

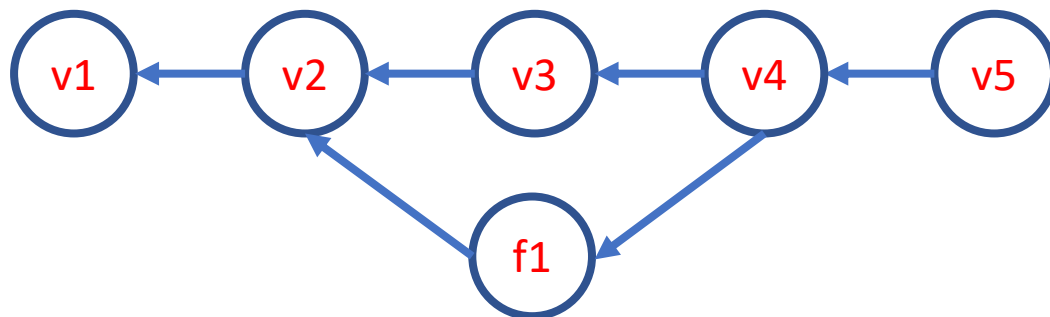
# Intro to Git & GitHub

Jay / [TDMDAL](#)

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

# What's Git git

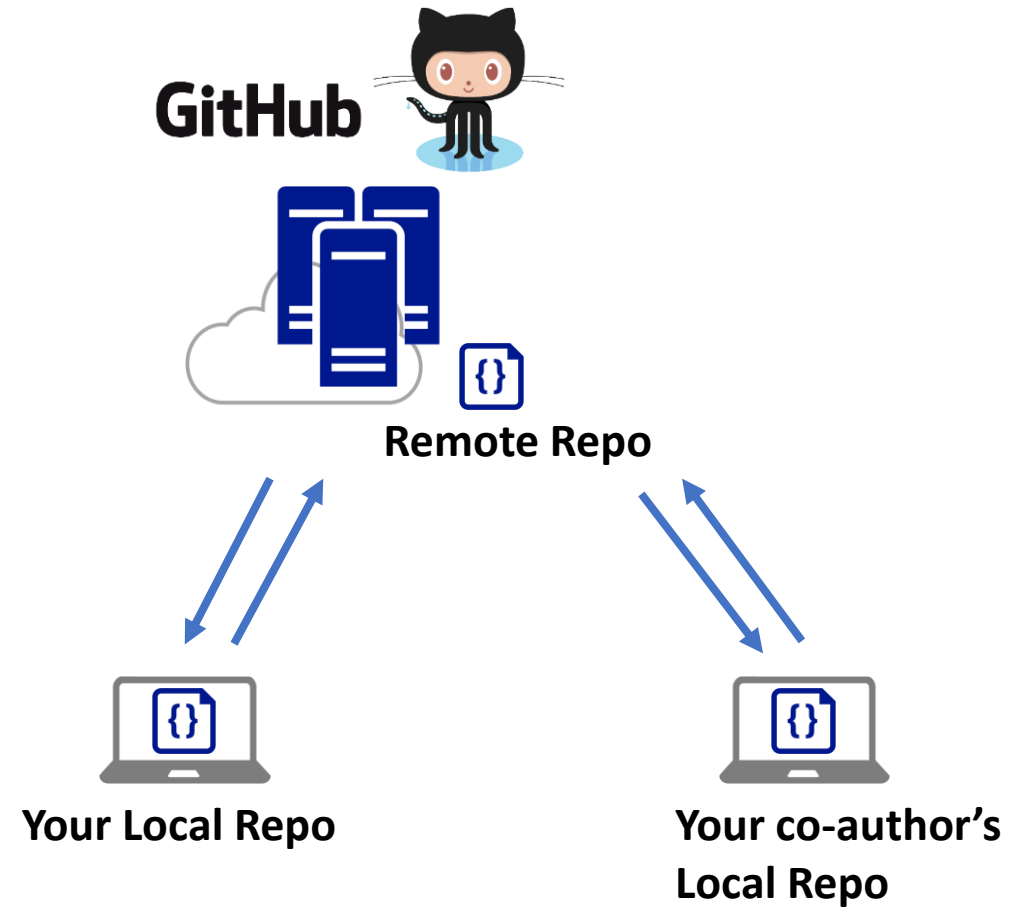
- A version control system
  - manage the evolution of a set of files (repository / repo)
  - usually for source code or text files
- Version control?
  - keep track of changes: version 1, version 2, etc.
  - like “Track Changes” or “undo” in MS Word, but much more powerful



# 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.com), [gitlab](https://gitlab.com), etc.

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



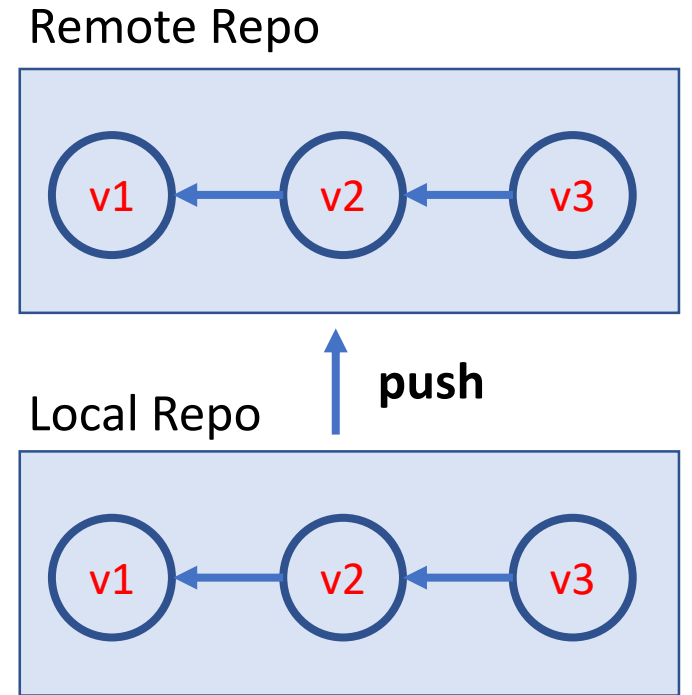
# Why Git & GitHub

- Organize (record keeping; traceability)
  - Track, compare and undo changes
  - Manage multiple versions/ideas at the same time efficiently
  - Backup your work
- Share
- Collaborate
  - co-authors
  - open source community
- Others...



# Plan for Today

- Focus on a simple linear workflow (hands-on)
  - manage version history in local repo
  - push local repo to GitHub
- If time permits, intro to
  - a simple collaboration workflow
  - a simple branching workflow



# Using Git: Command Line vs GUI Clients

- Command line is universal
  - i.e. same commands for Windows, Mac, and Linux
- Easy to go from command line to GUI clients
- Today, we will focus on command line

# Hands-on?

Option 1: Install Git: <https://git-scm.com/downloads>

Option2: Use this in-browser [Linux emulator](#) for Git practice.

- may have problem accessing internet (i.e. when you use github)

# The simplest git workflow (demo)

1. Make changes to your files
2. Snapshot files in preparation for versioning (stage the changes): `git add`
3. Record version history (commit the changes): `git commit`
4. repeat (back to 1)...

