

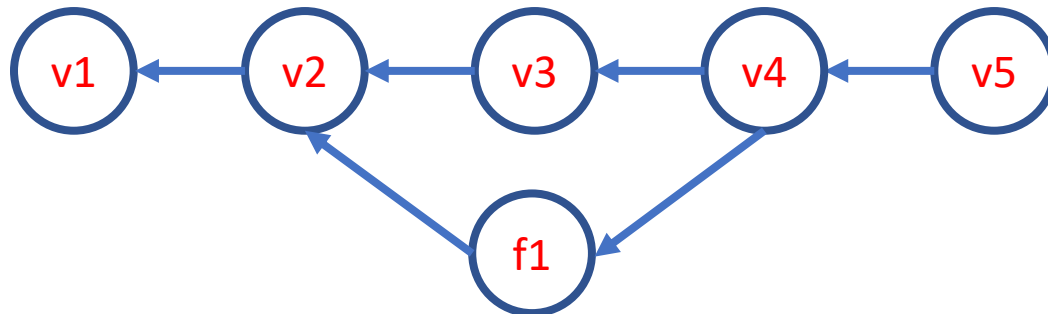
Intro to Git & GitHub

Jay / [TDMDAL](#)

Website for this workshop: <https://tdmdal.github.io/git-workshop-2022>

What's Git git

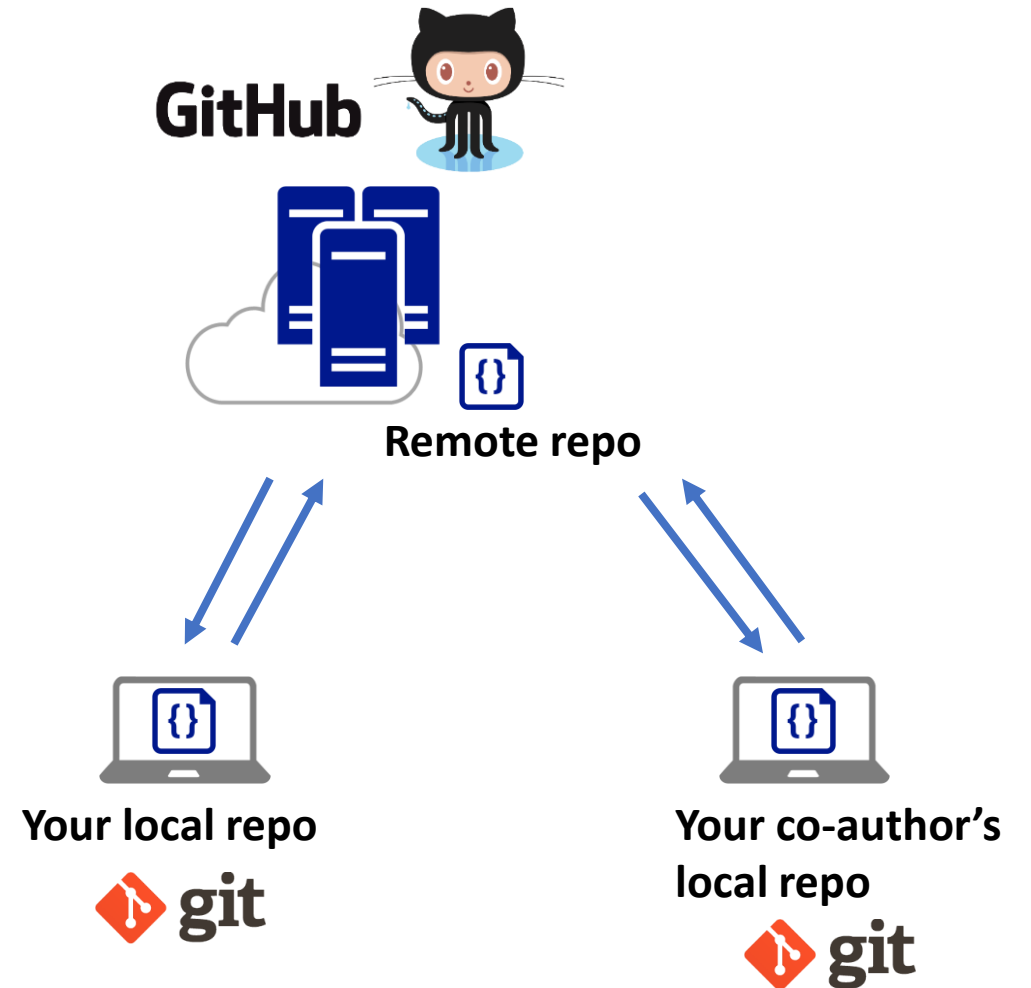
- A version control system
 - manage the evolution of a set of files (repository / repo)
 - usually for source code or text files
 - **NOT** for large datasets, but see [git lfs](#) and [github lfs](#)
- Version control?
 - keep track of changes: version 1, version 2, etc.
 - like “Track Changes” in 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>

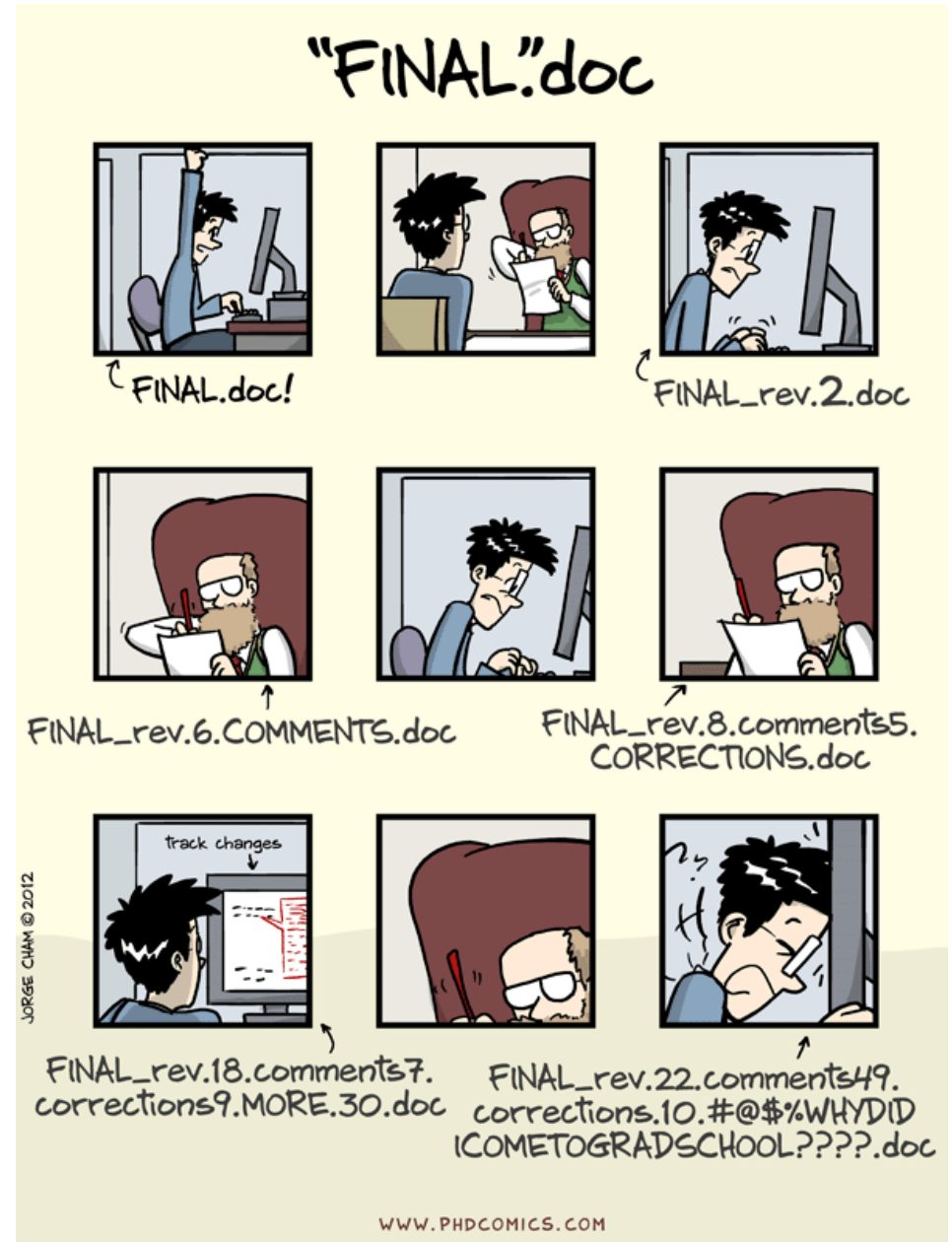


What's GitHub (Other than a Git Repo Host)

- [GitHub Pages](#): static web site host
 - The workshop website is hosted on github,
 - <https://tdmdal.github.io/git-workshop-2022>
- [GitHub Education](#); [GitHub Classroom](#)
 - Organize coding assignments, autograde, etc.
- [Actions](#): automating workflow
- [Codespaces](#): online code editor/developer environment
- [Copilot](#): 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**
 - e.g., code for your paper
- **Collaborate**
 - co-authors (no more emailing code around)
 - open-source community
- **Others...**
 - e.g., personal/project website, and blogs on GitHub, i.e., online presence, "[I web, therefore I am a spiderman.](#)"

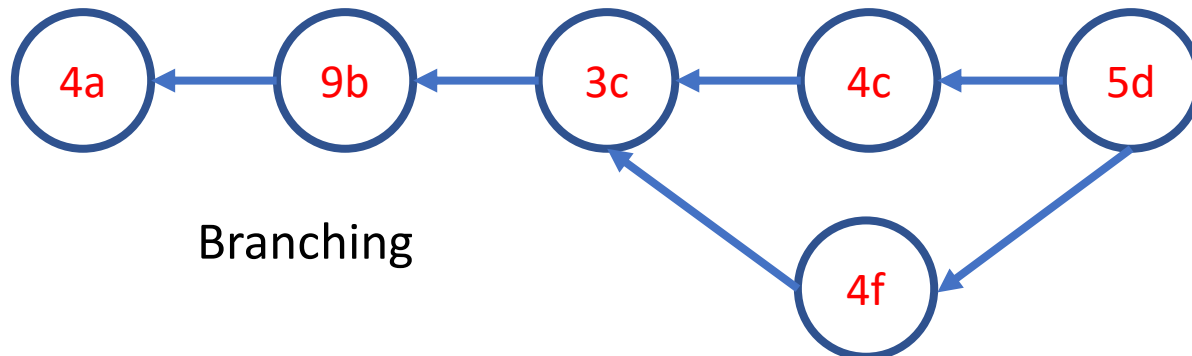


Using Git: GUI Clients vs Command Line

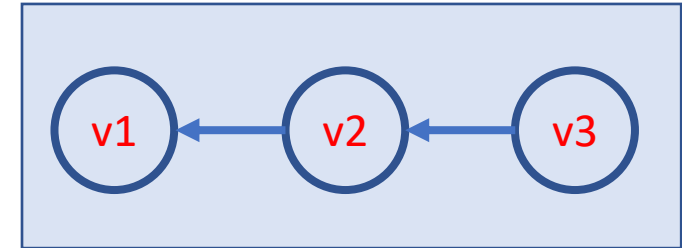
- GUI is easy to get started
 - Today, we will focus on a GUI client, [GitHub Desktop](#)
 - Briefly discuss the underlying concept & command associated with each GUI operation
 - Note that many code editors comes with Git integration too (semi-GUI)
 - e.g., [RStudio](#), [VSCode](#), etc.
- Command 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 Today

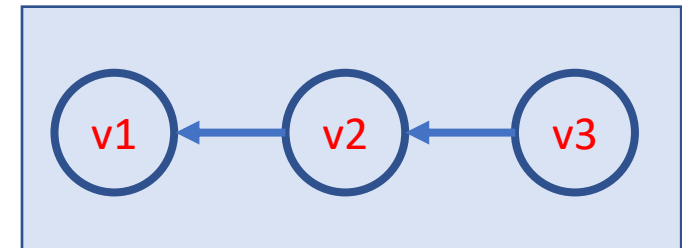
- Focus on a simple linear workflow (demo)
 - manage version history in local repo
 - push local repo to GitHub
- Intro to
 - a simple branching workflow
 - a simple collaboration workflow via GitHub



Remote Repo




Local Repo



push

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 main editor area on the right. Red numbers 1 through 13 are placed over various UI elements to identify them.

