

A photograph of several LEGO minifigures in a workshop-like setting. In the foreground, a female minifigure with brown hair in a ponytail, wearing a pink lab coat, holds a large black wrench. To her right is a computer monitor on a stand, displaying a green screen with a white icon of a document with a checkmark. Behind the monitor, a male minifigure with a blue cap and a sad expression is visible. To the right of the monitor, another male minifigure wearing a white hard hat and a red safety vest is looking towards the camera. The background is a plain, light-colored wall.

Git Background—Distributed Version Control

Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

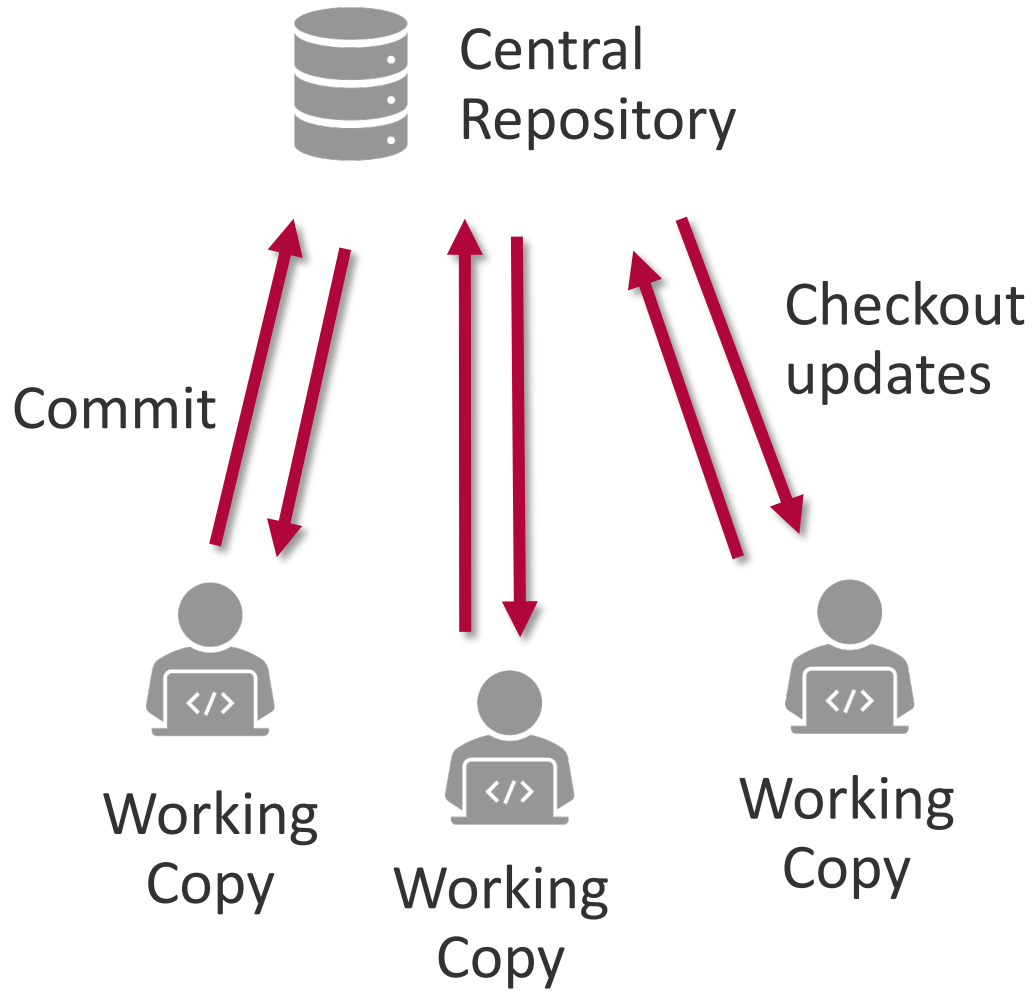
Outline

- Design of VCS
- What happens under the hood?
- Collaboration in teams



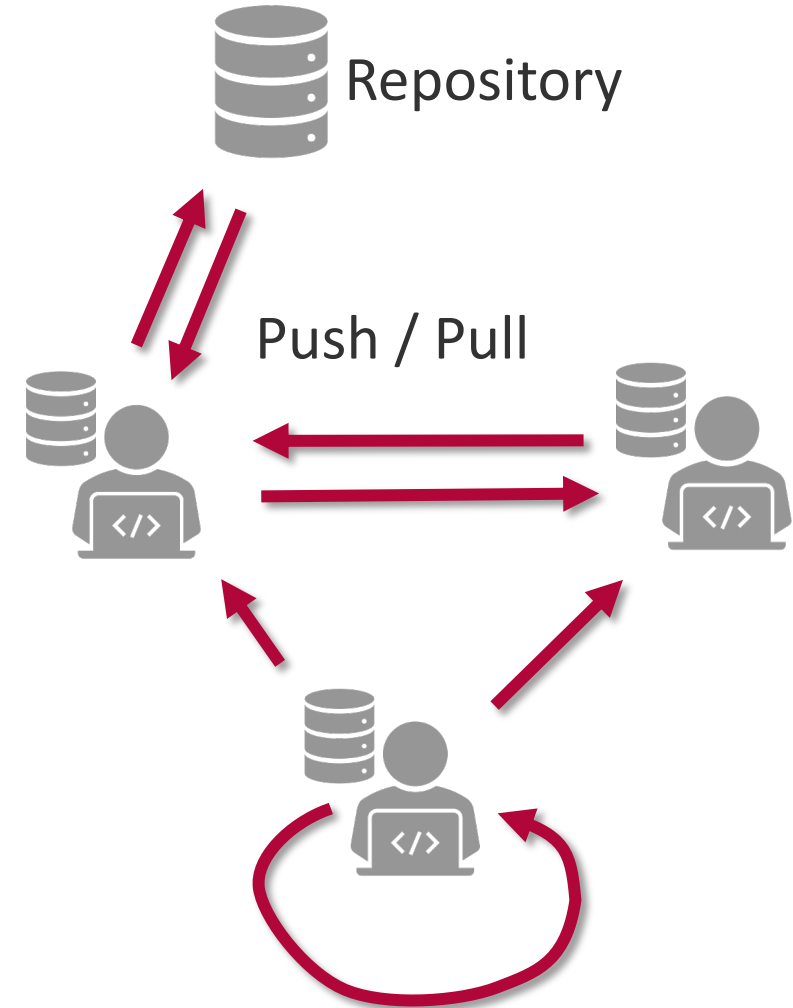
<https://xkcd.com/1597/>

Centralized vs Distributed VCS



Client-Server

VS.



Peer-to-Peer

Centralized vs Distributed VCS



Using Version Control Systems

- **Distributed VCS mostly used like centralized ones**
 - One peer who is always online and acts as central (git)hub
- Local commits are blessing and curse
 - Commits can be made while offline
 - Higher chances of code diverging