Configure git for first-time use: `git config`

Create a new local repo: `git init`

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

Compare difference between changes: `git diff`

<https://tdmdal.github.io/git-workshop/basic-git-workflow.html>



# Git Concepts – three trees/areas

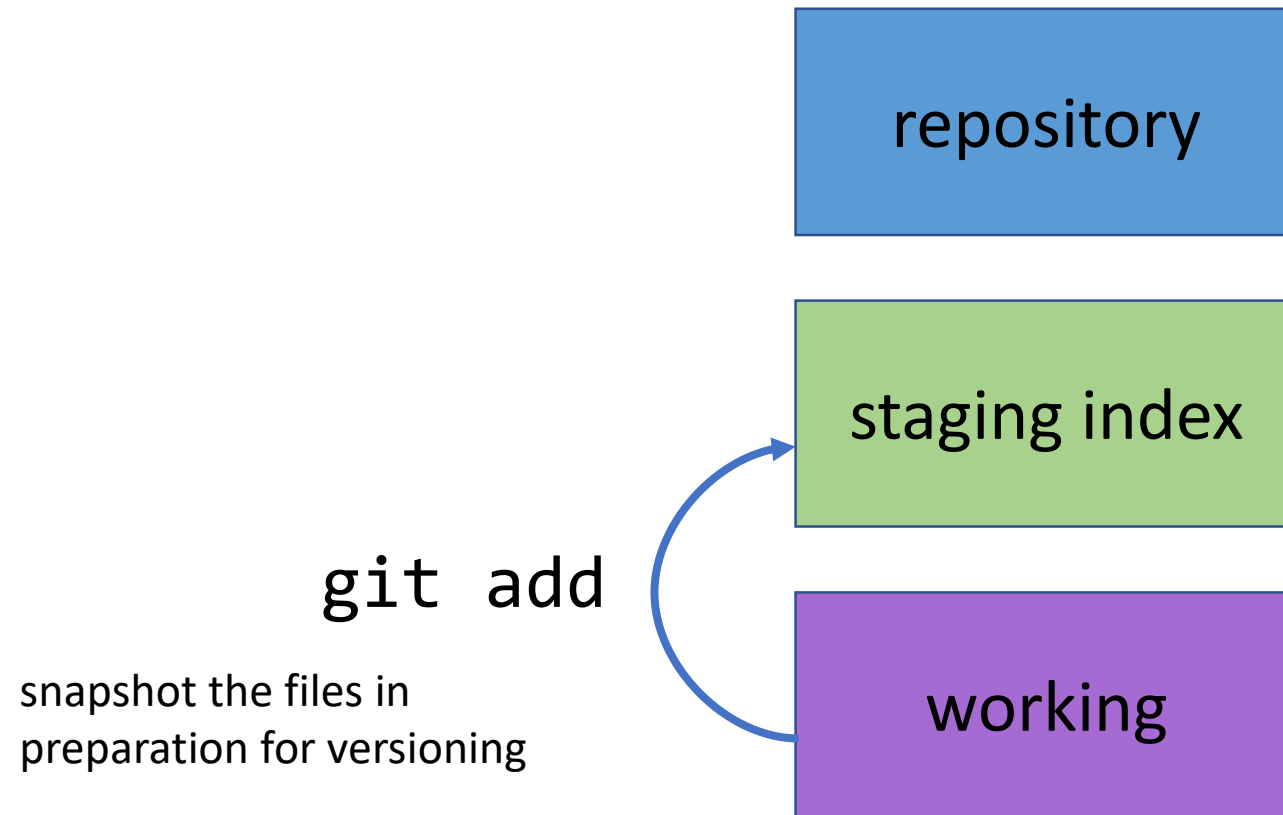


repository

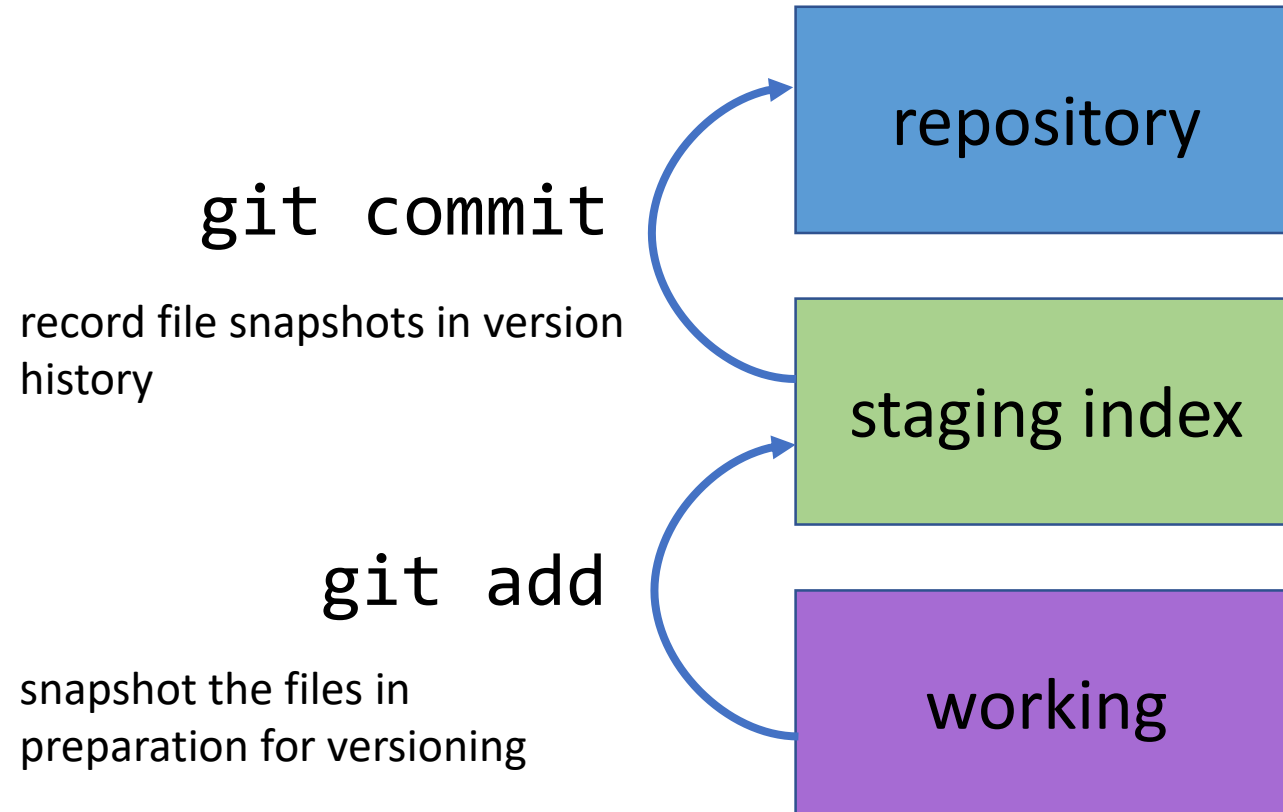
staging index

working

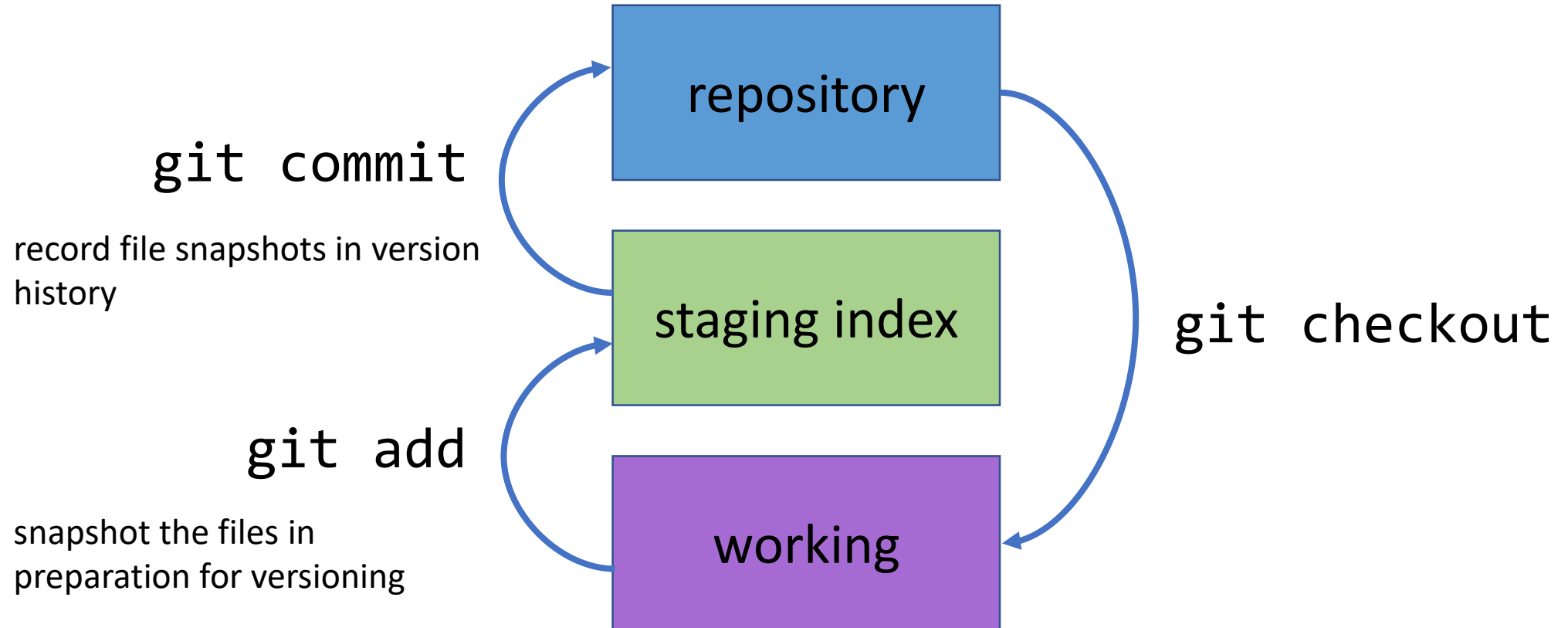
# Git Concepts – three trees/areas



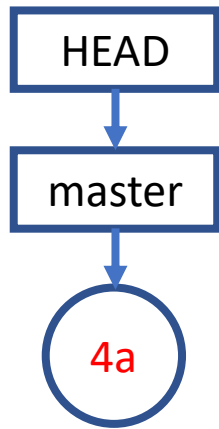
# Git Concepts – three trees/areas



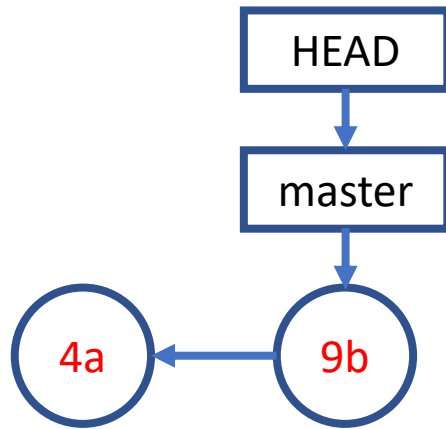
# Git Concepts – three trees/areas



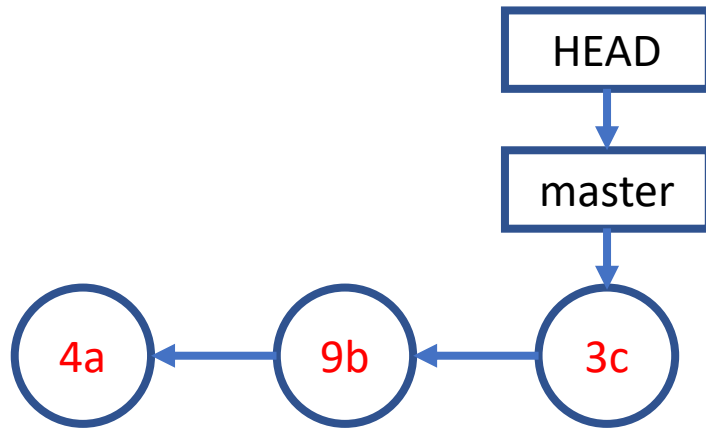
# Git Concepts – First commit



# Git Concepts – Second commit



# Git Concepts – Third commit and so on...



# Remove and Rename Files (FYI)

- Remove files (**demo**)

```
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 / FYI)

- Retrieve old version of a file (to staging index & working dir) (**demo**)  
`git checkout <commit-id> -- <file>`