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: Main VS Code Interface

- 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 (Publish this repository to GitHub)
- Text: 'No local changes' and 'There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.'
- Section: 'Publish your repository to GitHub' with a 'Publish repository' button.
- Section: 'Open the repository in your external editor' with a button 'Open in Visual Studio Code'.
- Section: 'View the files of your repository in Explorer' with a button 'Show in Explorer'.
- Status bar: 'Committed just now', 'Initial commit', and 'Undo' button.

Stage and Commit

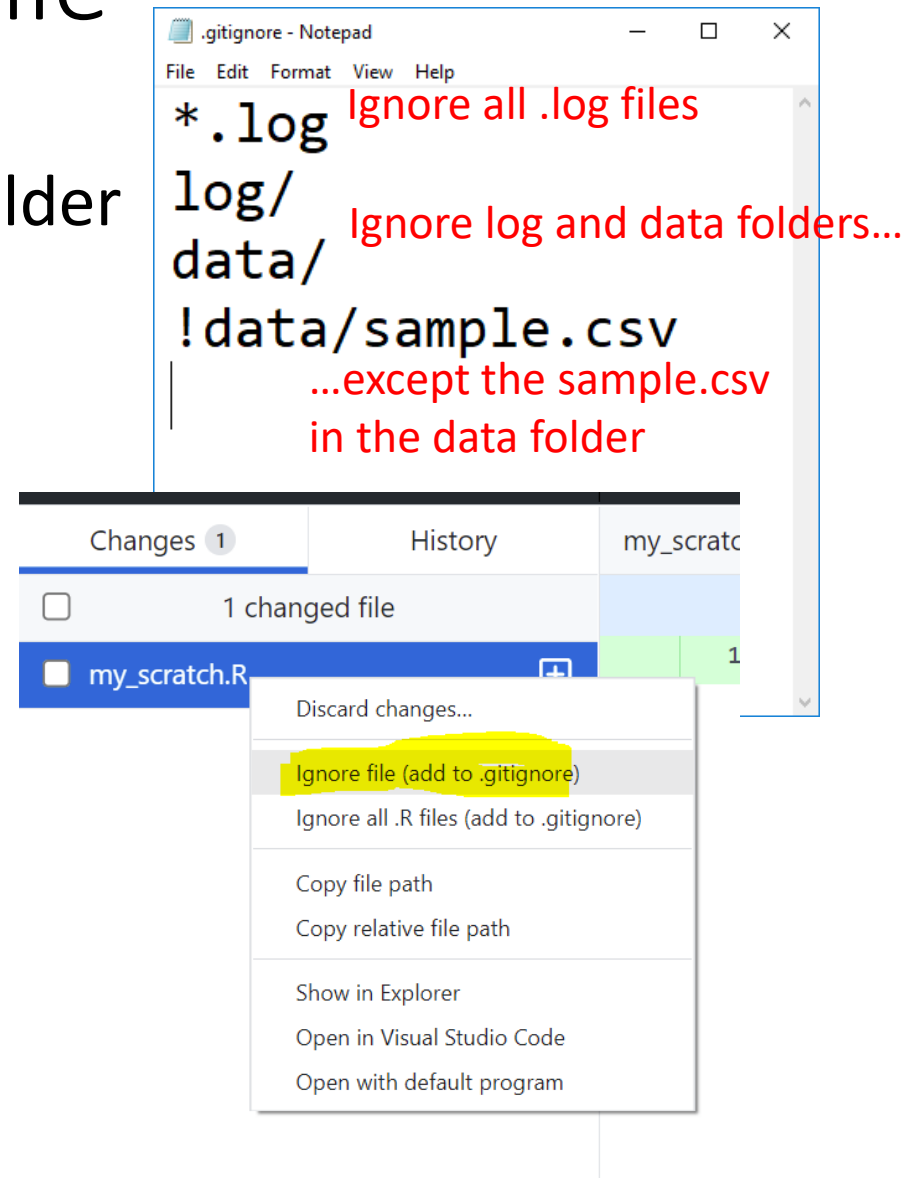
The screenshot displays the Git GUI interface with the following components:

- Menu Bar:** File, Edit, View, Repository, Branch, Help.
- Repository Information:**
 - Current repository: **r-demo-proj**
 - Current branch: **main**
 - Push origin: Last fetched 7 minutes ago (4 ↑)
- Changes Panel:**
 - 2 changed files
 - ☒ test.R
 - ☒ test04.R
- Diff View (test.R):**

Line	Original	Modified
4	4	z <- 3
5	5	
6	6	a <- 4
7		+b <- a
- Commit Summary:**
 - Summary (required): [Highlighted]
 - Description: [Empty text area]
 - Commit to: **main** [Highlighted]
- Commit History:**
 - Committed 16 minutes ago
 - Revert "add b variable and test04"
 - Undo

Suppress Tracking: .gitignore file

- a file named .gitignore in your git repo folder
 - e.g. my_proj/.gitignore
- A collection .gitignore templates
 - <https://github.com/github/gitignore>



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)...

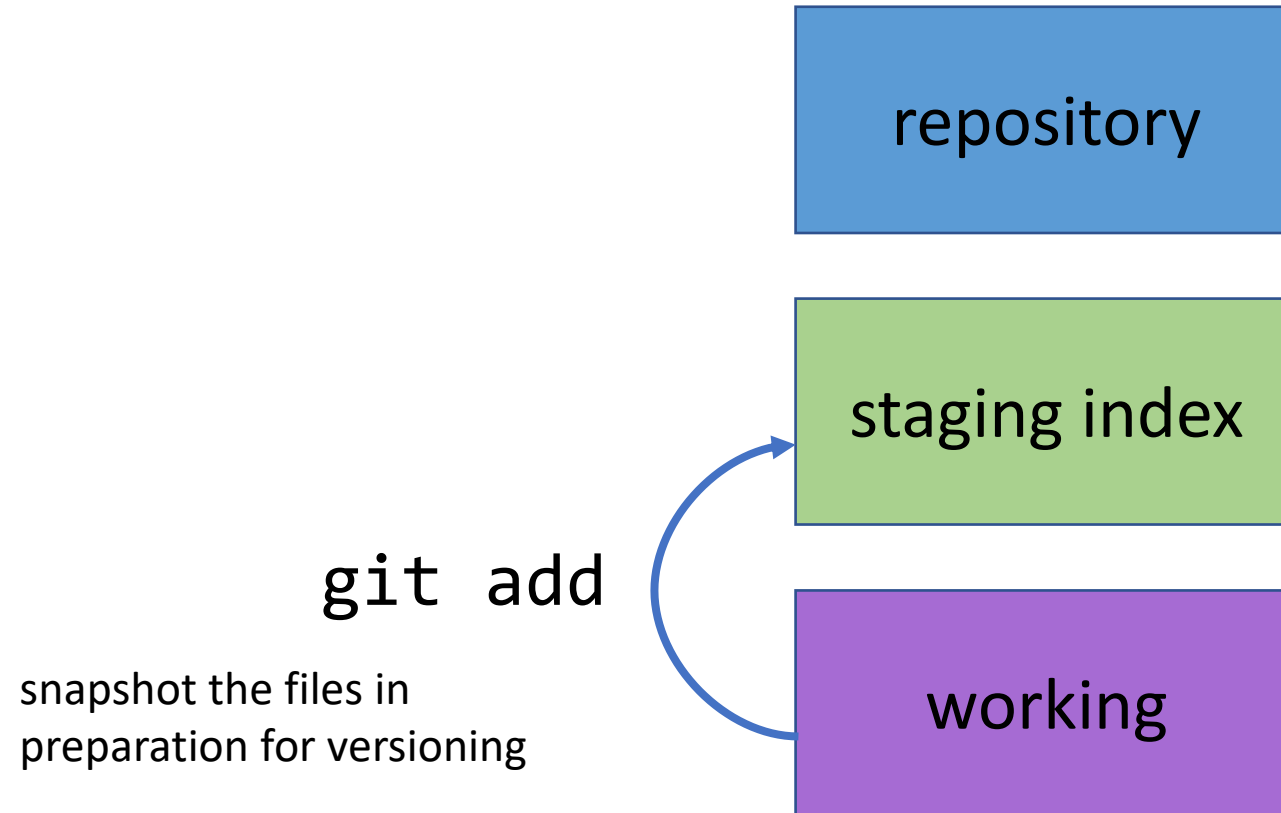
Check commit history: `git log`; `git show`