Git Objects



Blob

- Content of a file
- Nothing else

Tag

- Reference to other object

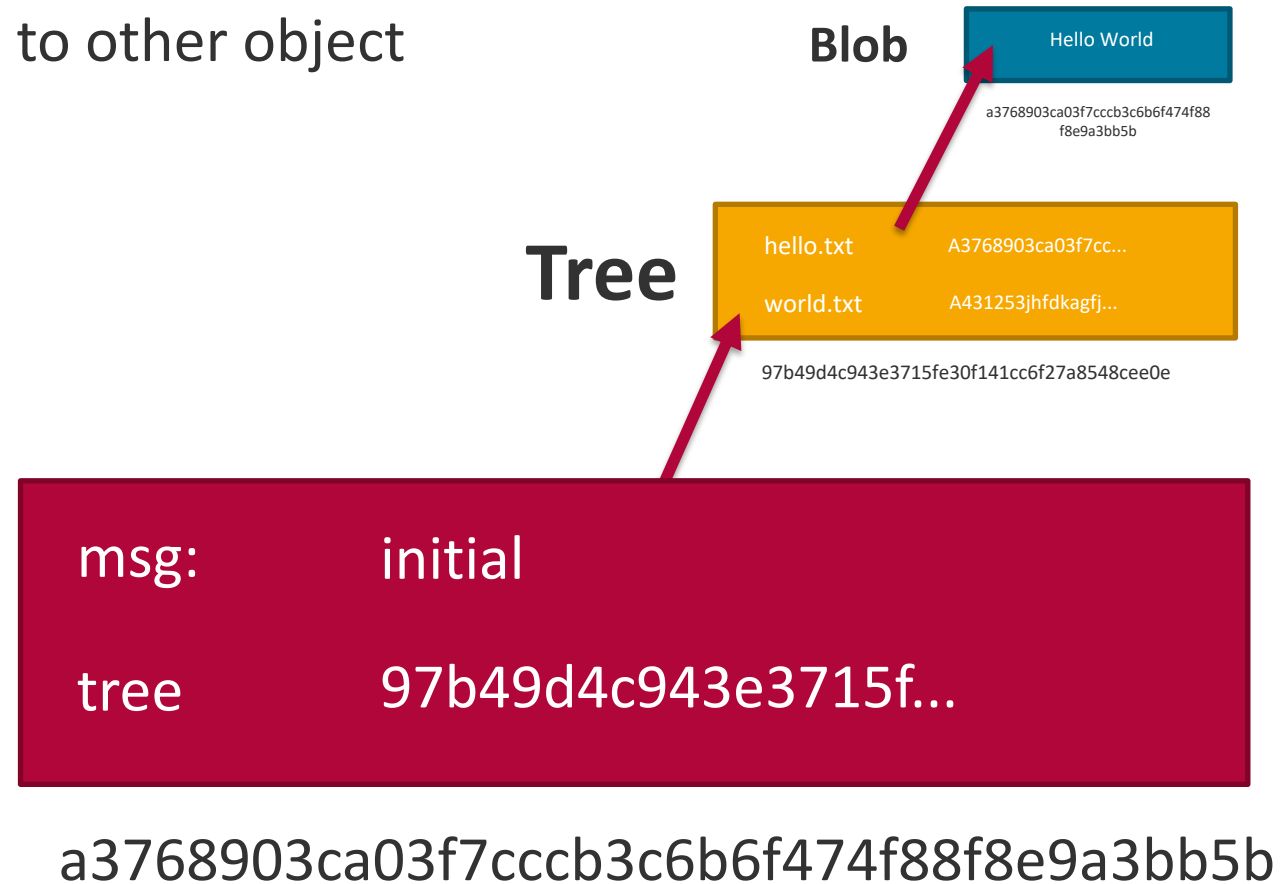
Tree

- Folder structure, file names
- References Blobs