# Undo (1 / FYI)

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

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

- Undo working directory changes

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

- Unstaging files

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

# Undo (2 / FYI)

- Amending last commit

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

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

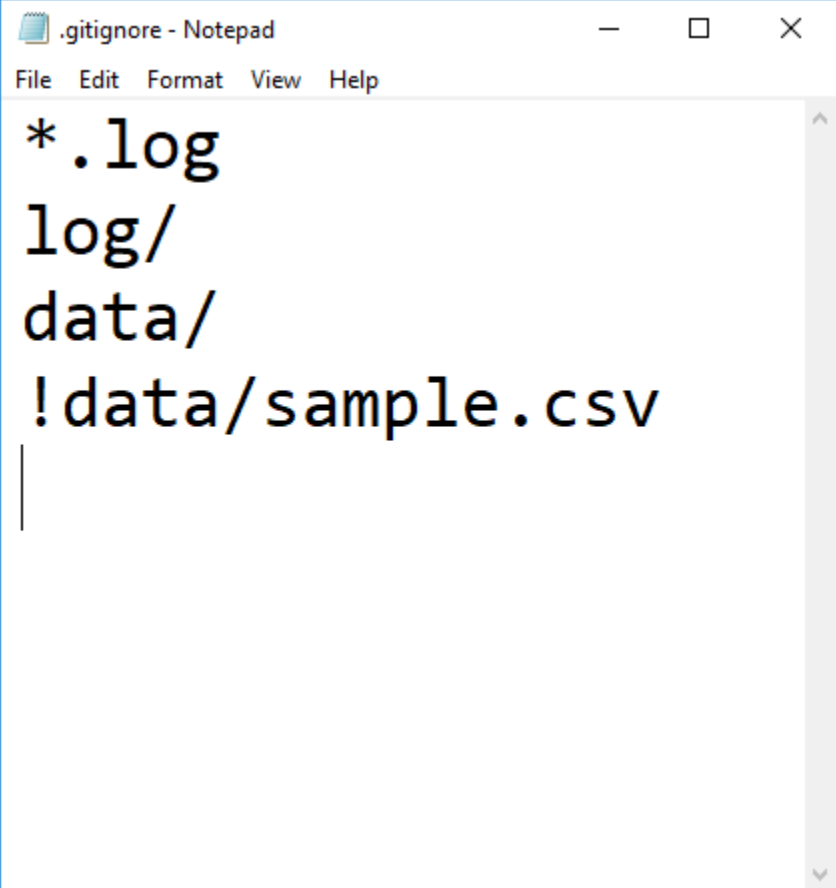
```
git revert <commit-id>
```

- Undo multiple commits

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

# Suppress Tracking: .gitignore file

- Create a file named `.gitignore` in your project folder
  - e.g. `my_proj/.gitignore`



```
.gitignore - Notepad
File Edit Format View Help
*.log
log/
data/
!data/sample.csv
```

# Work with GitHub (demo)

- GitHub Account
- Create a GitHub project repo & push your code there
  - backup
  - collaborate with your co-authors
  - collaborate with open source community
- Use a public repo as your project starting point