Compare difference between changes: `git diff`

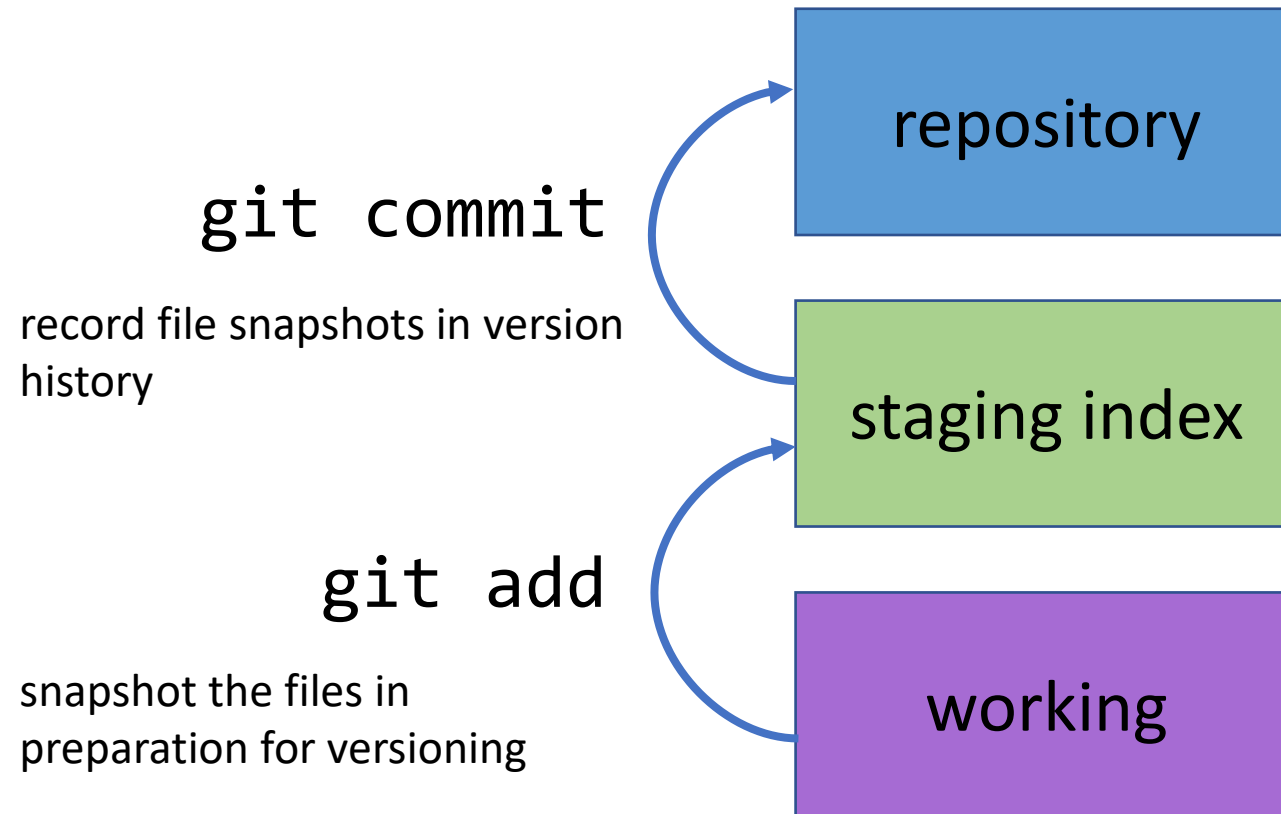
Git Concepts – three trees/areas



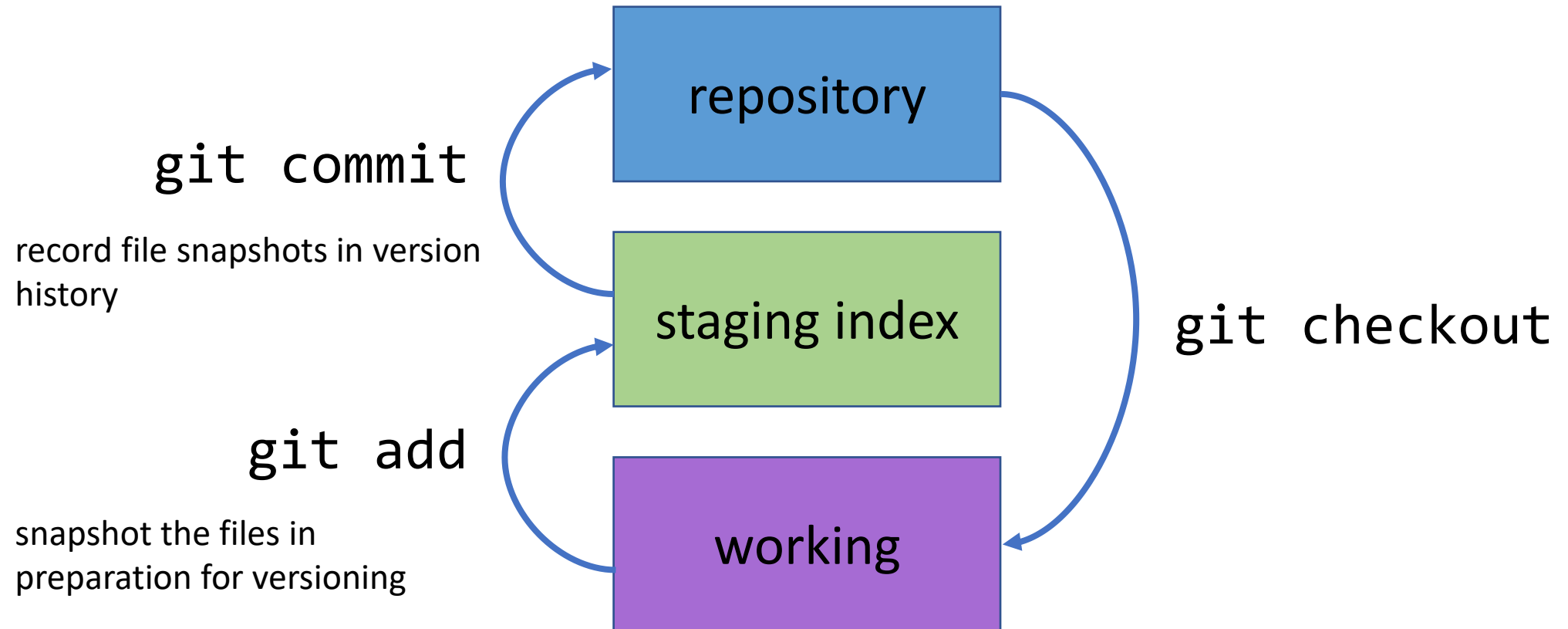
Git Concepts – three trees/areas



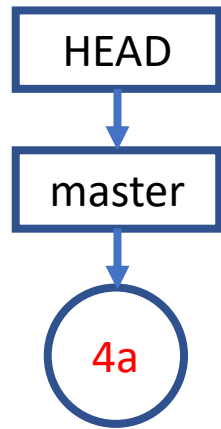
Git Concepts – three trees/areas



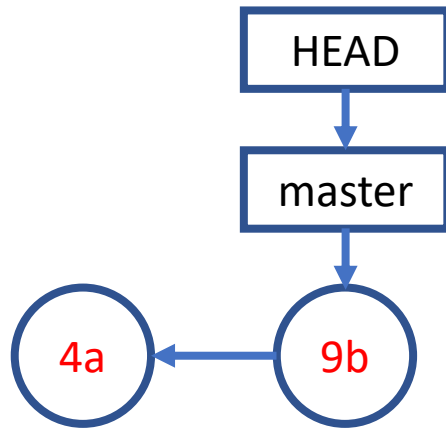
Git Concepts – three trees/areas



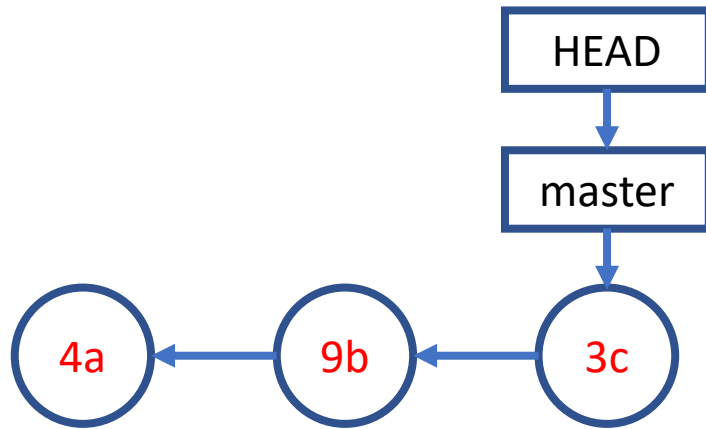
Git Concepts – First commit



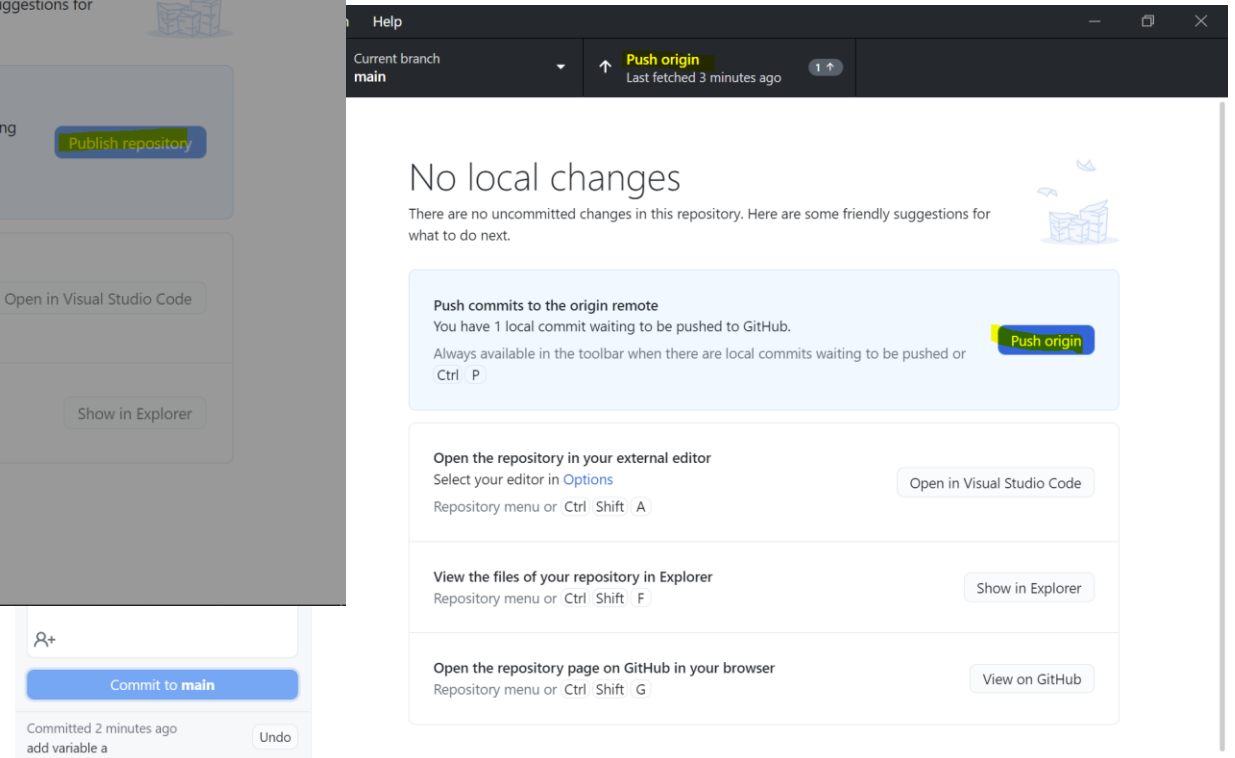
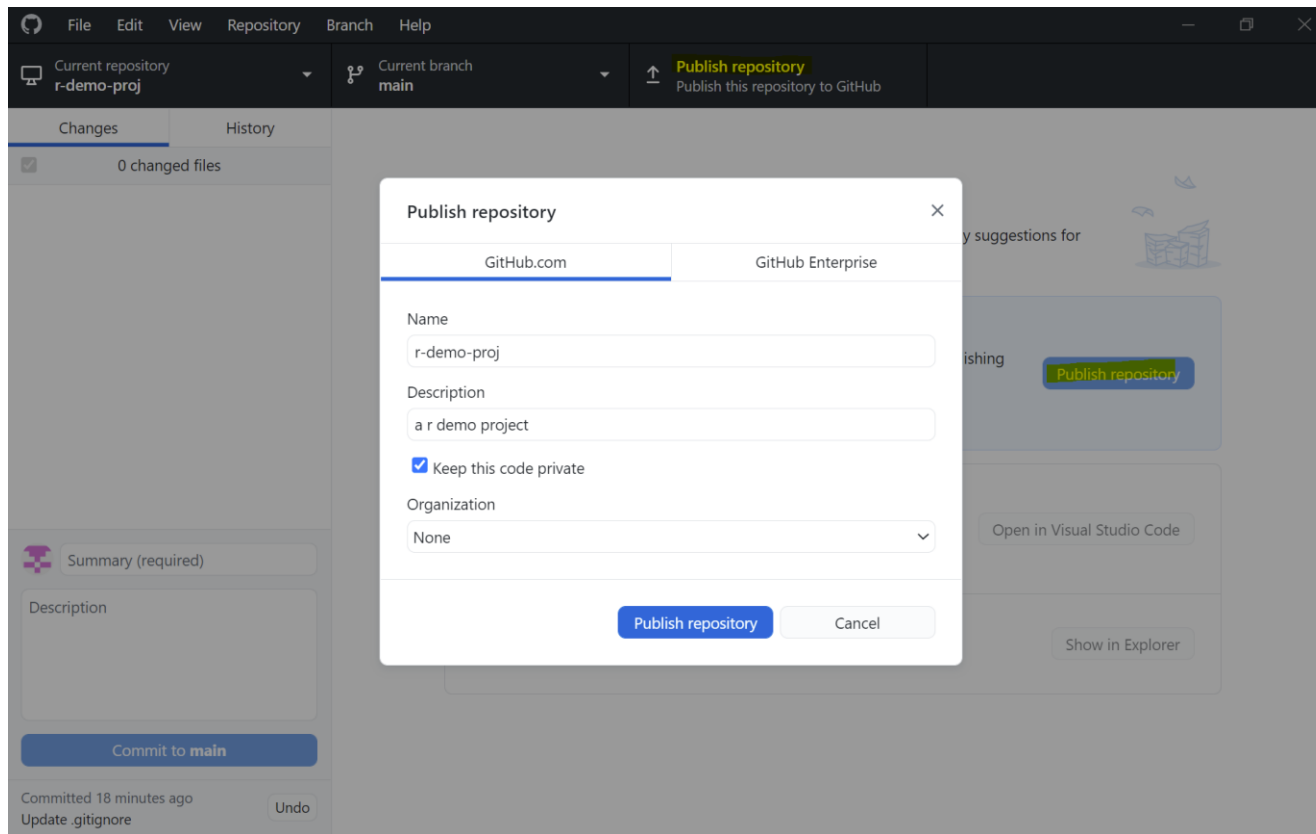
Git Concepts – Second commit



