GIT Tutorial For Beginners

Link

Juliette Alimena, based on the tutorial by Tadej Novak

FH Sustainable Computing Workshop

November 7, 2025

Version Control Systems

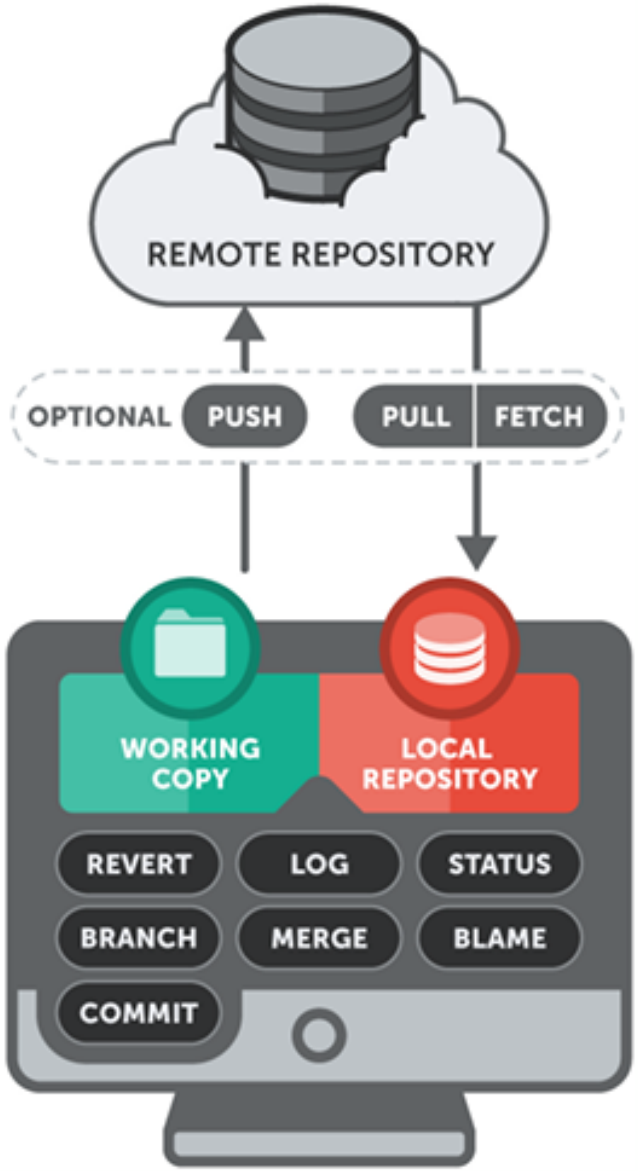
Why use version control systems/documentation, like GIT?

What is version control?

- Keep **history** of the code
  - Can revert if something goes wrong!
- **Collaborate** easily with other people
- **Automatically build or deploy** with every change (see CI tutorial!)

GIT is Distributed

- "**Distributed**": Each developer has (at least) one copy of the repository: **the local repository**
- Each developer interacts with one or multiple remote repository: **remotes**
- Online access needed only to share your work with others and obtain the changes introduced by others



GIT Basics

- **Initialise** a git repository: `git init`
- **Clone** a copy of a remote repo: `git clone <url> <local-folder>` Exercise 1
- main or master is the main branch of your repository
- **Make your changes available (steps to push to a remote):**
  - `git diff <filename>` (check what changes you made in a file)
  - `git add <filename>` (add a file with all changes)
    - Another option: `git add -p` (cycle through changes and add one by one)
    - Never use: `git add *`
  - `git commit -m "My commit message"`
  - `git push <remote> <branch>`
- **Display** which **remote repositories** are known: `git remote`
- **Display the current status** of your working copy (and its relationship with remotes): `git status`
- **Check the history**: `git log`