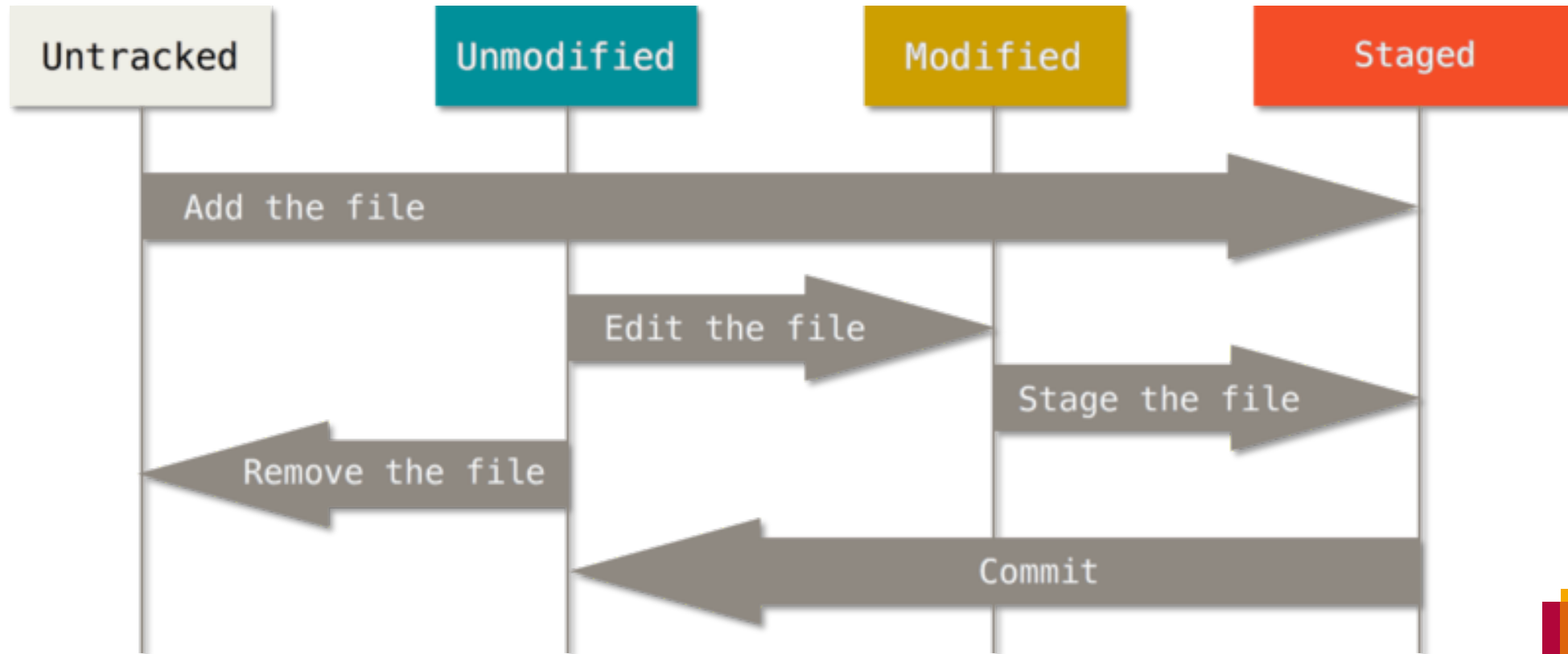
Commit

- References Tree object
- Metadata
- 0..* parent commits

Commit

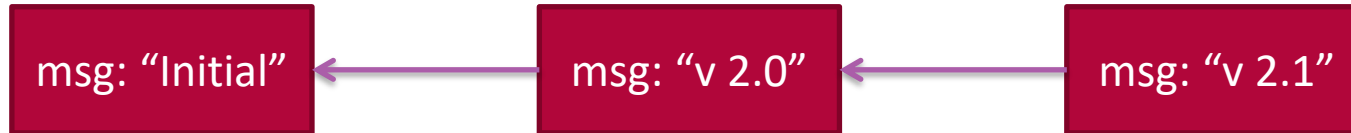