```
git remote add  
git push
```

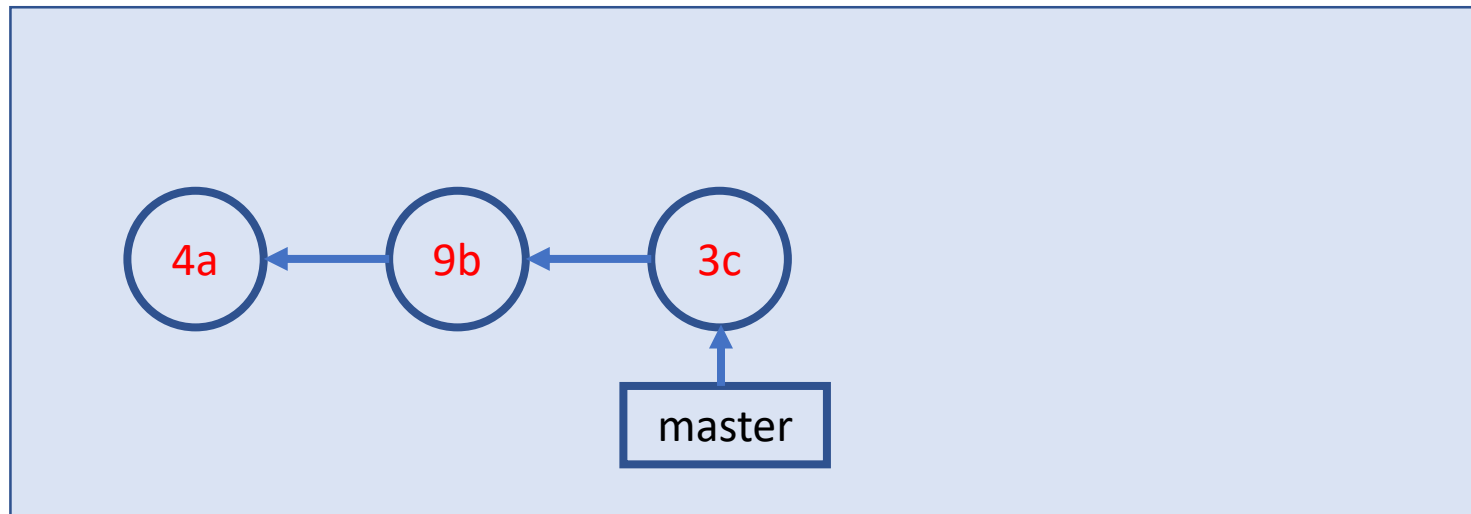
```
fork & git clone
```

# A Simple Remote Repo Workflow

Remote Repo

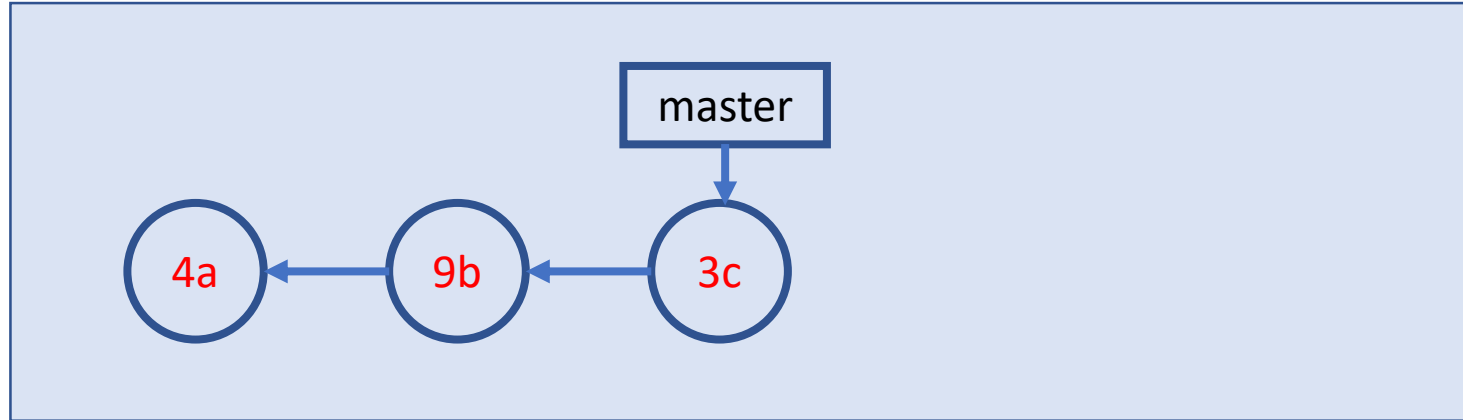


Local Repo

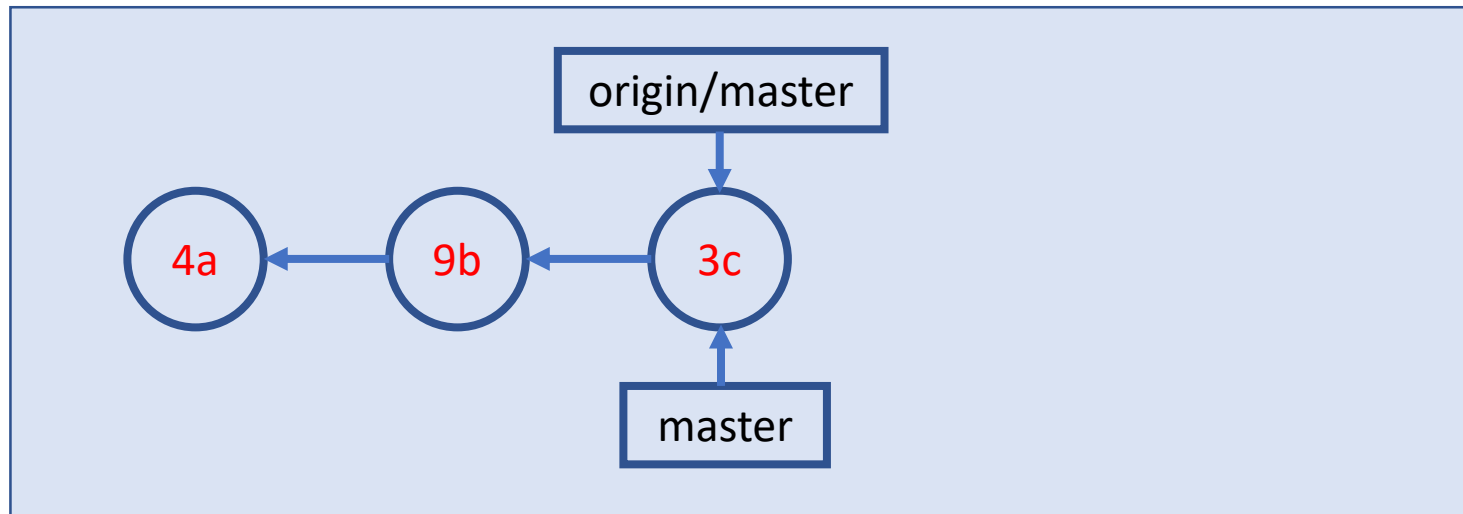


# A Simple Remote Repo Workflow `git push`

Remote Repo

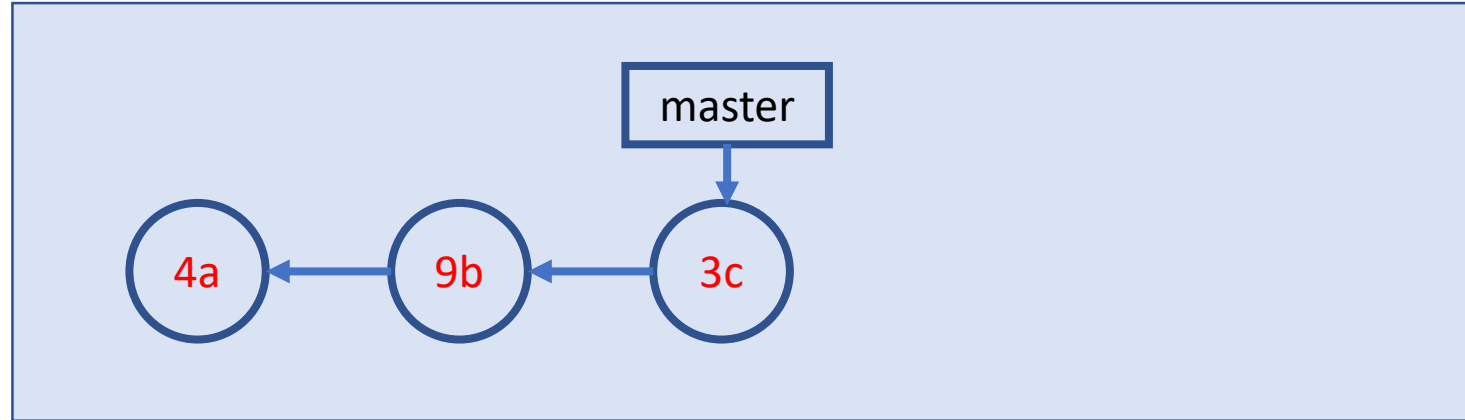


Local Repo

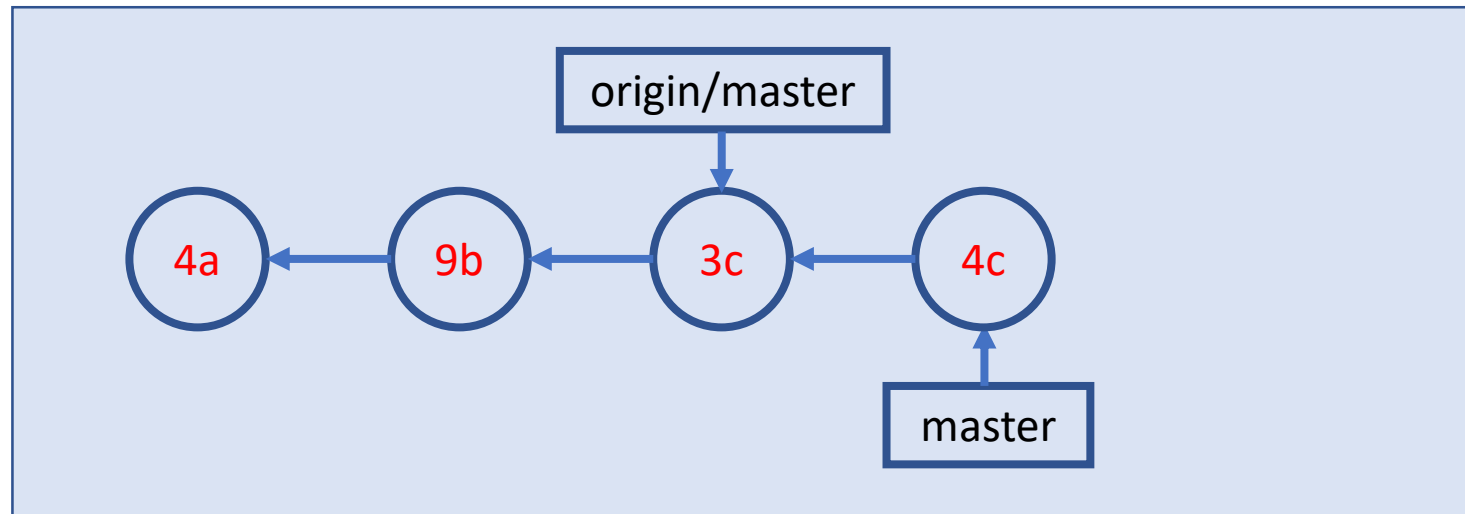


# A Simple Remote Repo Workflow

Remote Repo



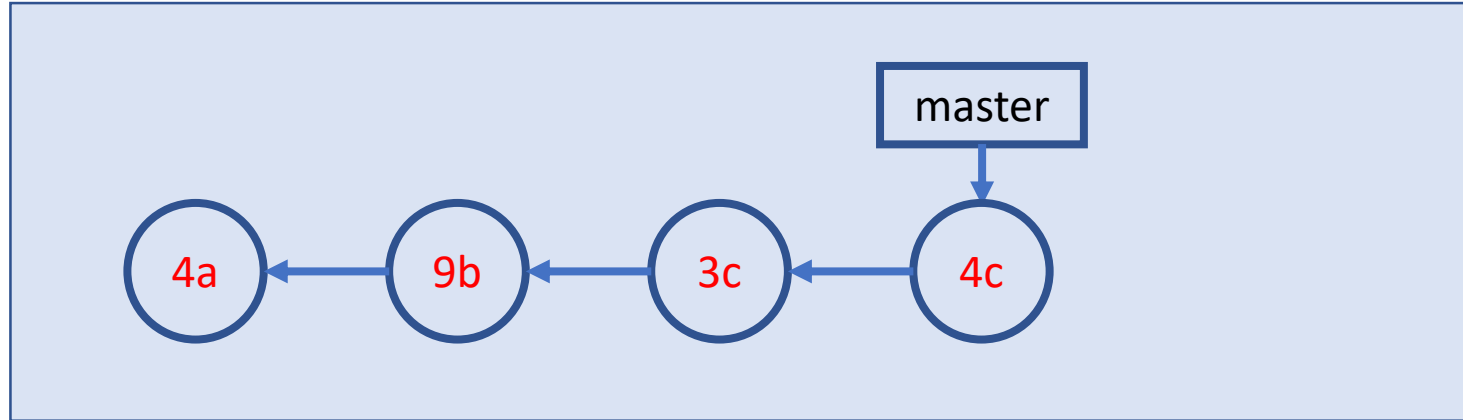
Local Repo