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

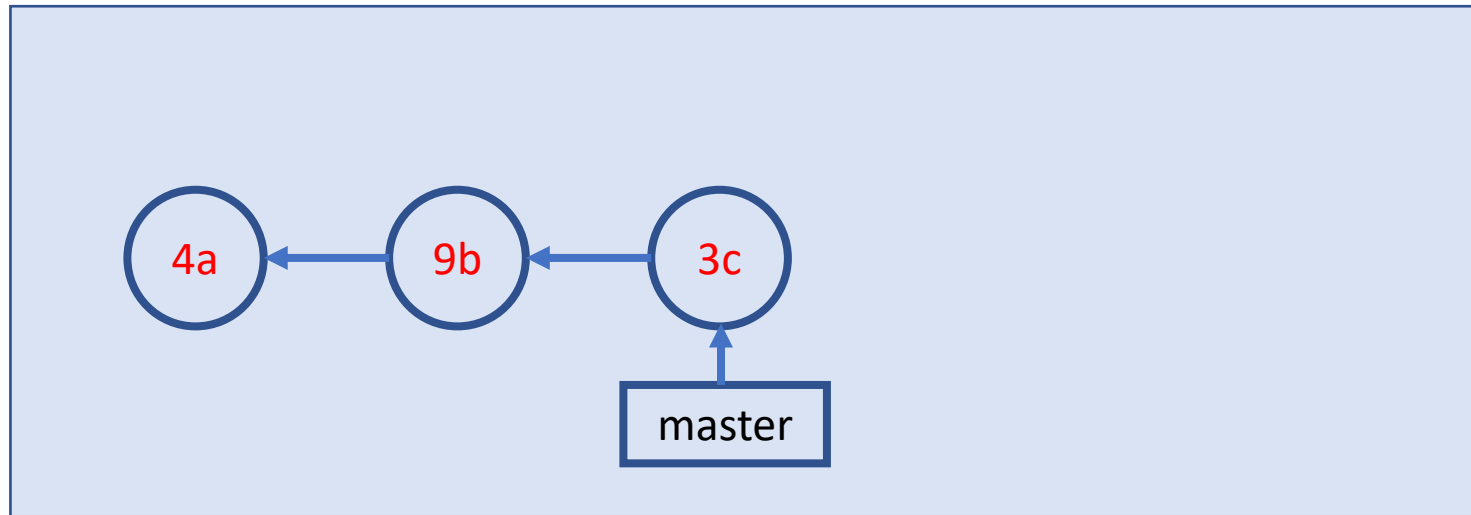
```
git remote add  
git push
```

A Simple Remote Repo Workflow

Remote Repo

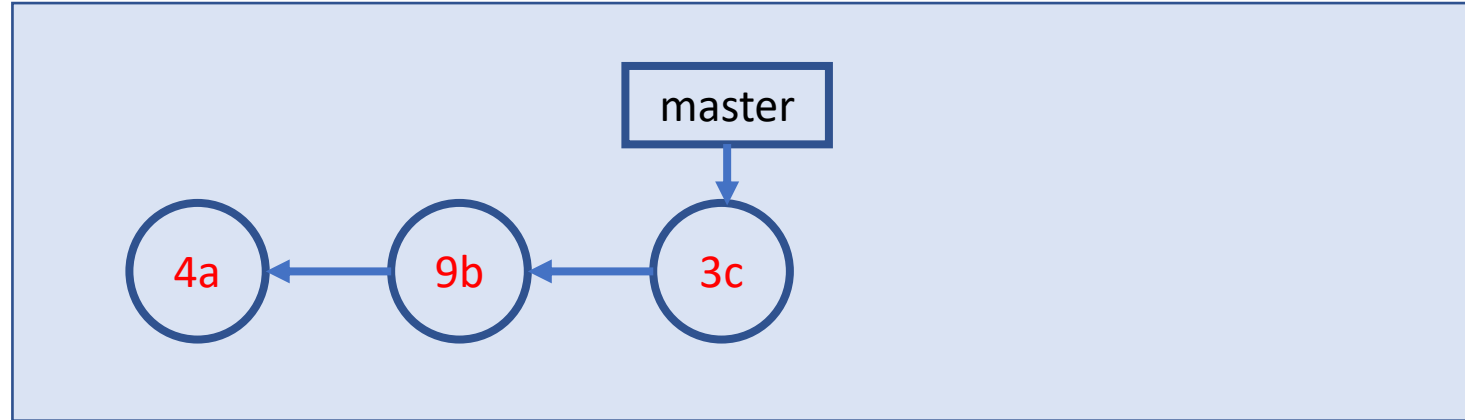


Local Repo

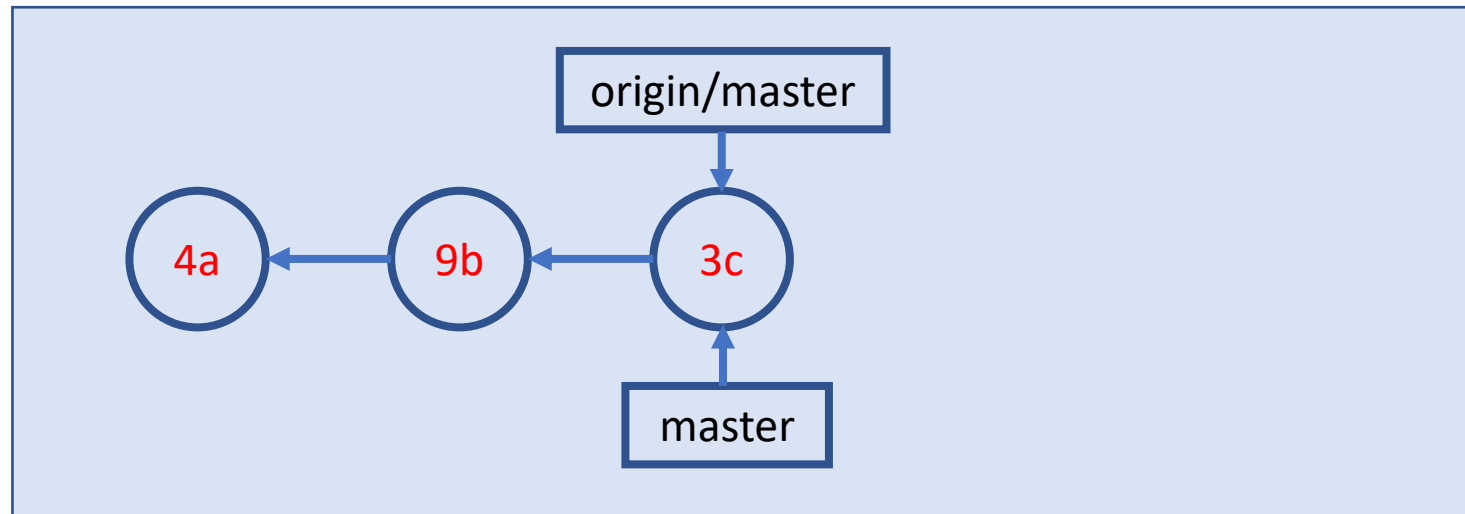


A Simple Remote Repo Workflow `git push`

Remote Repo

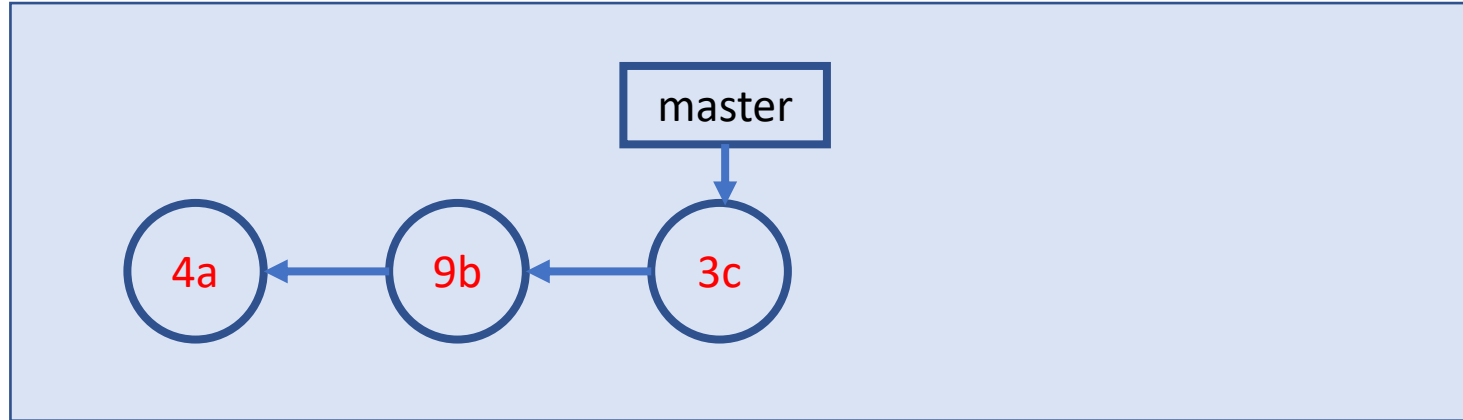


Local Repo

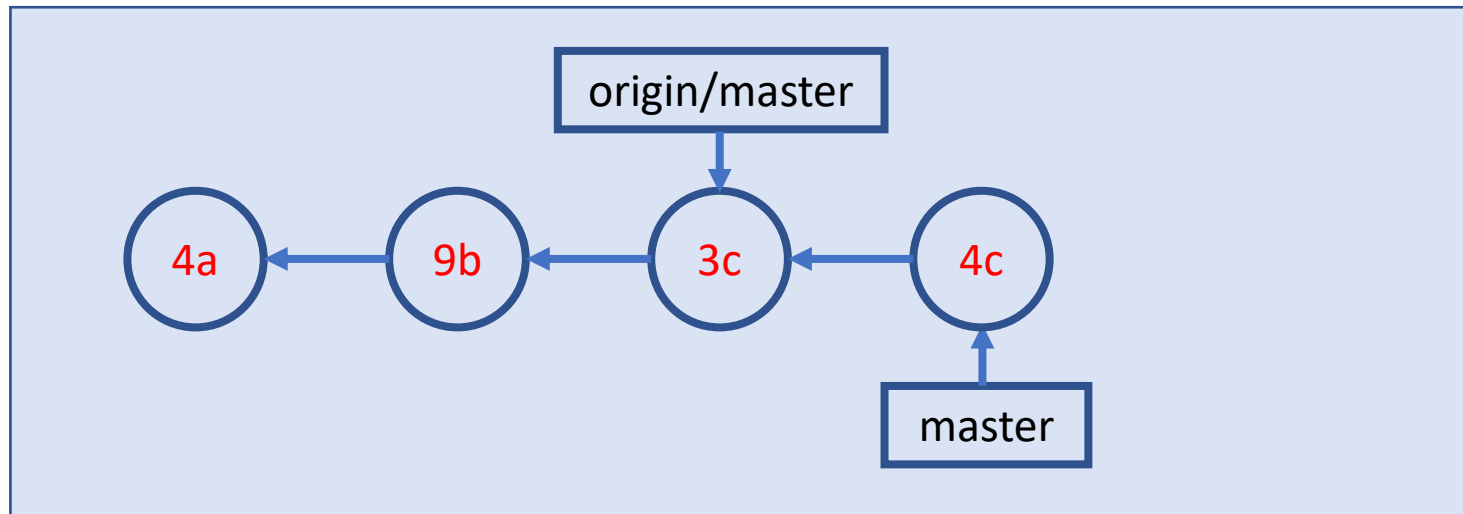


A Simple Remote Repo Workflow

Remote Repo

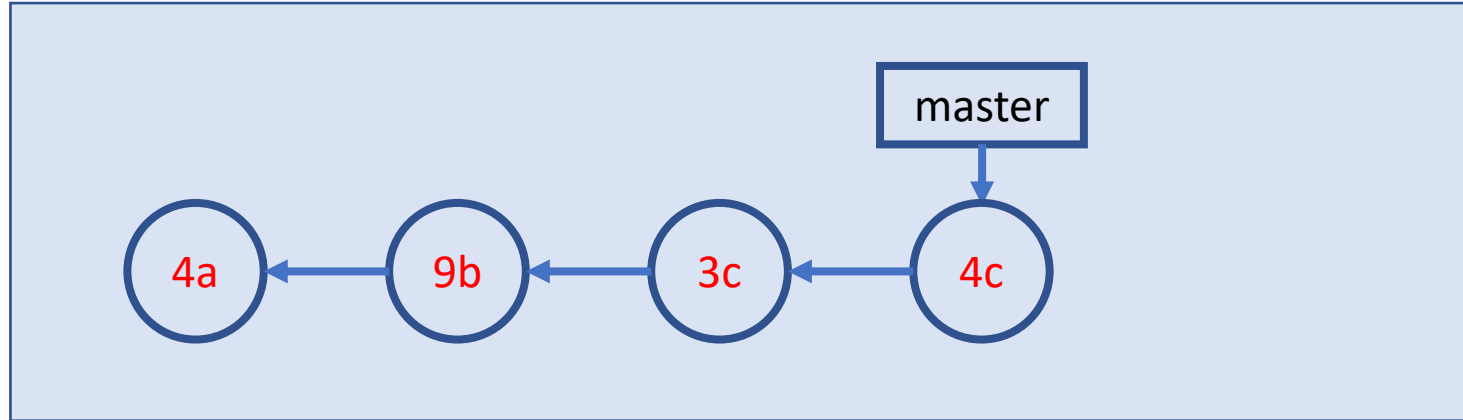


Local Repo

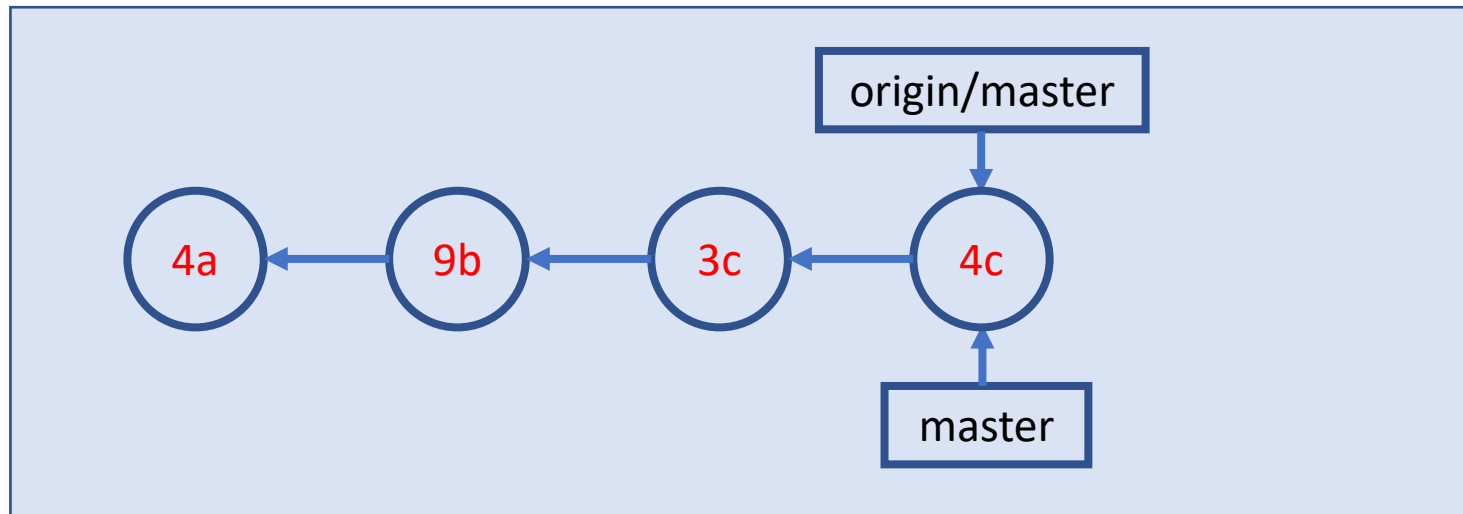


A Simple Remote Repo Workflow `git push`

Remote Repo




Local Repo

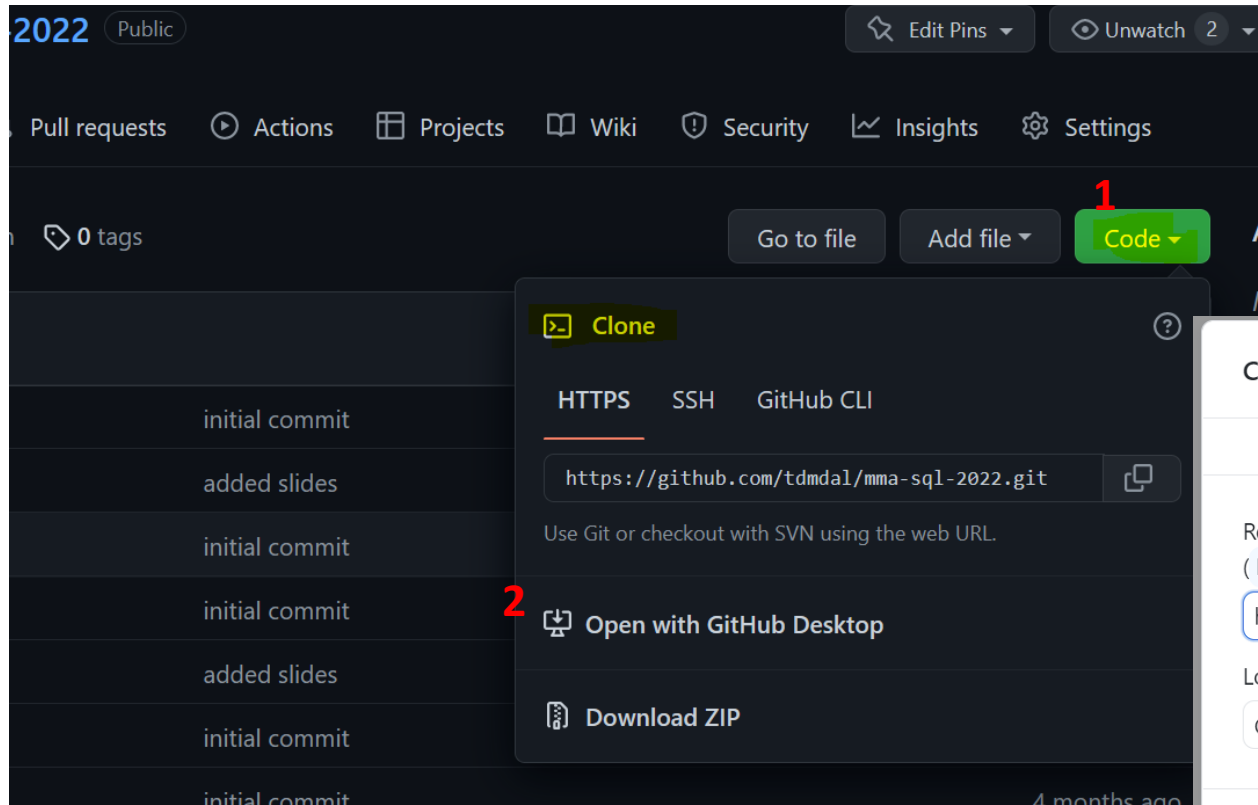