Git Local File Status Lifecycle



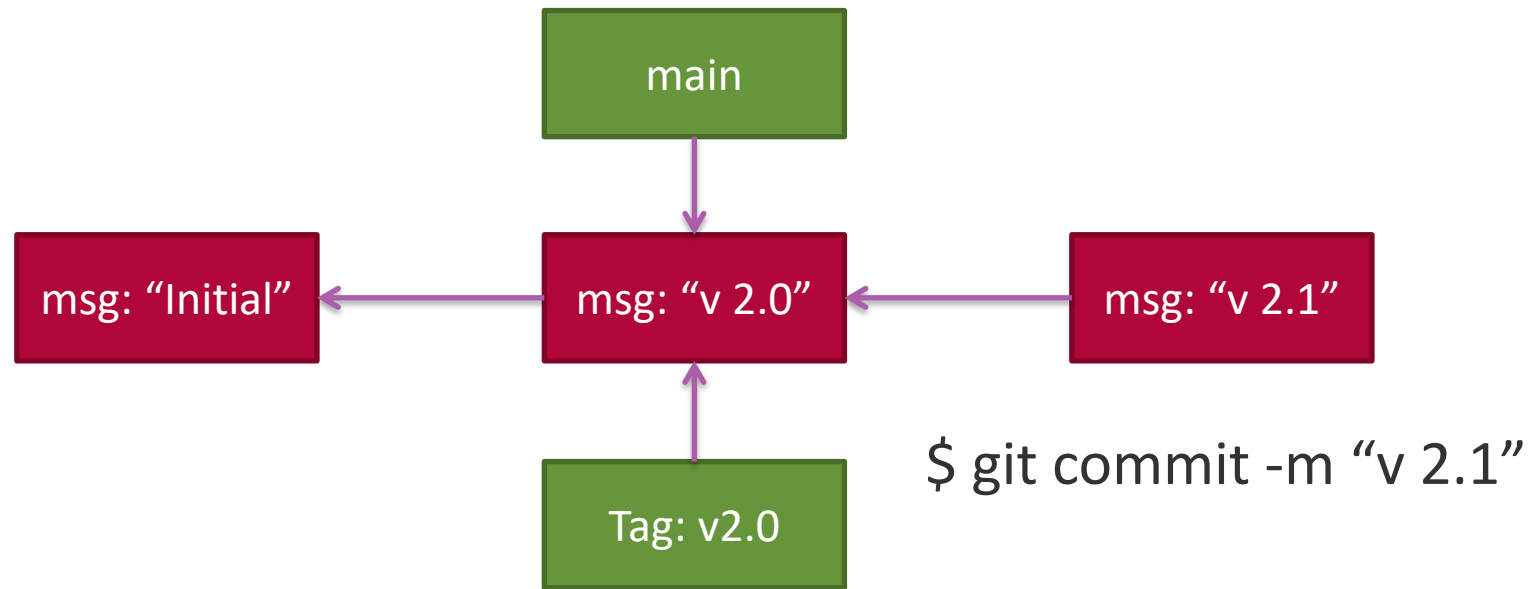
Short git status:
`git status -s`

Commit Parent

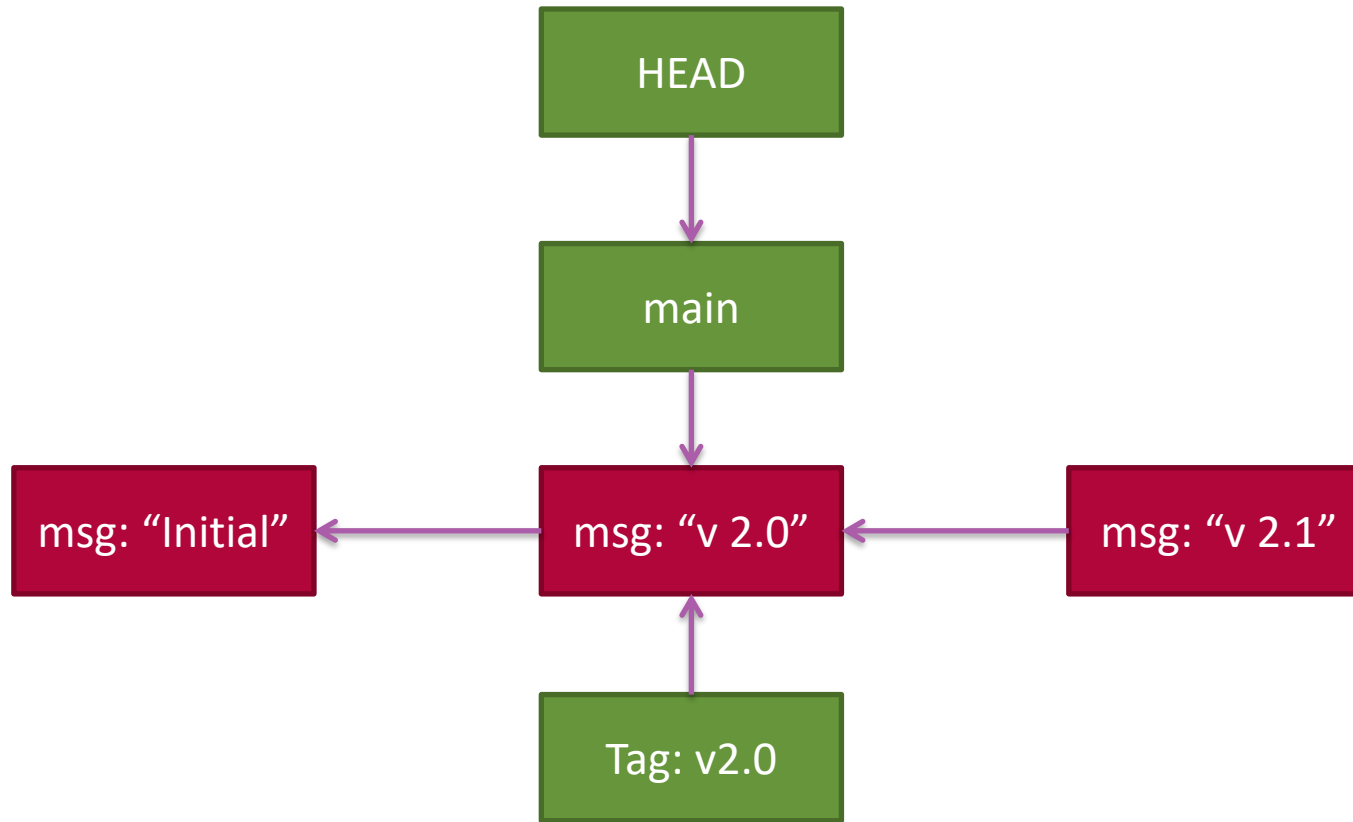


```
$ git commit -m "v 2.1"