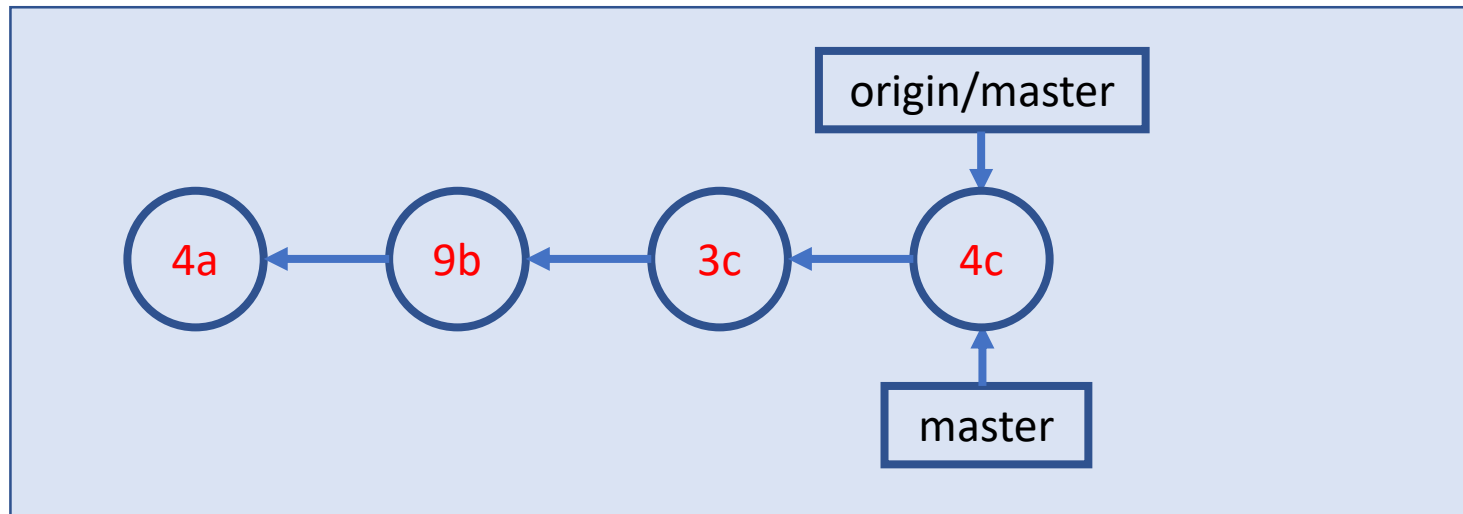


# A Simple Remote Repo Workflow `git push`

Remote Repo



Local Repo



# Many more to explore on your own

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

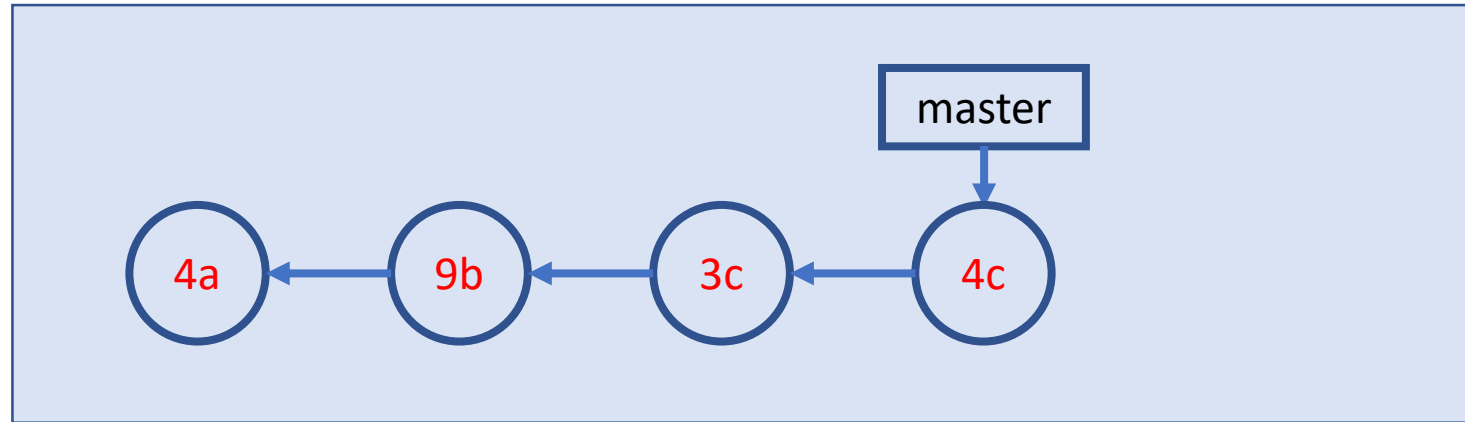
# Resources

- Git Ref Book: <https://git-scm.com/book/en/v2>
- Git Tutorials
  - [Version Control with Git](#) by Software Carpentry
  - [Git Essential Training](#) by Kevin Skoglund at lynda.com
    - login from [here](#) for UofT free access