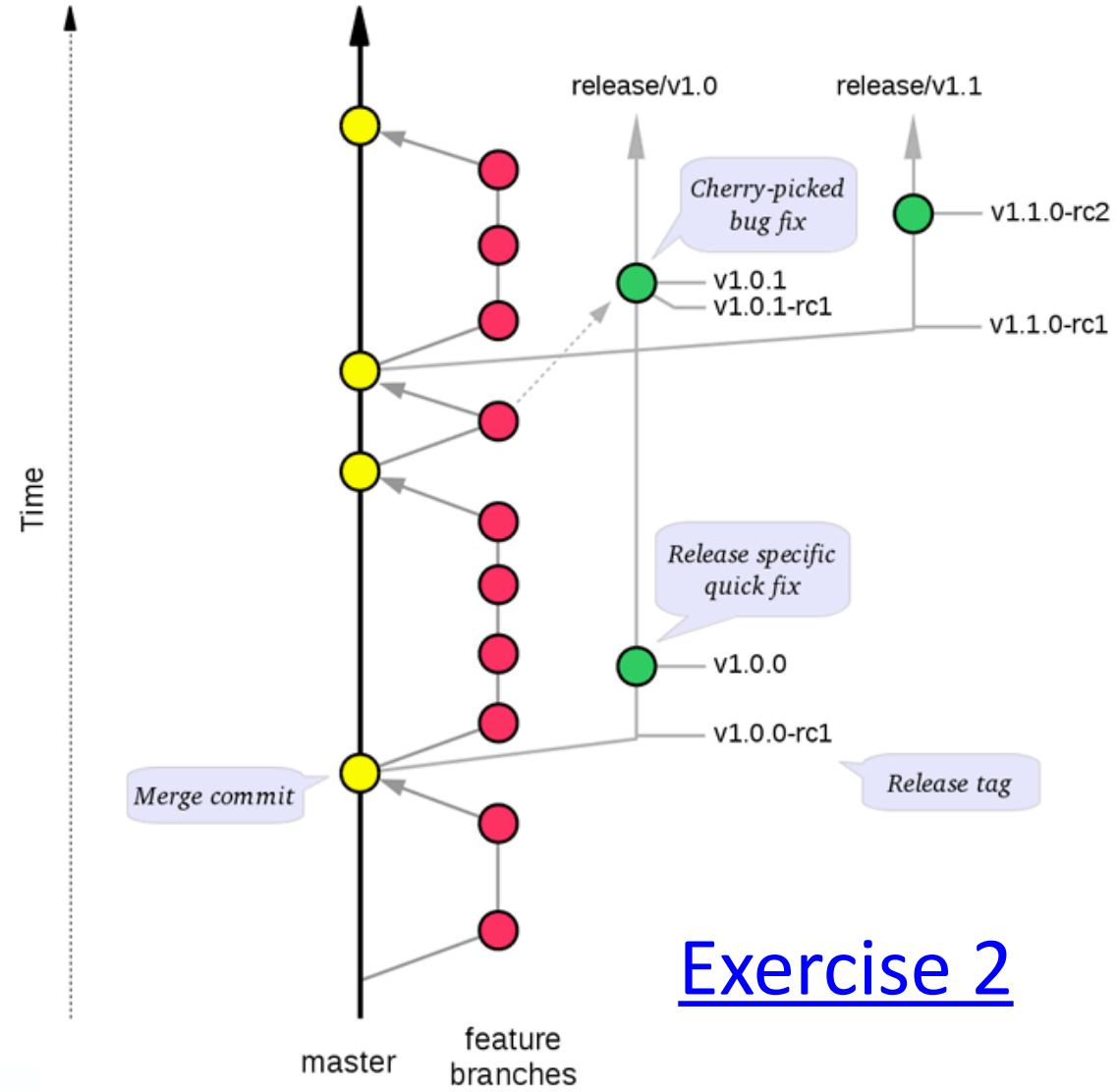
Organizing Your Work

Several options available to collaborate with others:

- Everyone pushes to the remote repository, OR
- Everyone make a fork of the repository Exercise 1
  - Each person pushes to their fork
  - When ready, submit a pull/merge request to the main repository
  - Can set the main repository as an "upstream" remote repository

