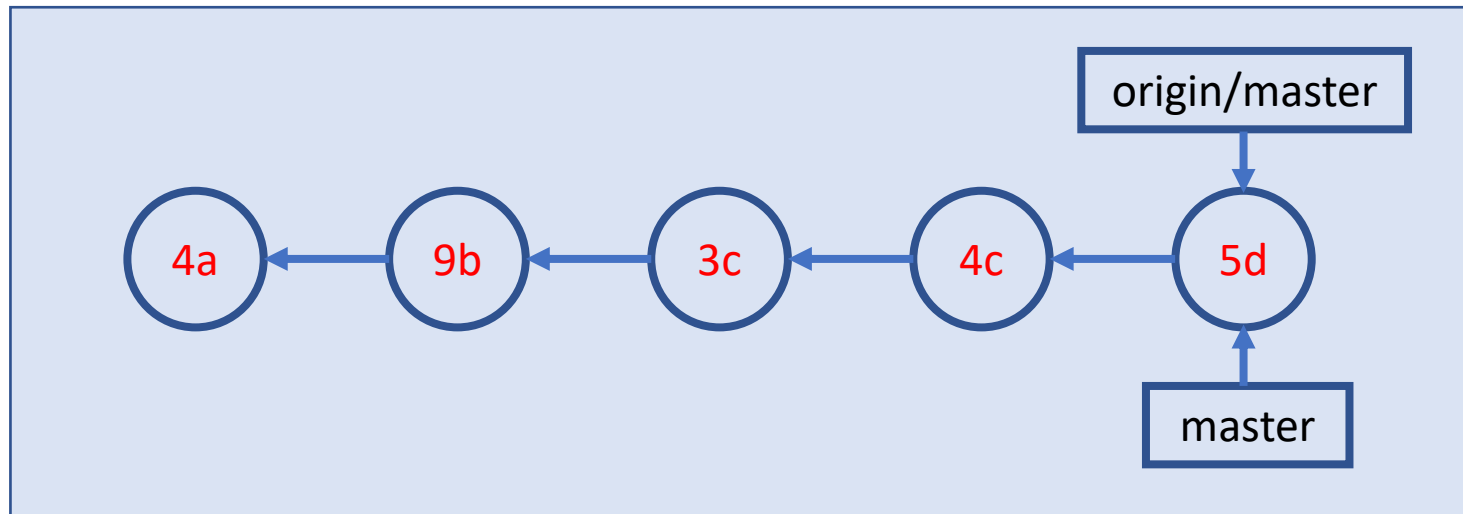


A Simple Collaboration Workflow `git merge`

Remote Repo



Local Repo



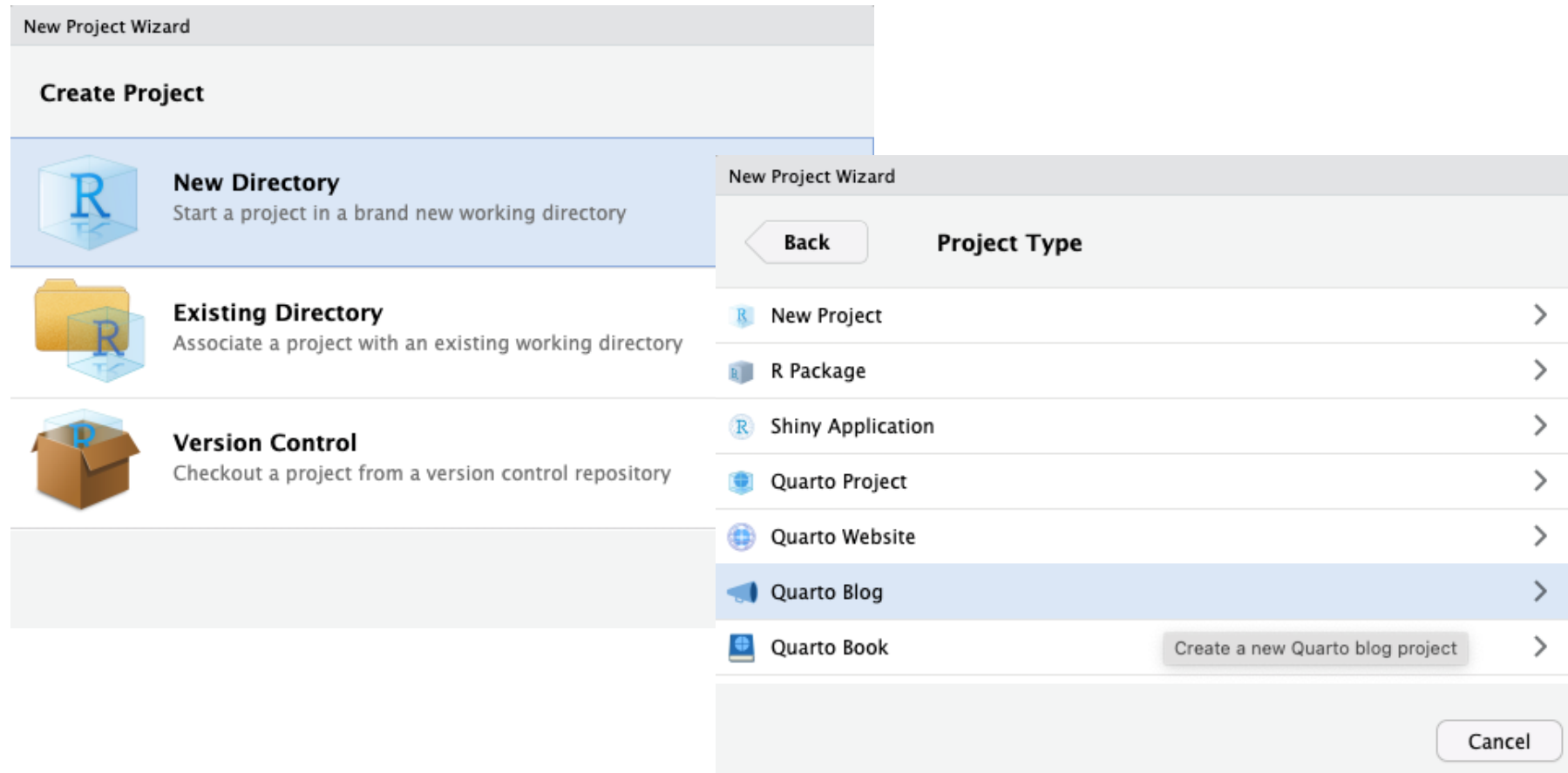
Source: [Git Essential Training](#) by Kevin Skoglund on LinkedIn Learning; Note: git pull = git fetch + git merge

Build/Author a Blog Site & Host it on GitHub

Quarto

- An authoring and publishing system built on [Pandoc](#)
 - The workshop website is built using Quarto
- Authoring uses markdown
- Output can be html (website), PDF, or Word Doc.

Build a Blog Site



Note: You don't have to use R and RStudio to use Quarto

Ref: <https://quarto.org/docs/websites/website-blog.html>

Publish via GitHub Pages

jjallaire / website-publish Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights **Settings** ¹

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Pages

Security

GitHub Pages

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

Your site is live at <https://jjallaire.github.io/website-publish/>
Last deployed by github-pages 2 days ago

Build and deployment

Source ²

Deploy from a branch

Branch

Your GitHub Pages site is currently being built from the gh-pages branch. [Learn more.](#)

³ main ⁴ /docs Save

```
_quarto.yml

project:
  type: website
  output-dir: docs
```

Specify Quarto output folder

Ref: <https://quarto.org/docs/publishing/github-pages.html>