  - [Get Started Tutorials](#) from Bitbucket Atlassian
  - [GitHub Guides](#)
- Git GUI (I recommend starting with command line)
  - dedicated GUI client: <https://git-scm.com/downloads/guis>
  - GUI integrated with IDE or code editor (e.g. RStudio, vscode, etc.)

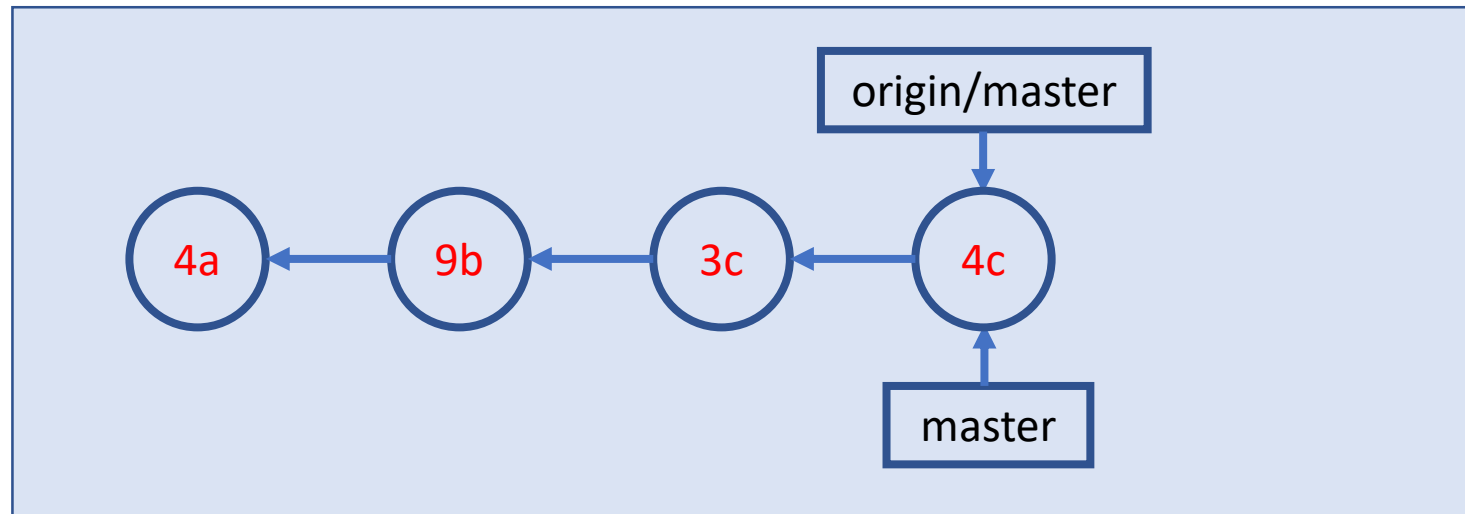
# Two More Git Workflows

# A Simple Collaboration Workflow

Remote Repo

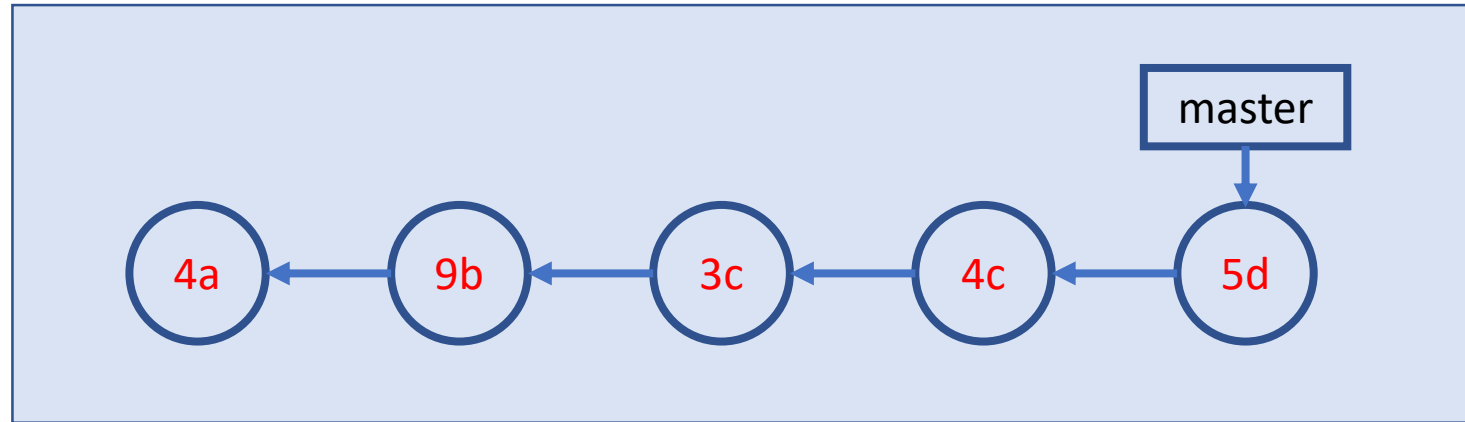


Local Repo

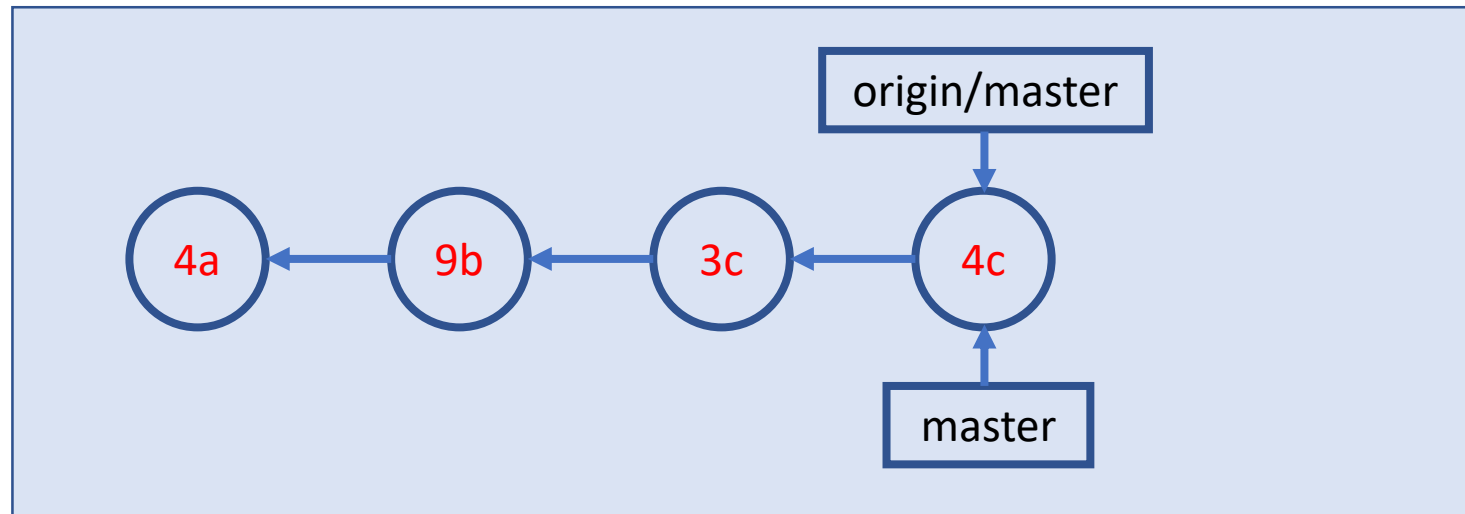


# A Simple Collaboration Workflow

Remote Repo