Organizing Your Work: One Repo

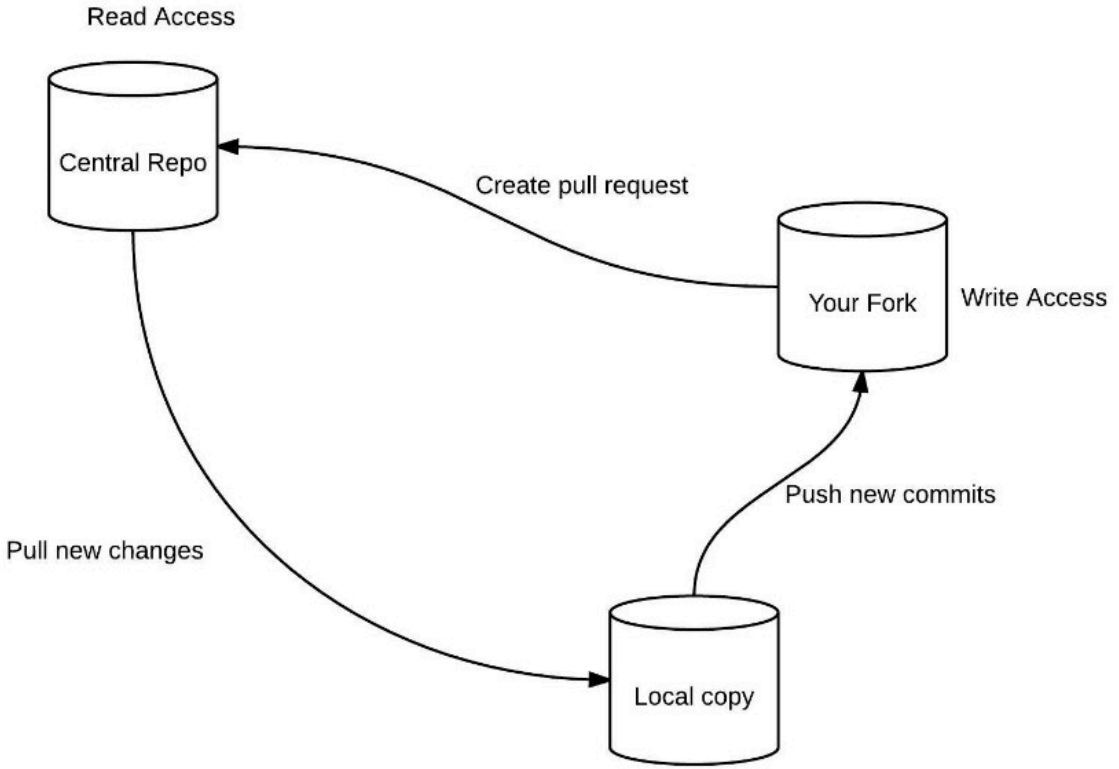
- One possible way to organize your work: everyone pushes to one repo
- main or master is the main branch of your repository
- Work on larger chunks in dedicated branches
  - `git checkout <branch>`
- Create tags to specify a state of your repository
  - `git tag <tag version>`



Exercise 2

Organizing Your Work: Forks and Merge/Pull Requests

- Another way to organize your work, particularly if you are collaborating with several other people, is to use forks and merge/pull requests
- **Fork**: your own online copy of the repository
  - After you fork the original repository, you can clone your fork locally
  - You can push changes to your fork without affecting the original repository
- When you are ready, you can submit a merge/pull request to merge your changes from your fork back into the original repository