```

Branches & Tags



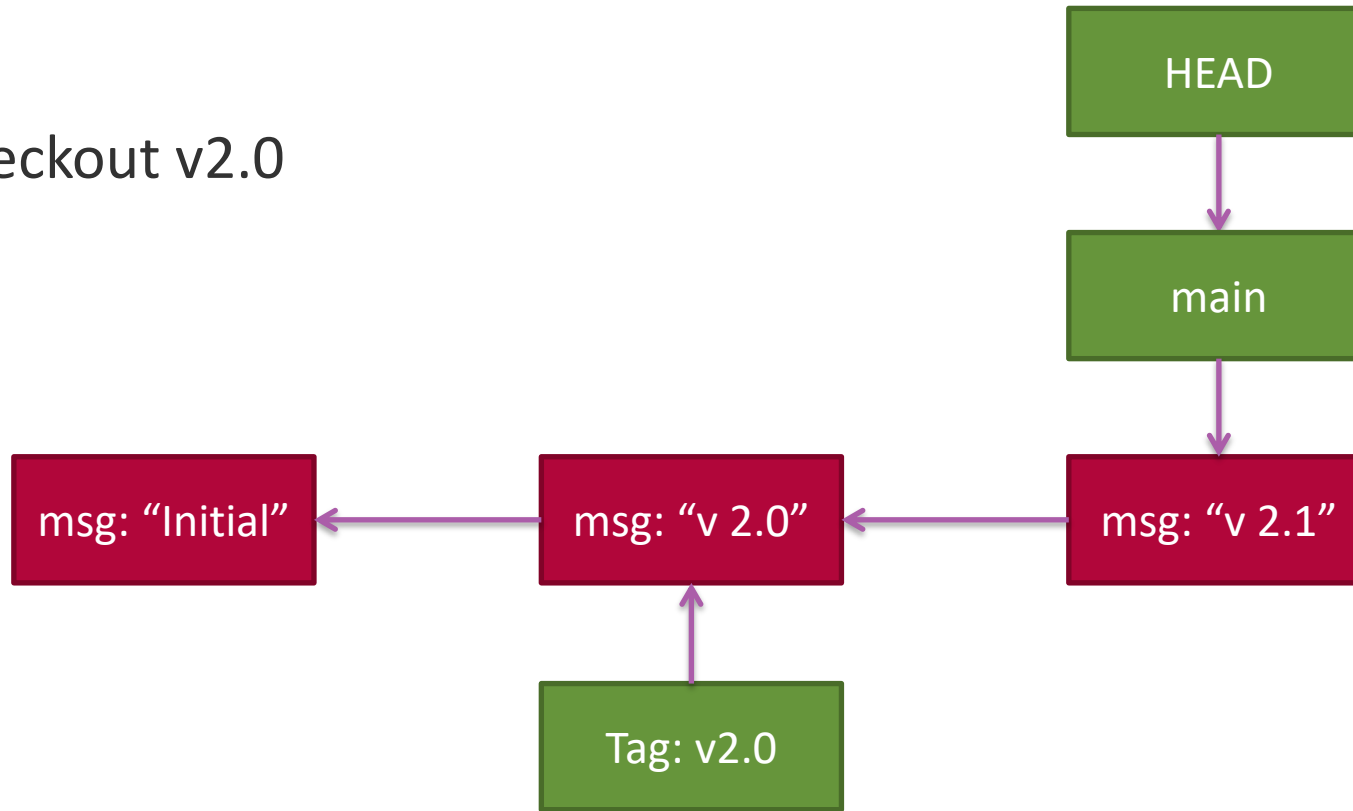
Head



Detached Head



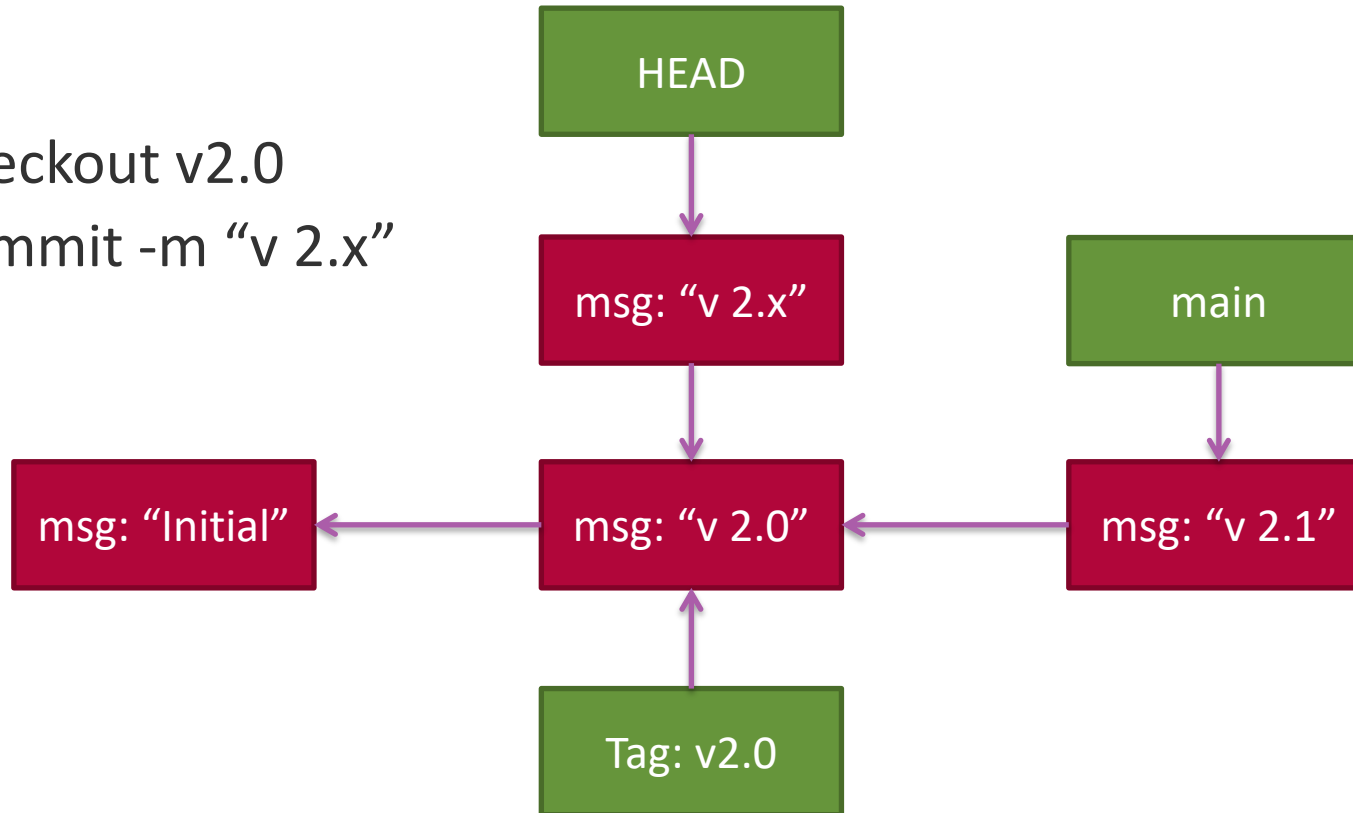
\$ git checkout v2.0



Detached Head