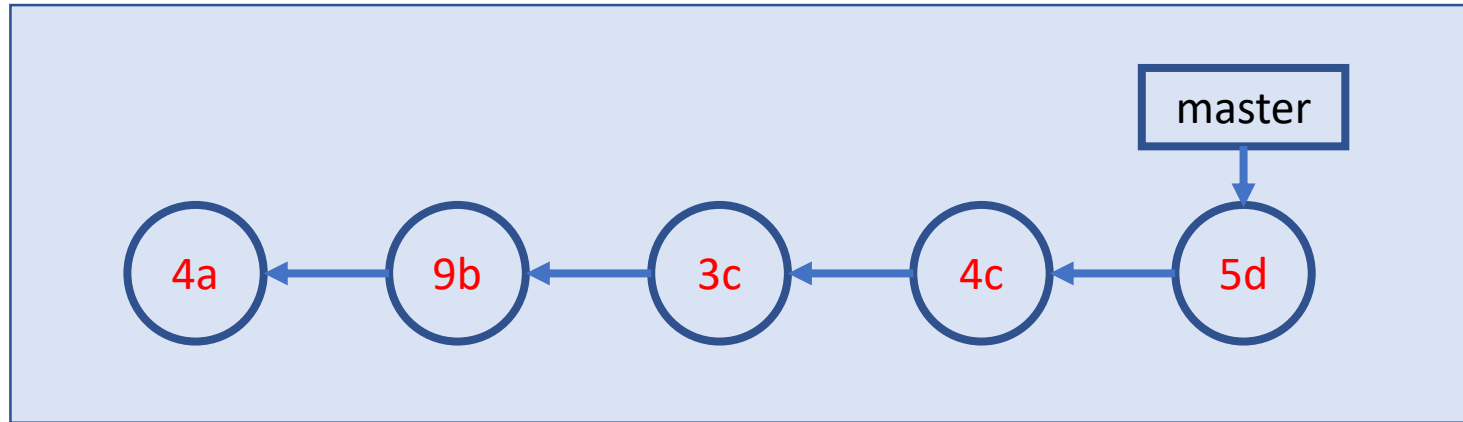


Local Repo

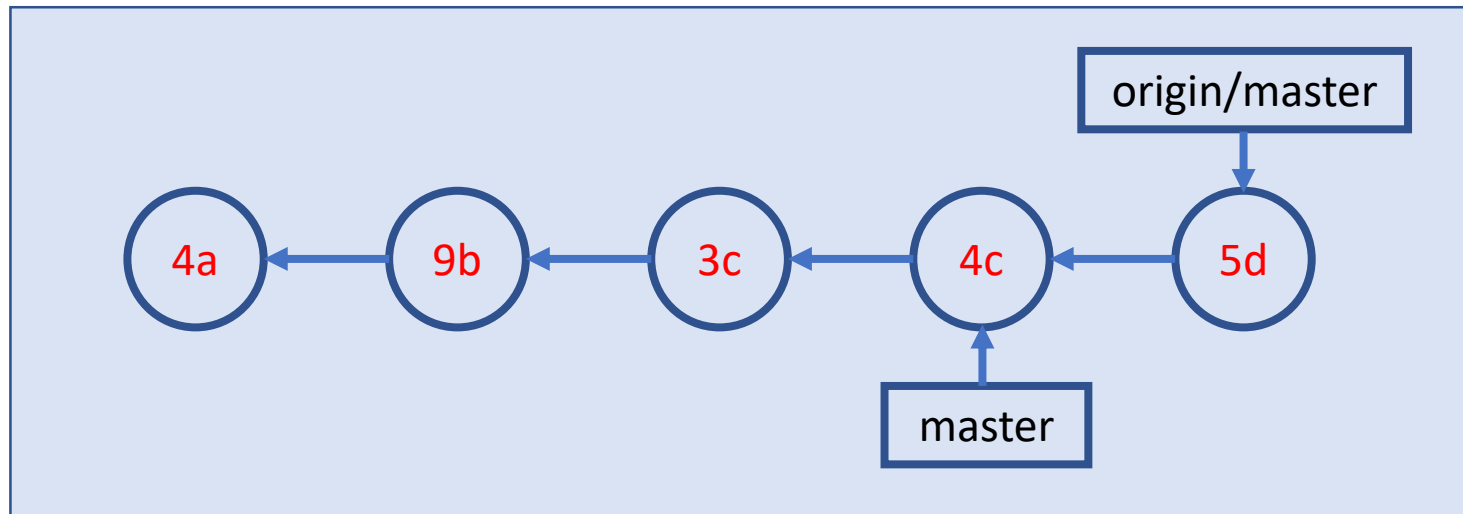


# A Simple Collaboration Workflow `git fetch`

Remote Repo

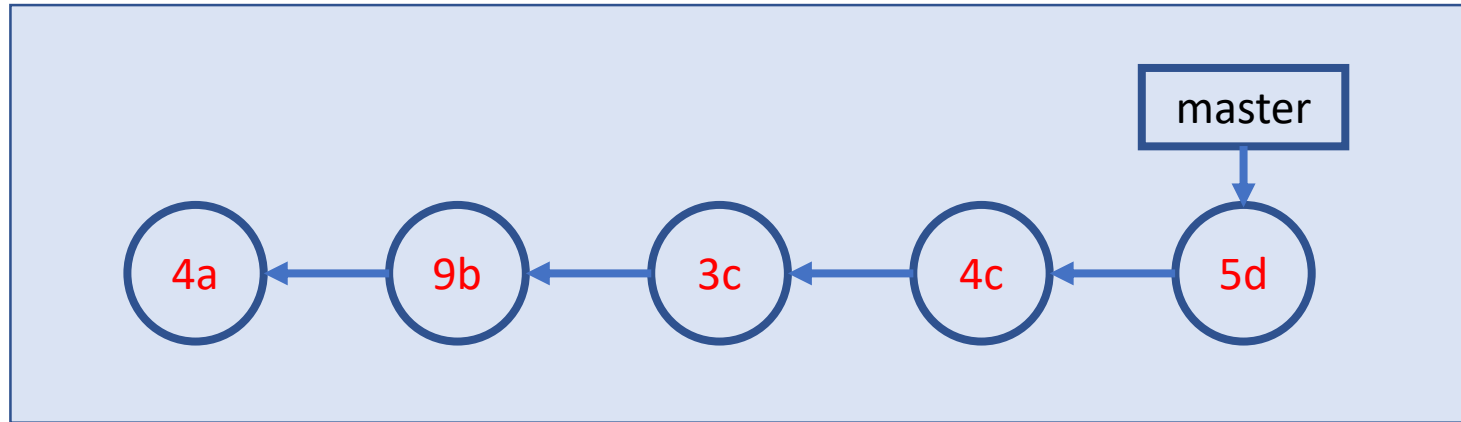


Local Repo

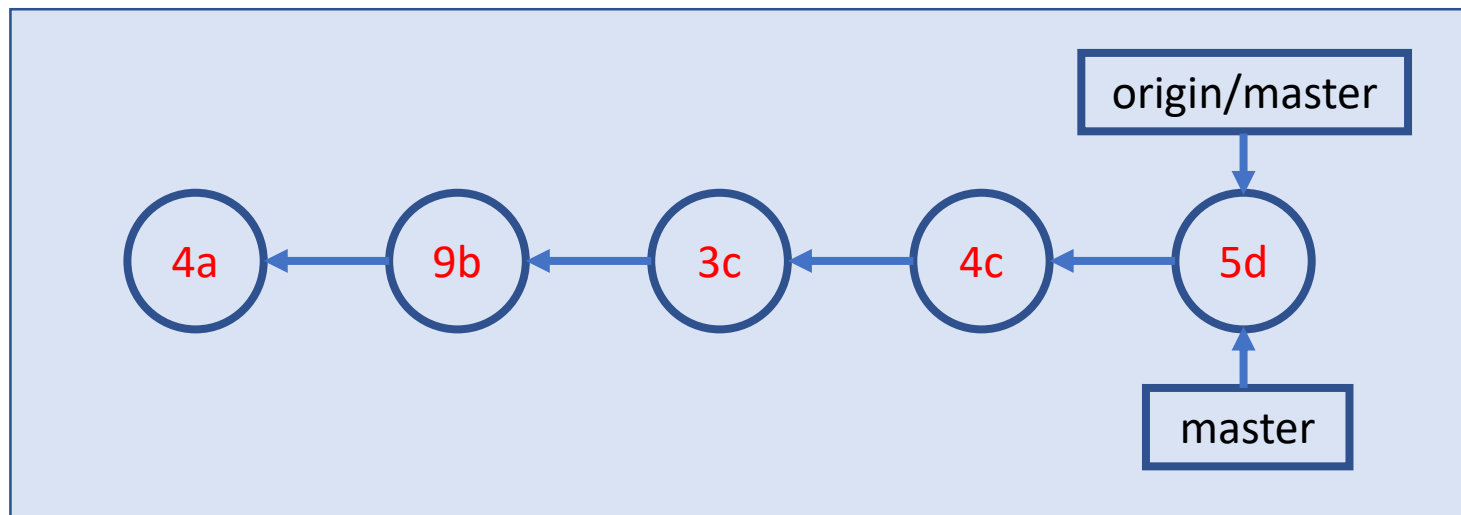


# A Simple Collaboration Workflow `git merge`

Remote Repo

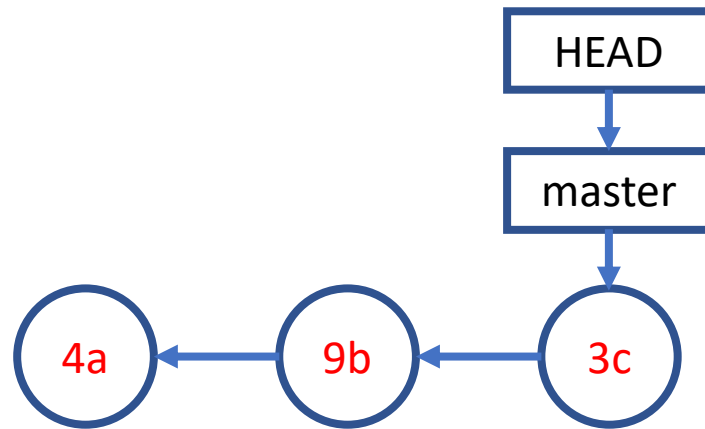


Local Repo



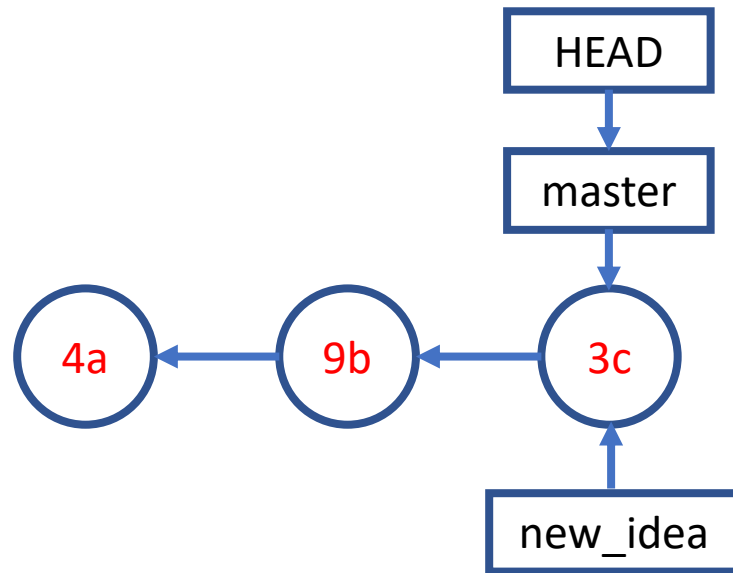


# A Simple Branching Workflow

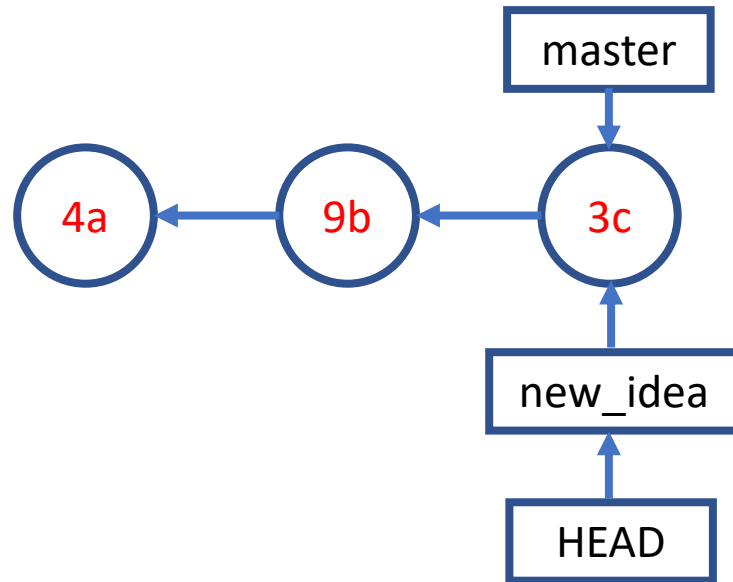


# A Simple Branching Workflow