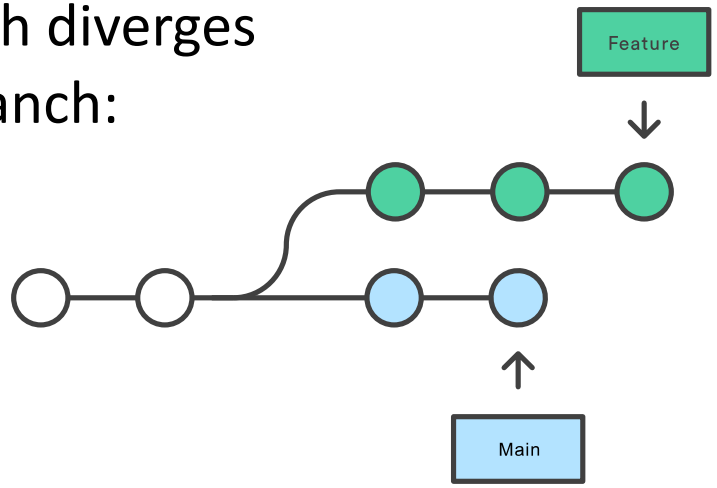


- Can add the original repo as an upstream remote repo
  - And get back the changes that occur in the upstream repository into your fork with merge or rebase

Merge vs Rebase

Exercise 3

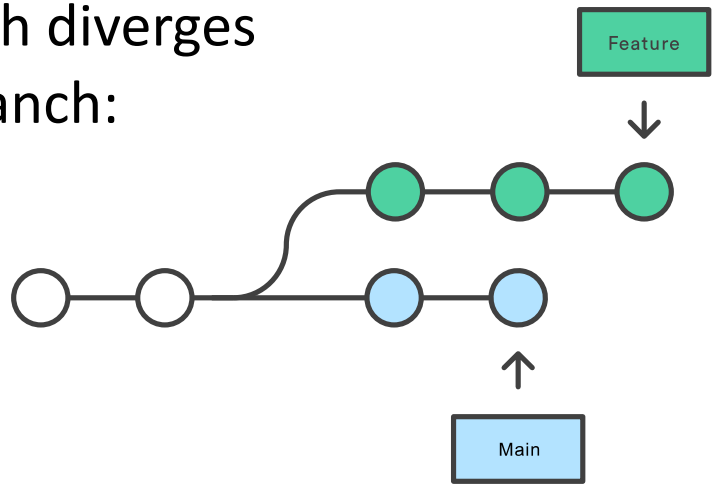
The feature branch diverges from the main branch:



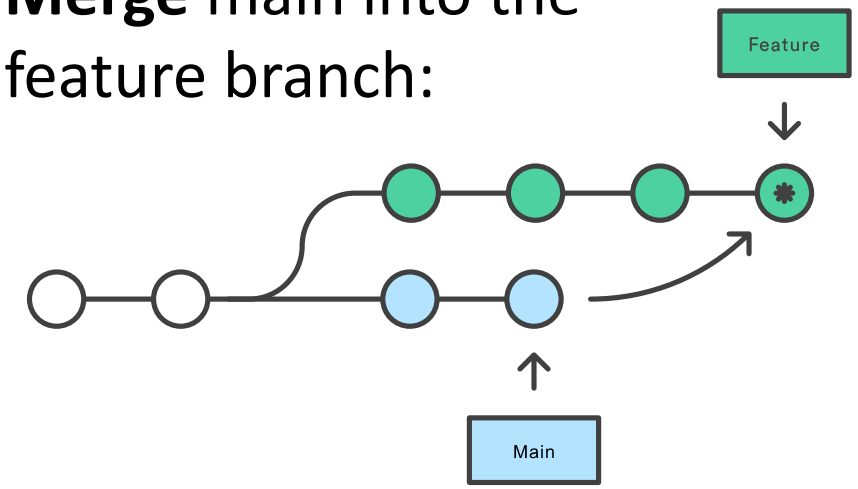
Merge vs Rebase

Exercise 3

The feature branch diverges from the main branch:



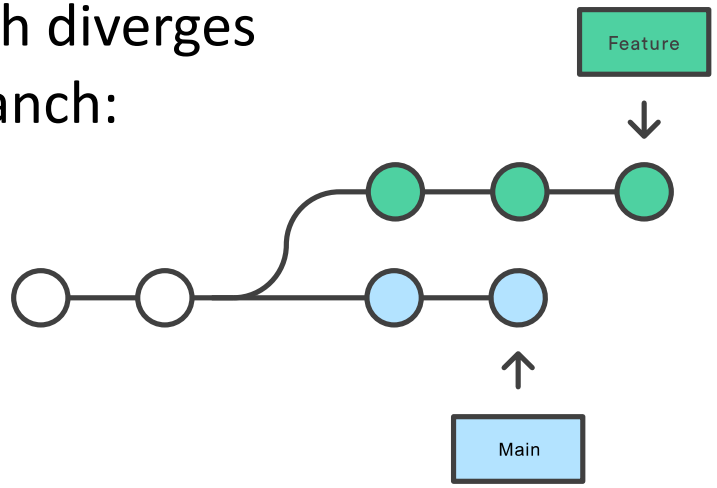
Merge main into the feature branch:



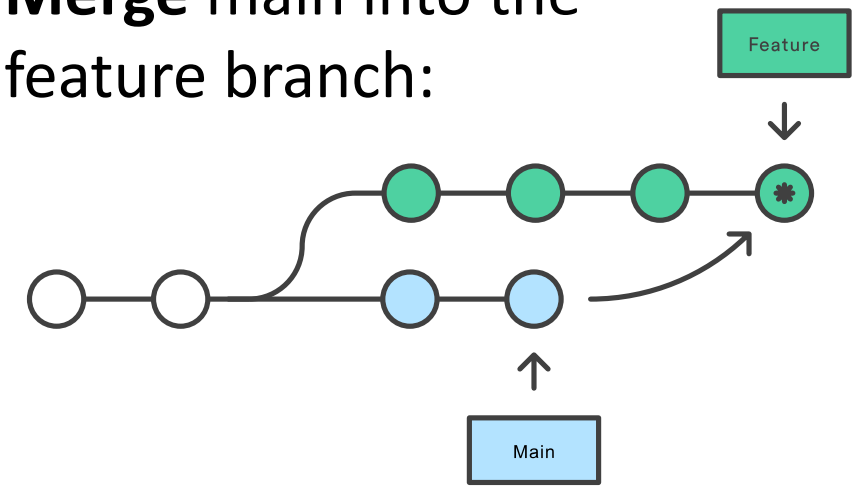
Merge vs Rebase

[Exercise 3]

The feature branch diverges from the main branch:

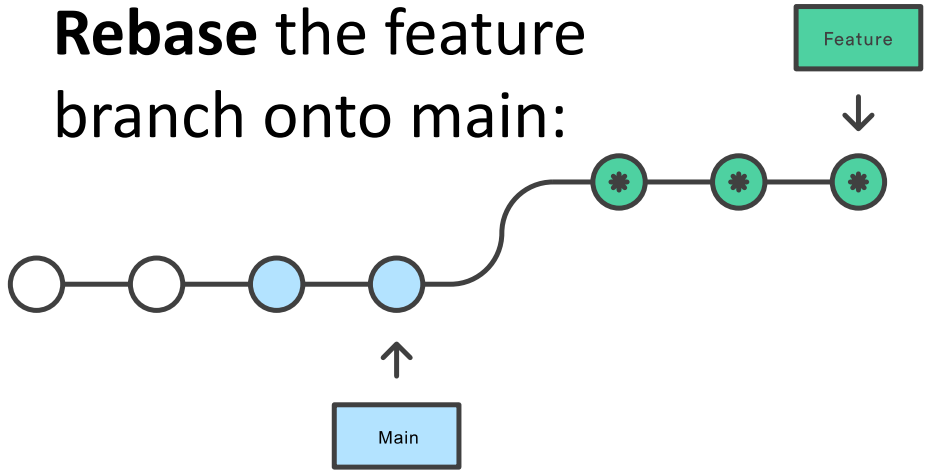


Merge main into the feature branch:



OR

Rebase the feature branch onto main:



Merge vs Rebase

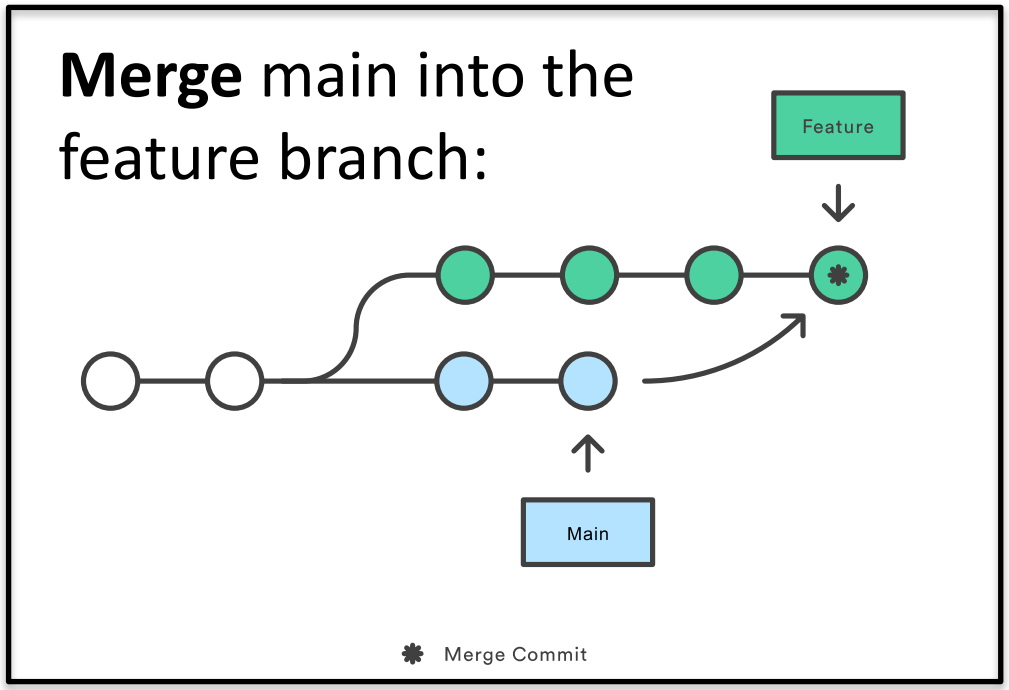
Exercise 3

In merging:

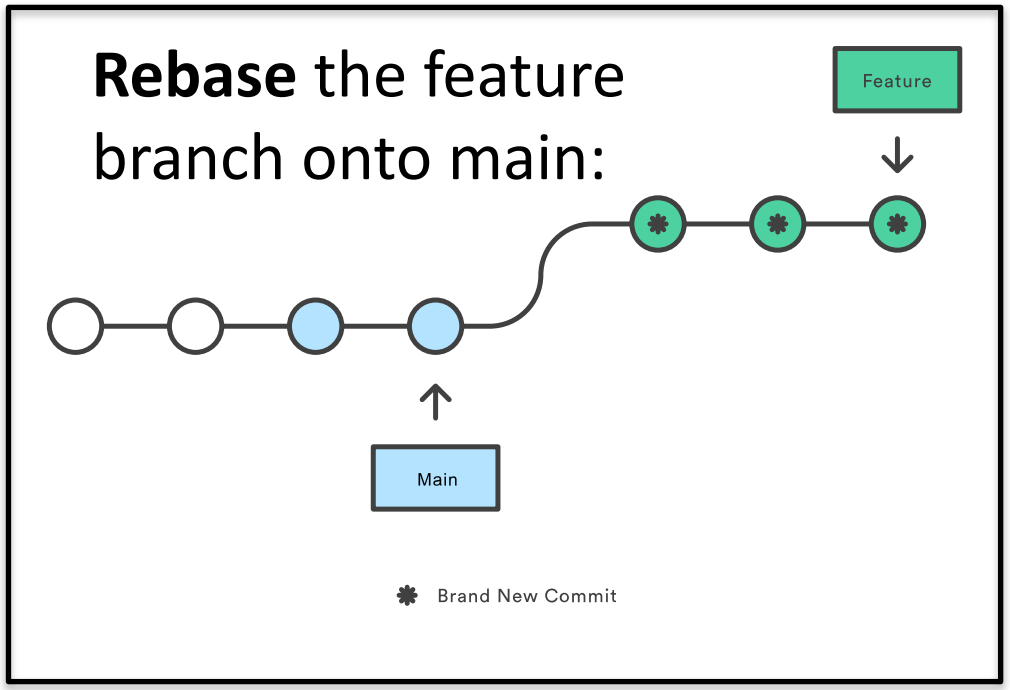
- Main doesn't get changed
- The feature branch will get an additional merge commit at the end