Amend, Undo, Revert, Remove & Rename

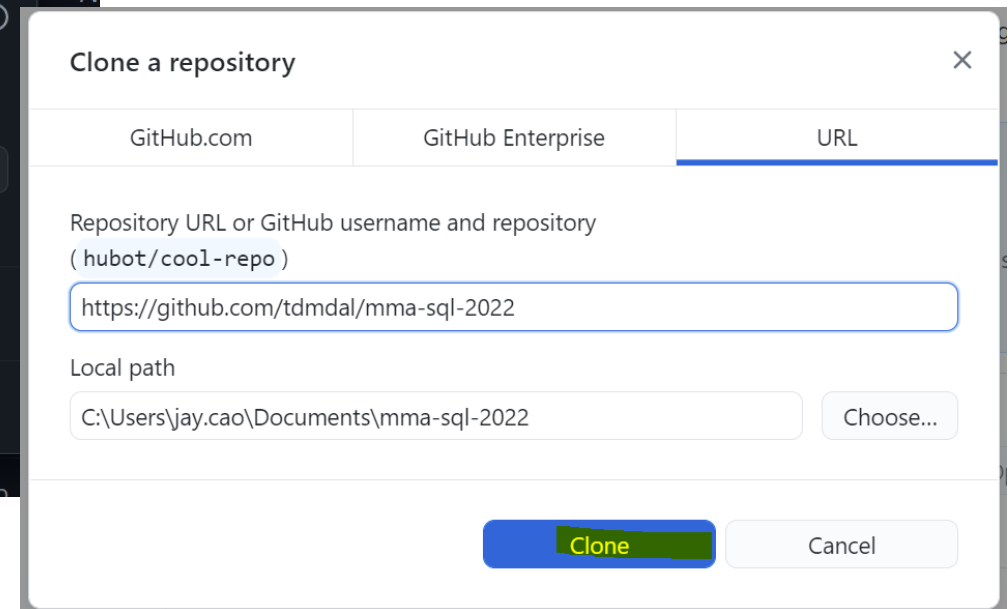
- Amend the last commit: change commit message or add new files
 - disabled by GitHub Desktop if the commit is already pushed
- Undo the last commit: “uncommit” the last commit
- Revert a previous commit: revert the code change and commit it
 - May need to resolve conflict
- Remove or Rename a file

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

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)

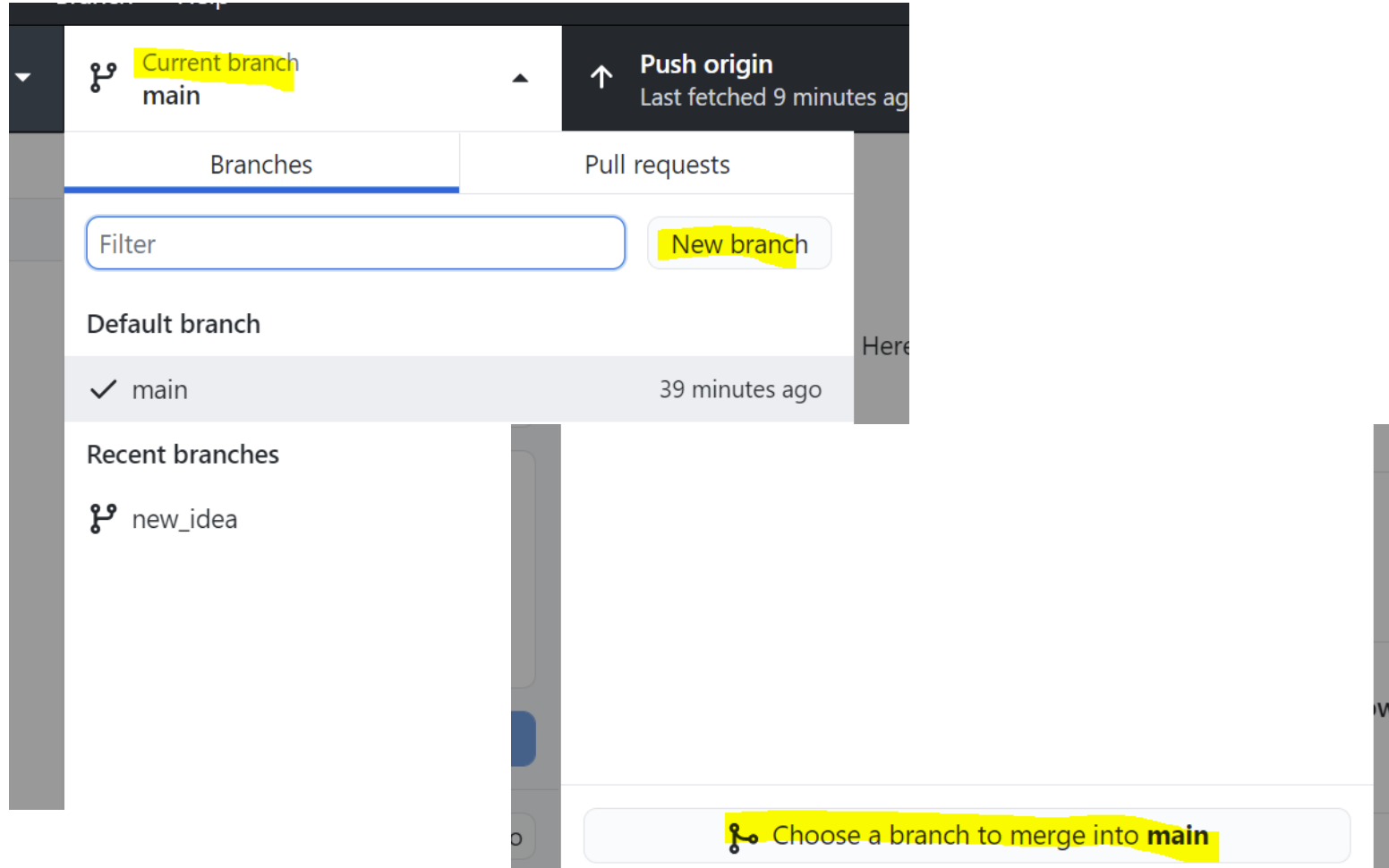
- Git concept / command
 - branch & remote branch
 - merge conflict
 - git reset
 - git stash, rebase, bisect
 - ...
- Git best practice
 - workflows
 - commit size / message
 - ...