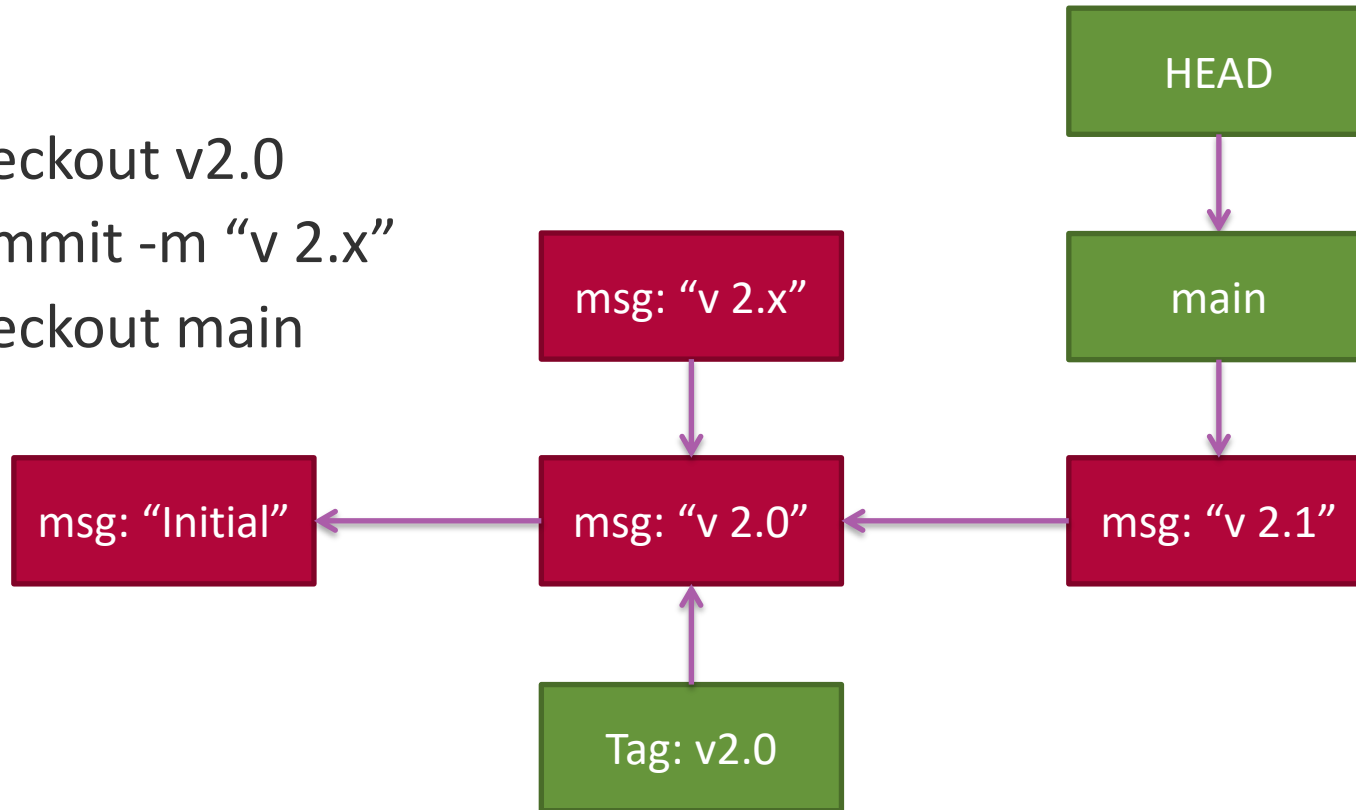
```
$ git checkout v2.0  
$ git commit -m "v 2.x"
```



Detached Head



```
$ git checkout v2.0  
$ git commit -m "v 2.x"  
$ git checkout main
```