In rebase:

- The entire feature branch is moved as if it happened after the latest commits in main



OR



Merge and Rebase Commands

Merge

- Always go to the branch you want to merge your changes to: `git checkout master`
- Merge the changes: `git merge super-new-feature`

Rebase

- Go to the branch you are working on: `git checkout super-new-feature`
- Rebase onto a branch (e.g. master): `git rebase master`
- Merge like usual

Providers

- Basically, you have GitLab and GitHub as online providers
- DESY provides an instance of GitLab: https://gitlab.desy.de
  - We will be using this today: 3 Git Exercises on DESY GitLab
- CERN also has an instance of GitLab
- GitHub is one of the other popular hosting providers

Some Technicalities

- The local repo is located under a folder named `.git` in your working copy
- Git uses 160-bit (SHA-1) hashes to point to a given state of your repository
- This hash is always **unique** (collision unlikely)
- It is hard to memorize and not very practical — we use **references**:
  - HEAD is a reference to the hash describing the current status
  - main or master is a **branch** (usually the main branch)
  - release/21.0.21 is a tag
- A given hash points to the same content, even if from different repositories, branches, tags, …
- For small repositories, 7 leading digits of the hash are (usually) enough
- For example, to show the contents of one commit: `git show e3153d7`

Git Ignore

- You can tell Git to ignore certain files and never look at them when you want to commit
- This can be useful when you have extra things in your local directory, for example root files or pdfs that you made but don't want to track in Git
- To do this, just create a file in your directory called **.gitignore** and in this file, create a list of files (or types of files, e.g. *.zip) for Git to ignore
  - Here's a list of some common file types to ignore
- Don't forget to add/commit/push your .gitignore file to your repo ;)

More Useful GIT Commands (~advanced)

- **Stash:**
  - Temporarily save and remove your local changes
  - Add to stash: `git stash save <name>`
  - List stashed: `git stash list`
  - Retrieve from stash
    - but do not remove it: `git apply <name>`
    - and remove it: `git pop <name>`
  - <name> is always optional
- **Cherry-pick:**
  - Take one hash and apply it to your current staging area:

    `git cherry-pick e3153d7`

What If?

I've made a lot of changes but now I want to submit just some of them?

- Look at the log and decide what you want (`git log`)
- Make a new branch from upstream/master: `git checkout -b feature-a upstream/master`
- Cherry-pick commit(s) that you found in the log:
  - `git cherry-pick e3153d7`
  - `git cherry-pick f249a34`
- Push to the origin: `git push -u origin feature-a`

What If?

I’ve made a commit but I forgot a file?

- Add the file that is missing: `git add missing.txt`
- Update, that is, amend the commit: `git commit --amend`

What If?

I've added too much to the staging area/commit?

- If not committed yet, reset the file: `git reset my-file.txt`
- If have already committed, reset the state to the previous HEAD: `git reset --soft HEAD~1`

Summary

- Git helps you keep **track of the history** of your (nonbinary) files and **easily collaborate with others**
  - Code
  - Thesis (git works with overleaf…)
  - Recipes
- A lot of material covered today — **the goal should be to understand the philosophy**
- With the exercises, the goal is for you to **understand the basics** of how to do these things in practice
- **Useful links:**
  - https://training.github.com/downloads/github-git-cheat-sheet.pdf
  - https://learngitbranching.js.org/
  - https://ohshitgit.com
- But in general, if you have a git problem, **google/ChatGPT is your friend**
  - Countless people have screwed up like you just did before you