Resources

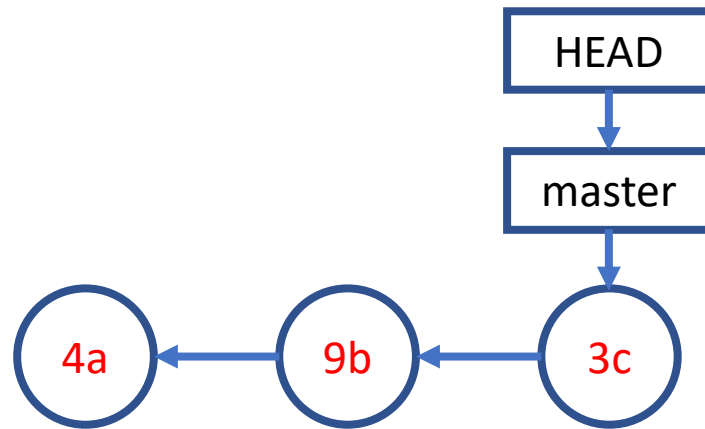
- Git/GitHub with GitHub Desktop
 - [Youtube Video](#) by Coder Coder (22mins; great review for today's workshop)
- Git Command Line Tutorials
 - [Version Control with Git](#) by Software Carpentry
 - [Git Essential Training](#) by Kevin Skoglund at LinkedIn Learning
 - Faculty and staff login from [here](#) 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

Branching (demo)

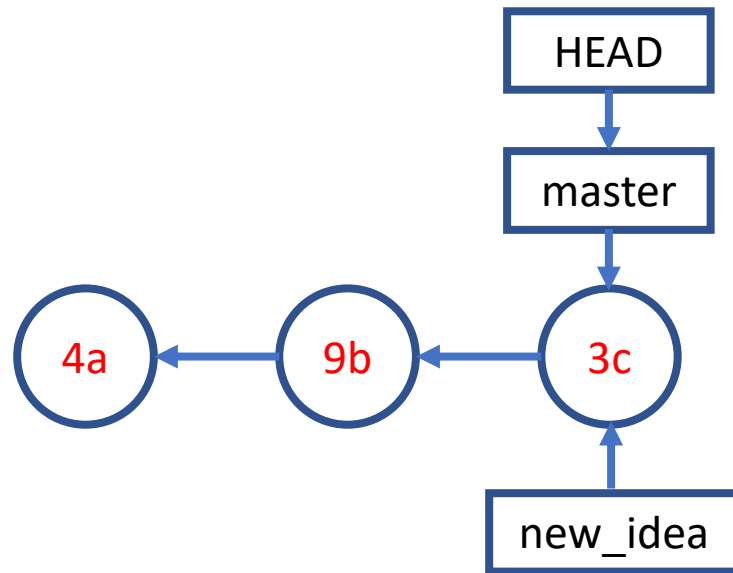


A Simple Branching Workflow



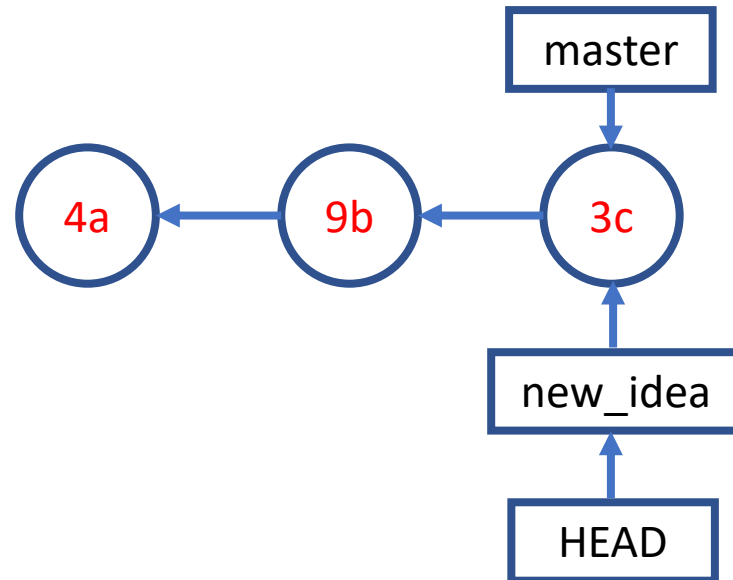
A Simple Branching Workflow

```
git branch new_idea
```



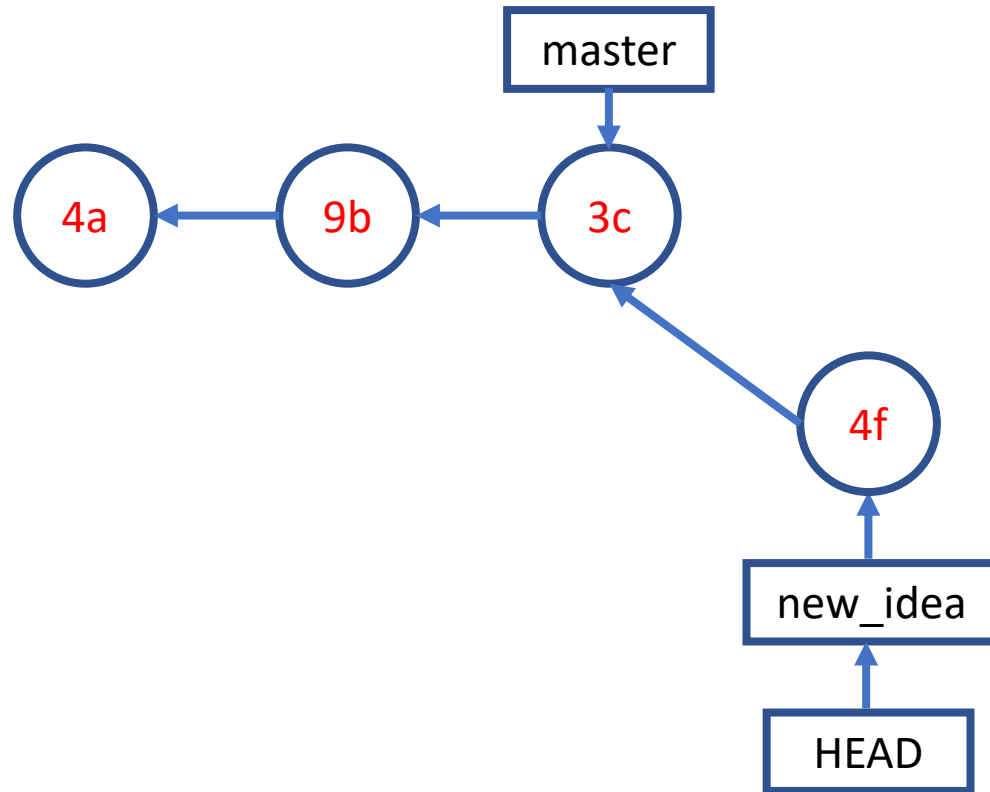
A Simple Branching Workflow

```
git checkout new_idea
```