```
git branch new_idea
```

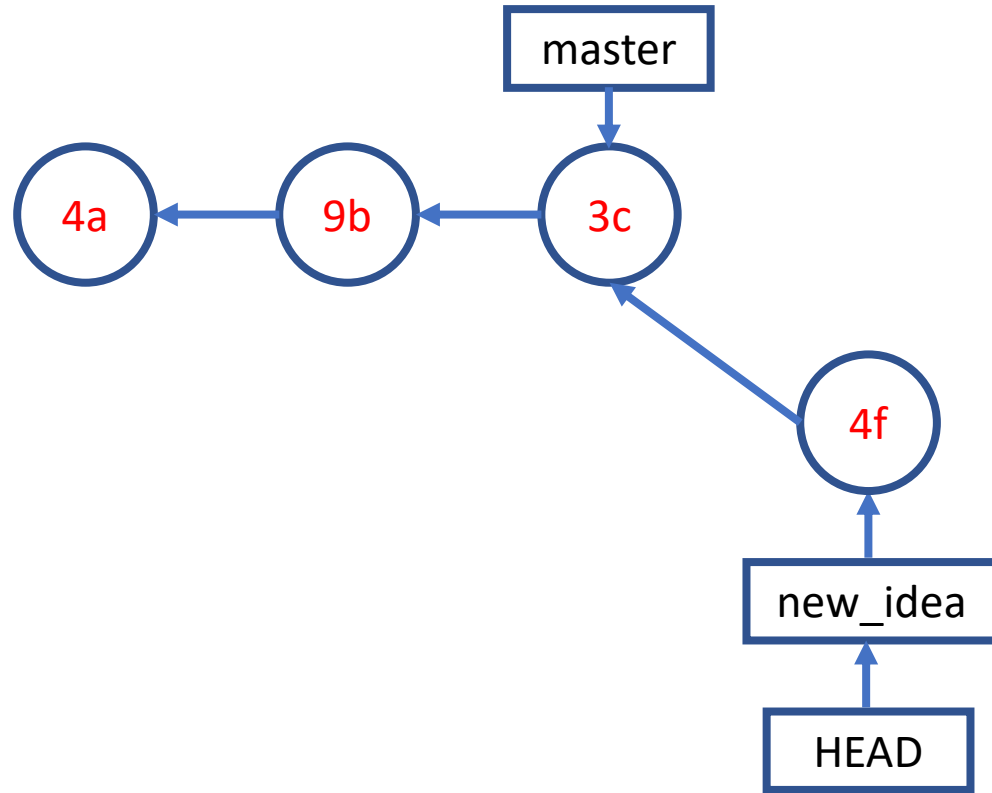


# A Simple Branching Workflow `git checkout new_idea`

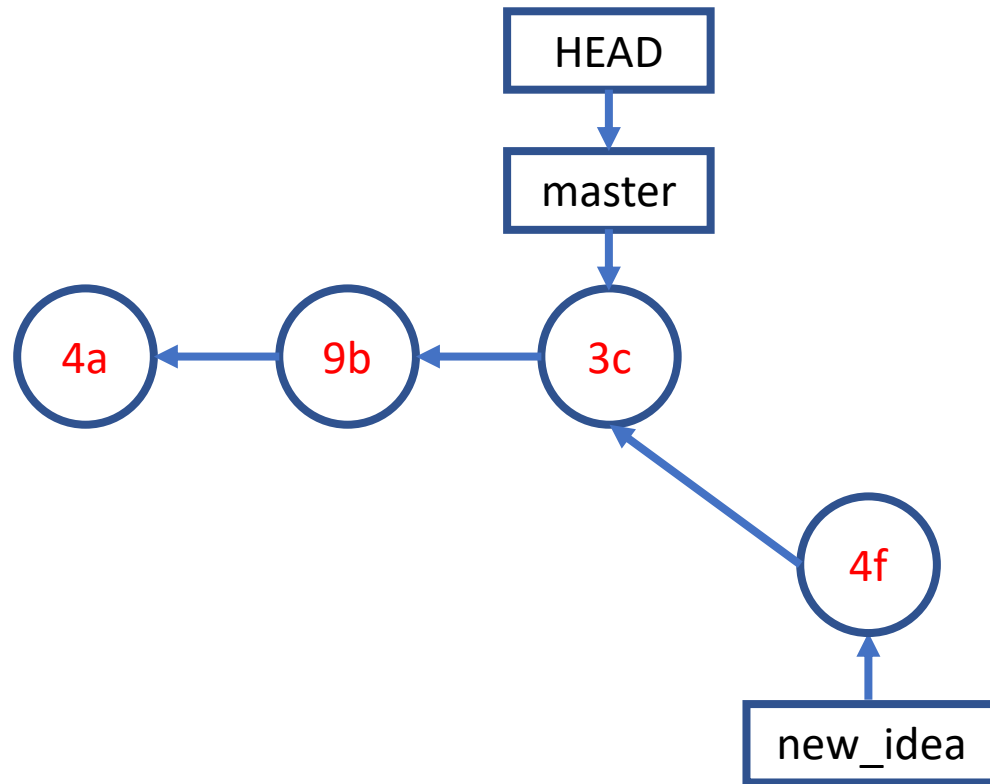


# A Simple Branching Workflow

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

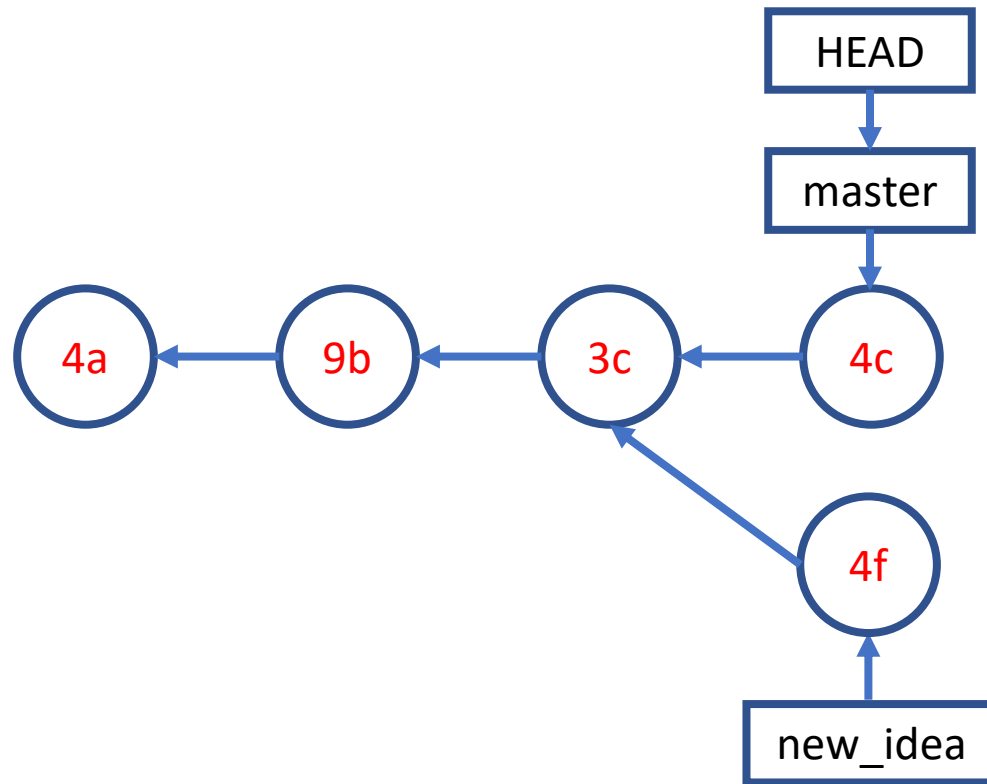


# A Simple Branching Workflow `git checkout master`



# A Simple Branching Workflow

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



# A Simple Branching Workflow `git merge new_idea`