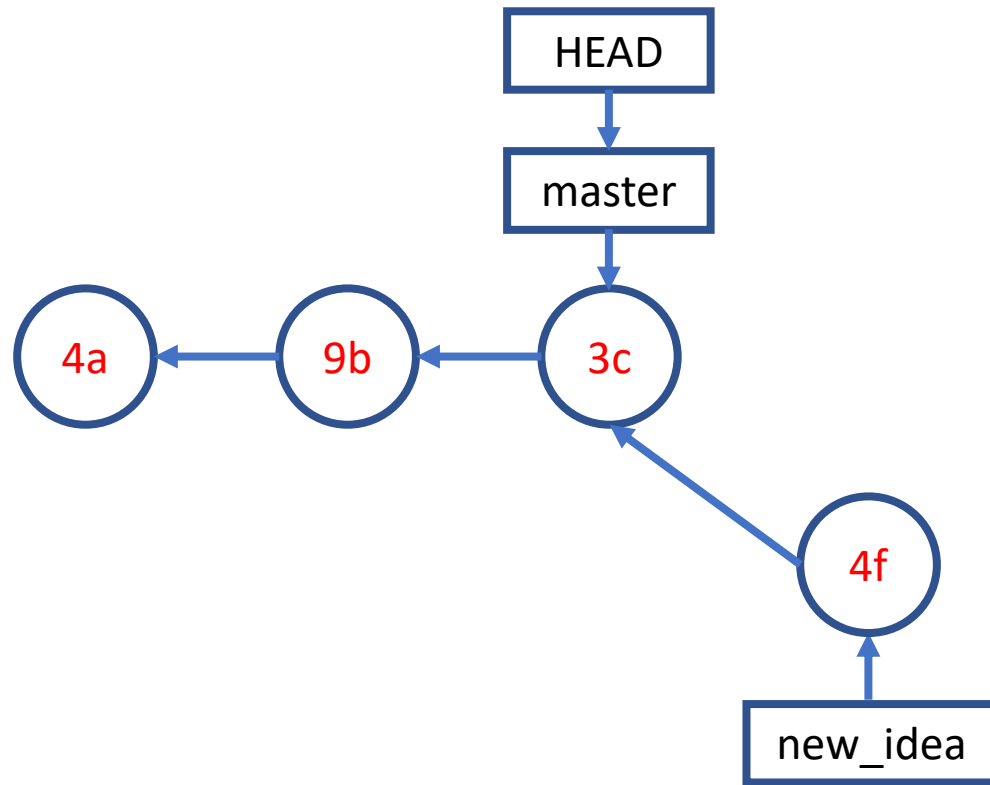
A Simple Branching Workflow

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



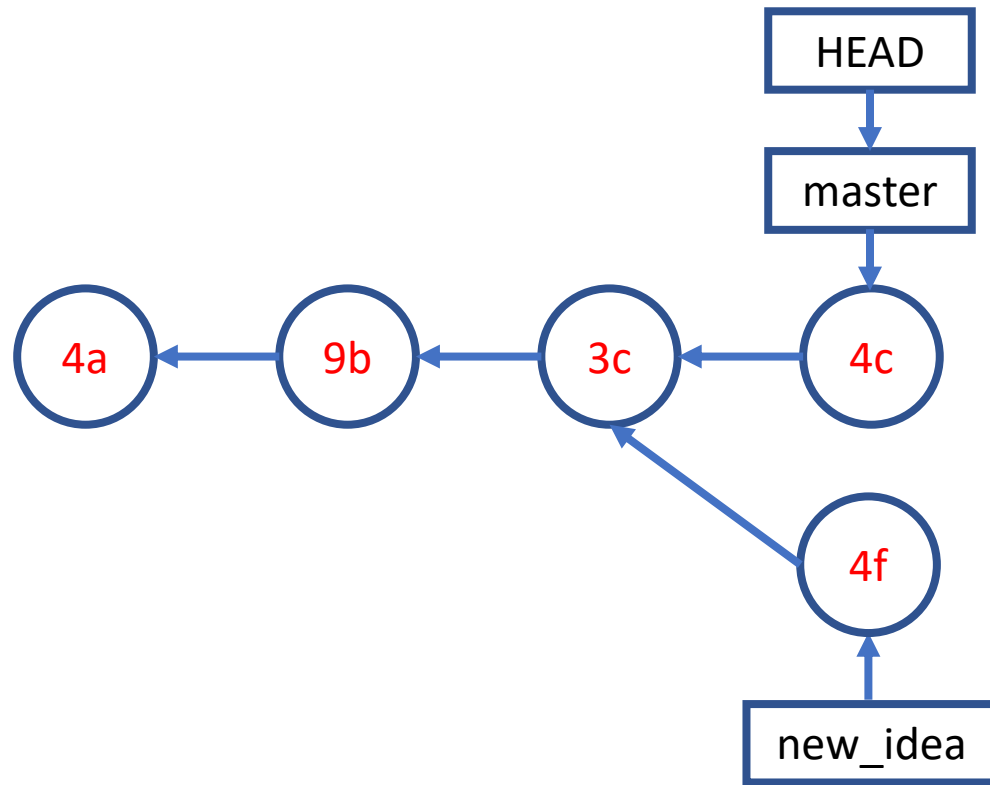
A Simple Branching Workflow

`git checkout master`



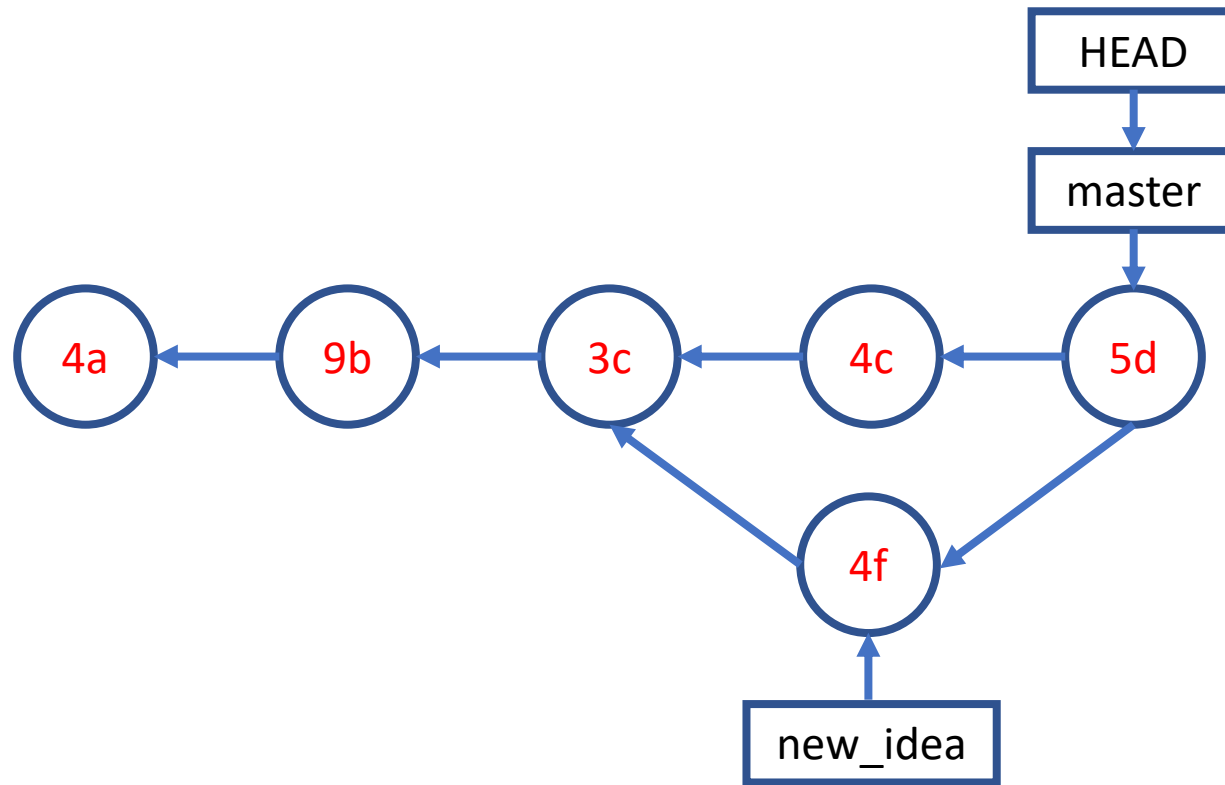
A Simple Branching Workflow

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



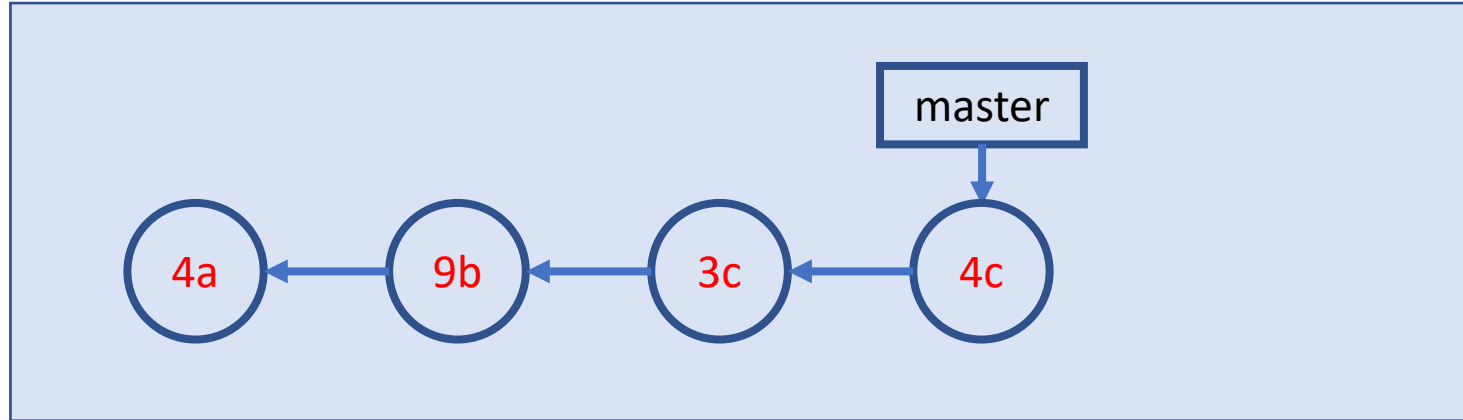
A Simple Branching Workflow

```
git merge new_idea
```

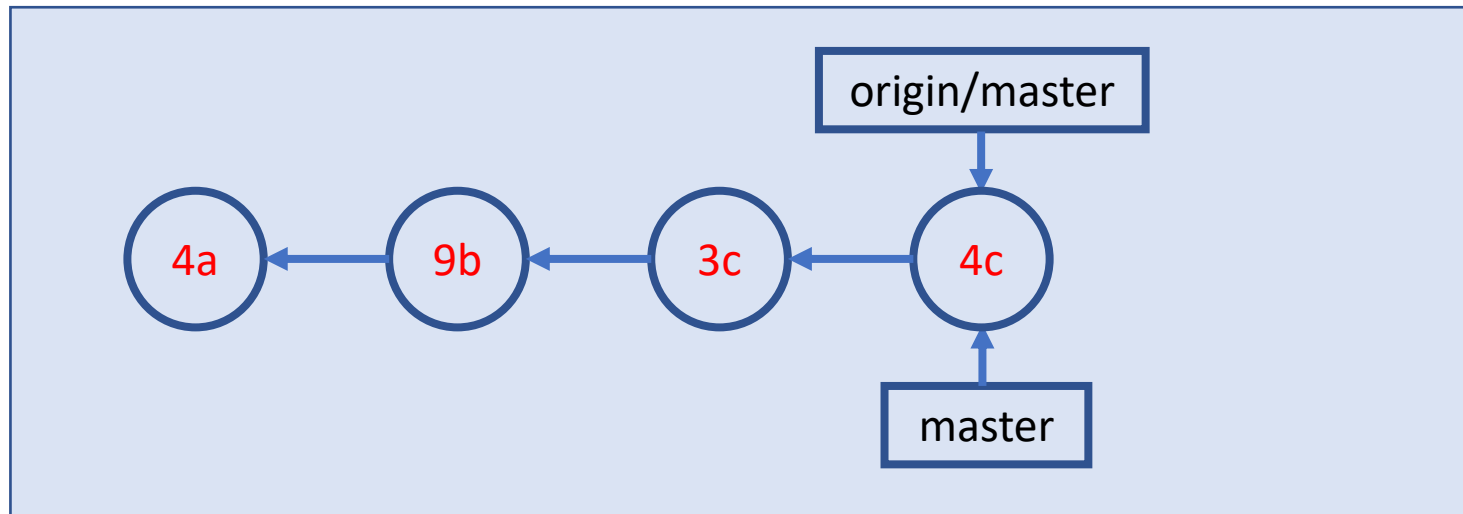


A Simple Collaboration Workflow

Remote Repo

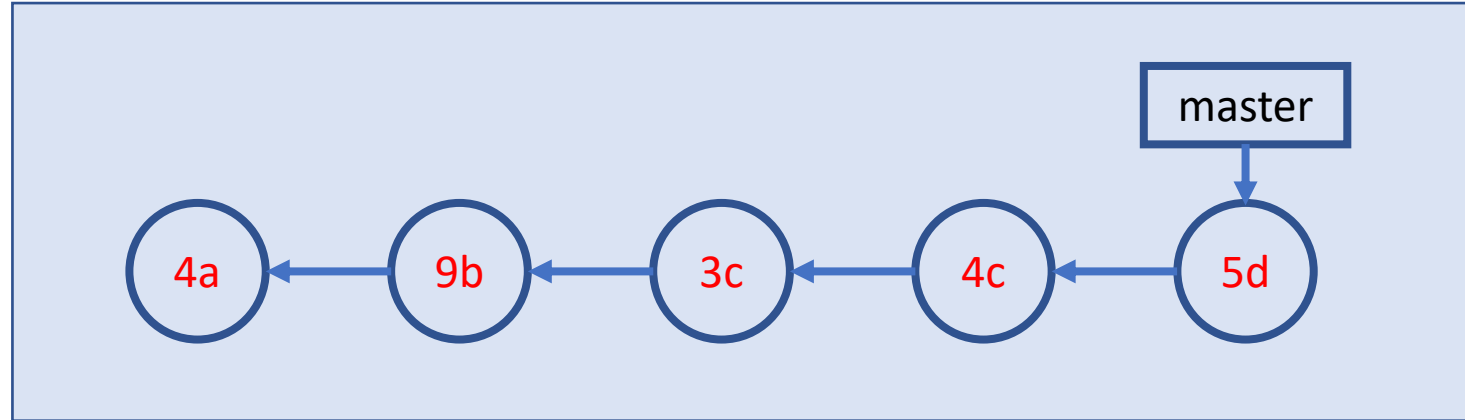


Local Repo

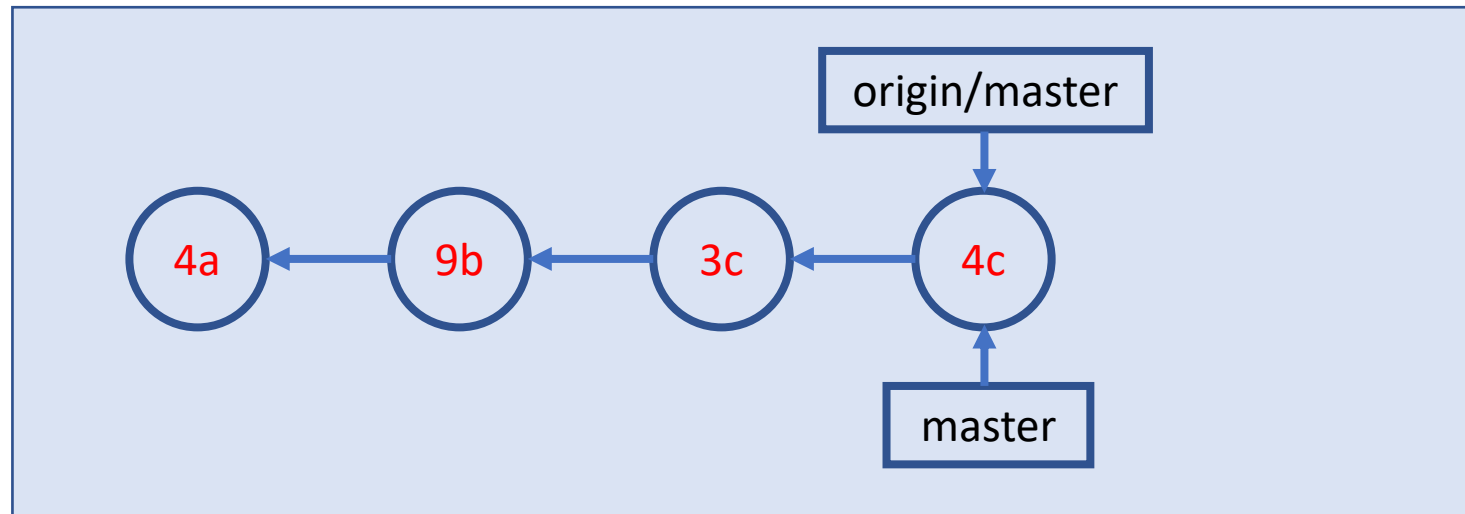


A Simple Collaboration Workflow

Remote Repo



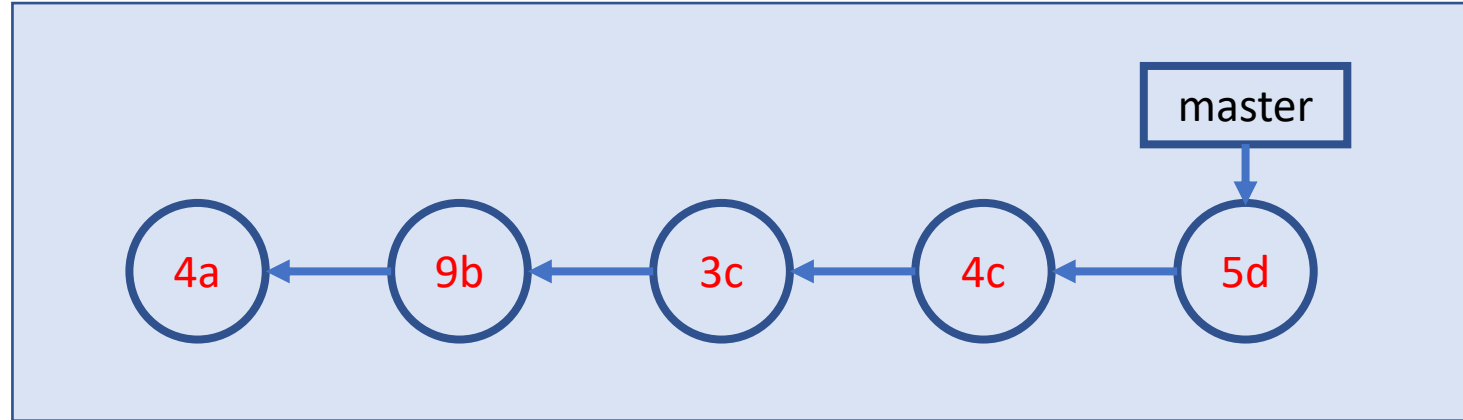
Local Repo



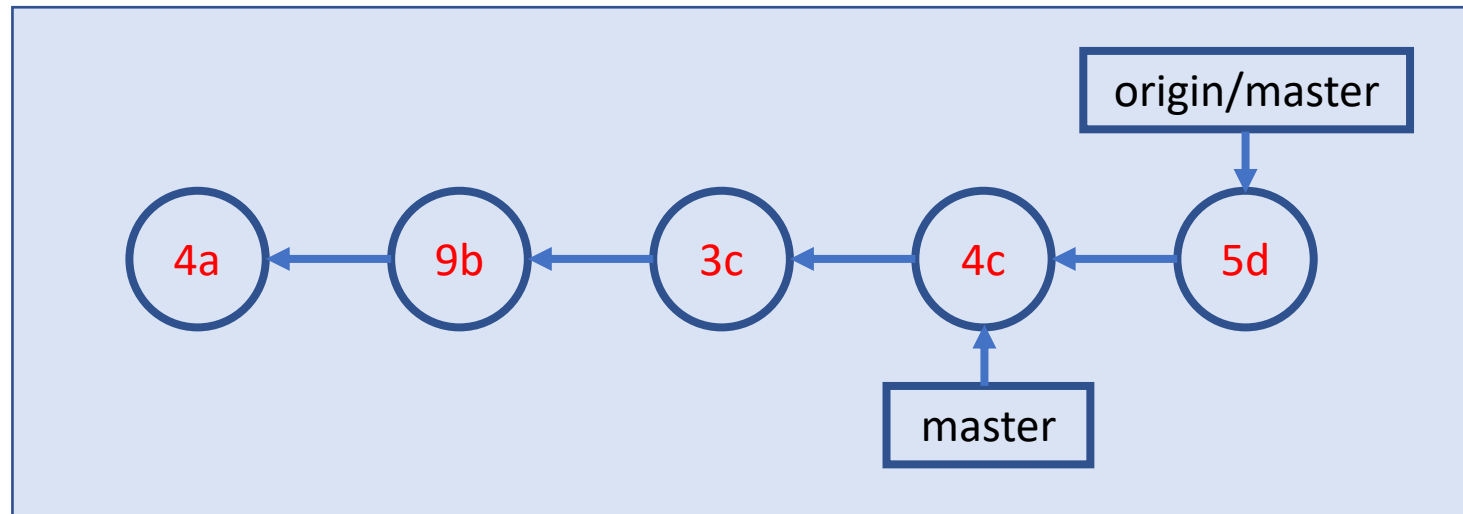
A Simple Collaboration Workflow

git fetch

Remote Repo

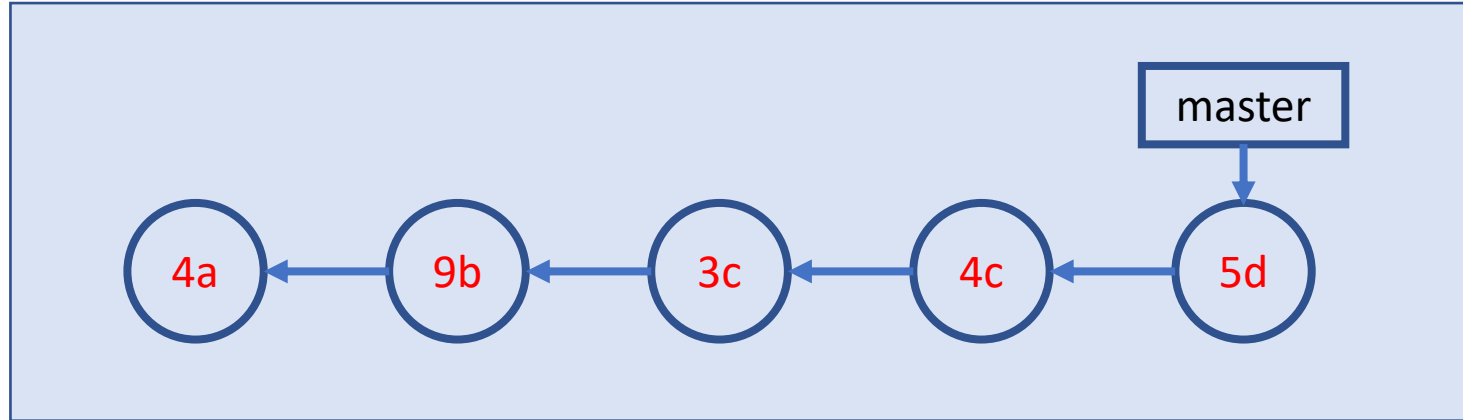


Local Repo

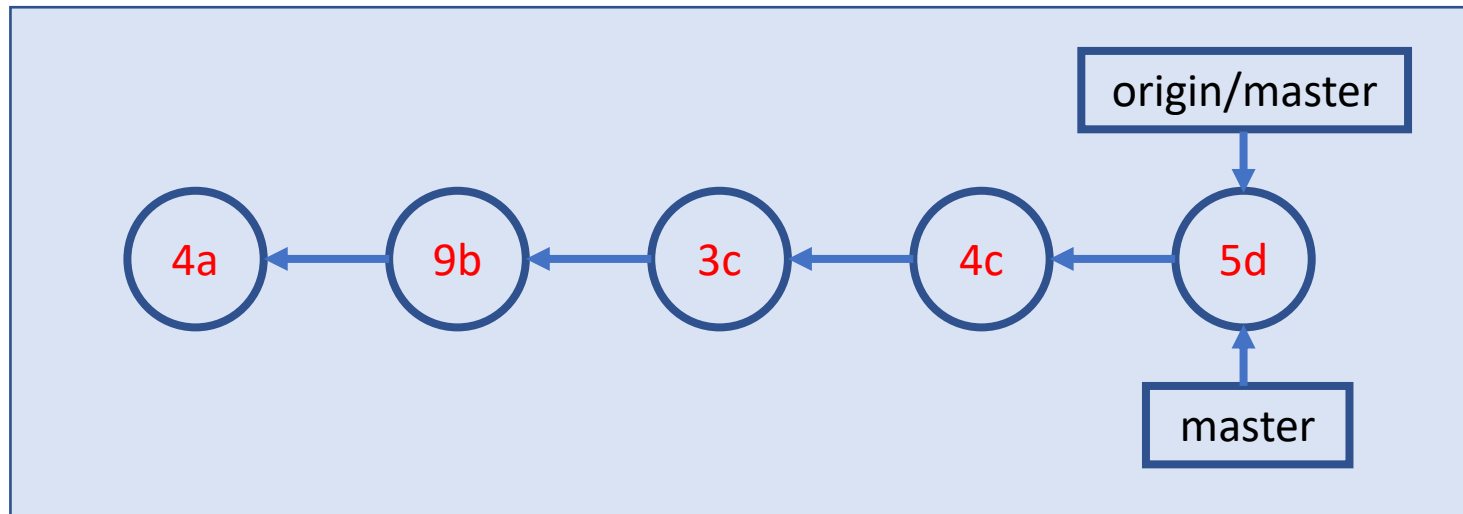


A Simple Collaboration Workflow `git merge`

Remote Repo



Local Repo



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

Appendix - “undo” ops in command line

Remove and Rename Files (FYR)

- Remove files

```
git rm <file>
```

- Rename files

```
git mv <file_old> <file_new>
```

- After removing or rename files

```
git commit -m "<remove or rename msg>"
```

Undo (1 / FYR)

- Retrieve old version of a file (to staging index & working dir)

```
git checkout <commit-id> -- <file>
```

- Undo working directory changes

```
git checkout -- <file>
```

- Unstaging files

```
git reset HEAD <file>
```

Undo (2 / FYR)

- Amending last commit

```
git commit -amend -m "commit message"
```

- Reverting a commit (by adding a new commit to undo last commit)

```
git revert <commit-id>
```

- Undo multiple commits

```
git reset [--soft|--mixed|--hard] <commit-id>
```