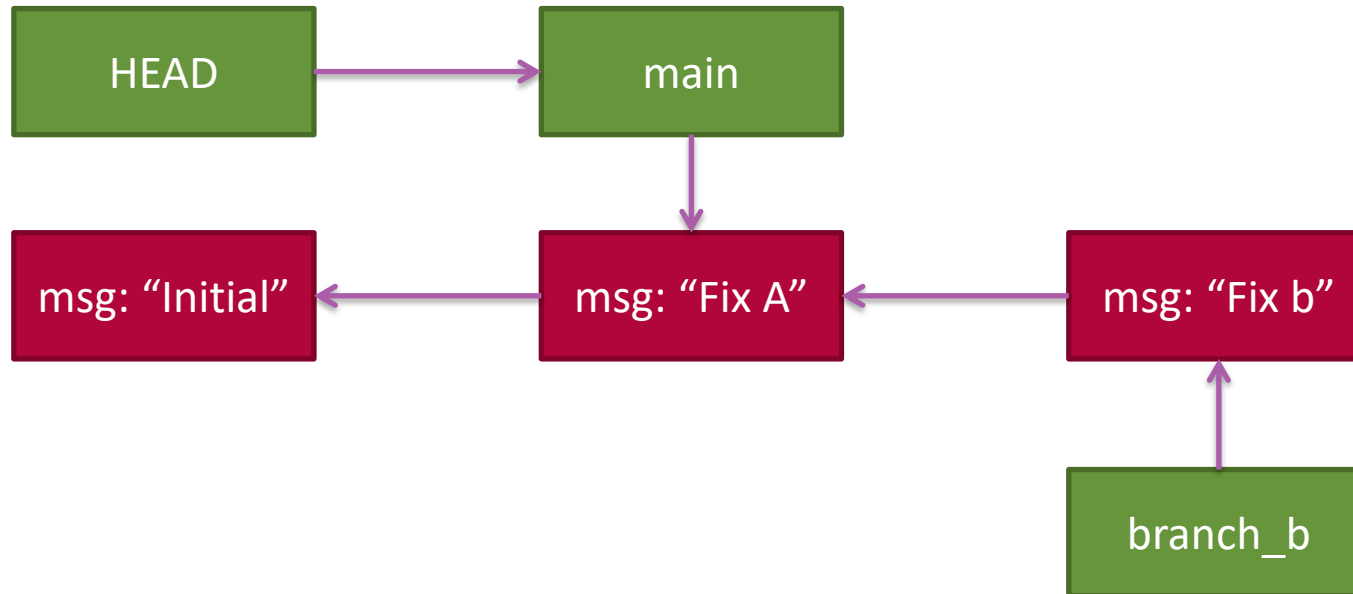


List all commits:
git reflog

Fast-forward



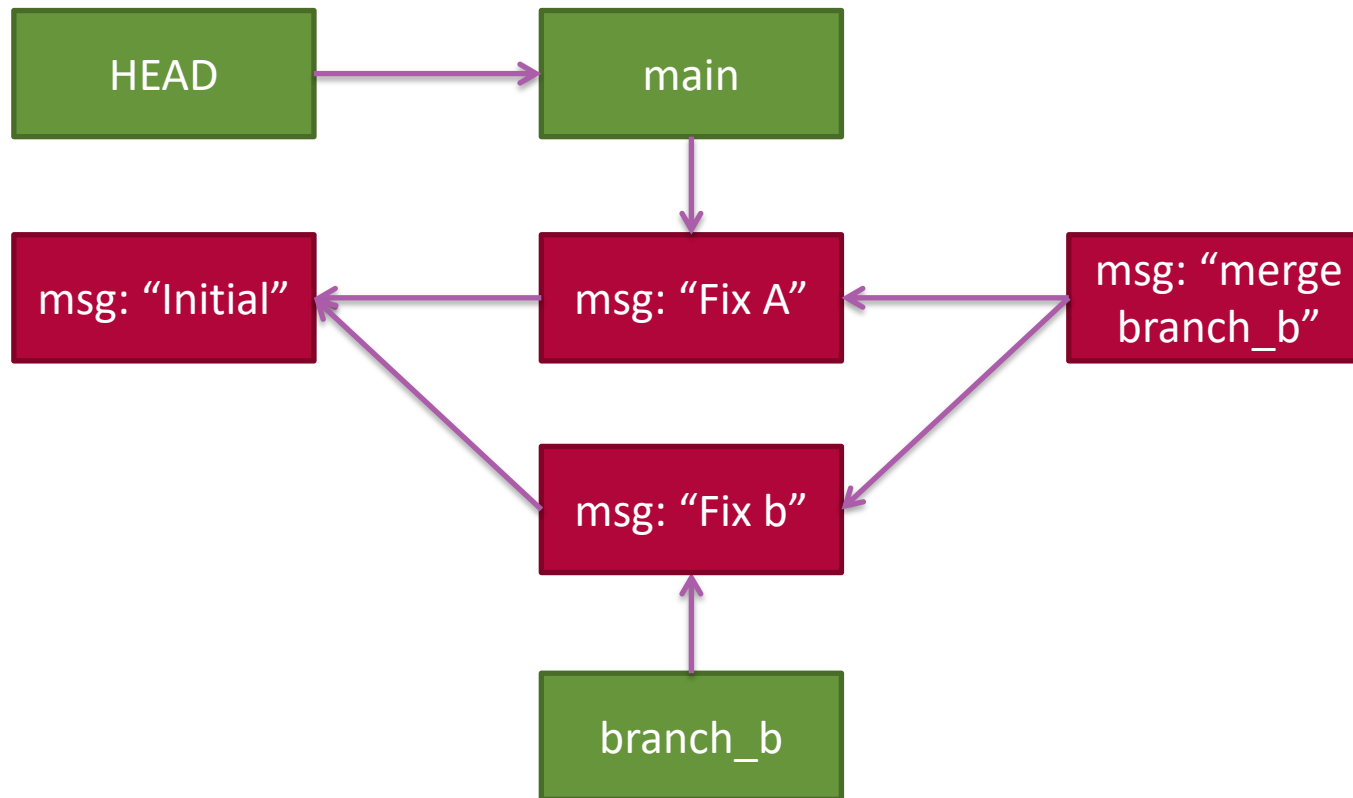
\$ git merge branch_b



Merge



\$ git merge branch_b



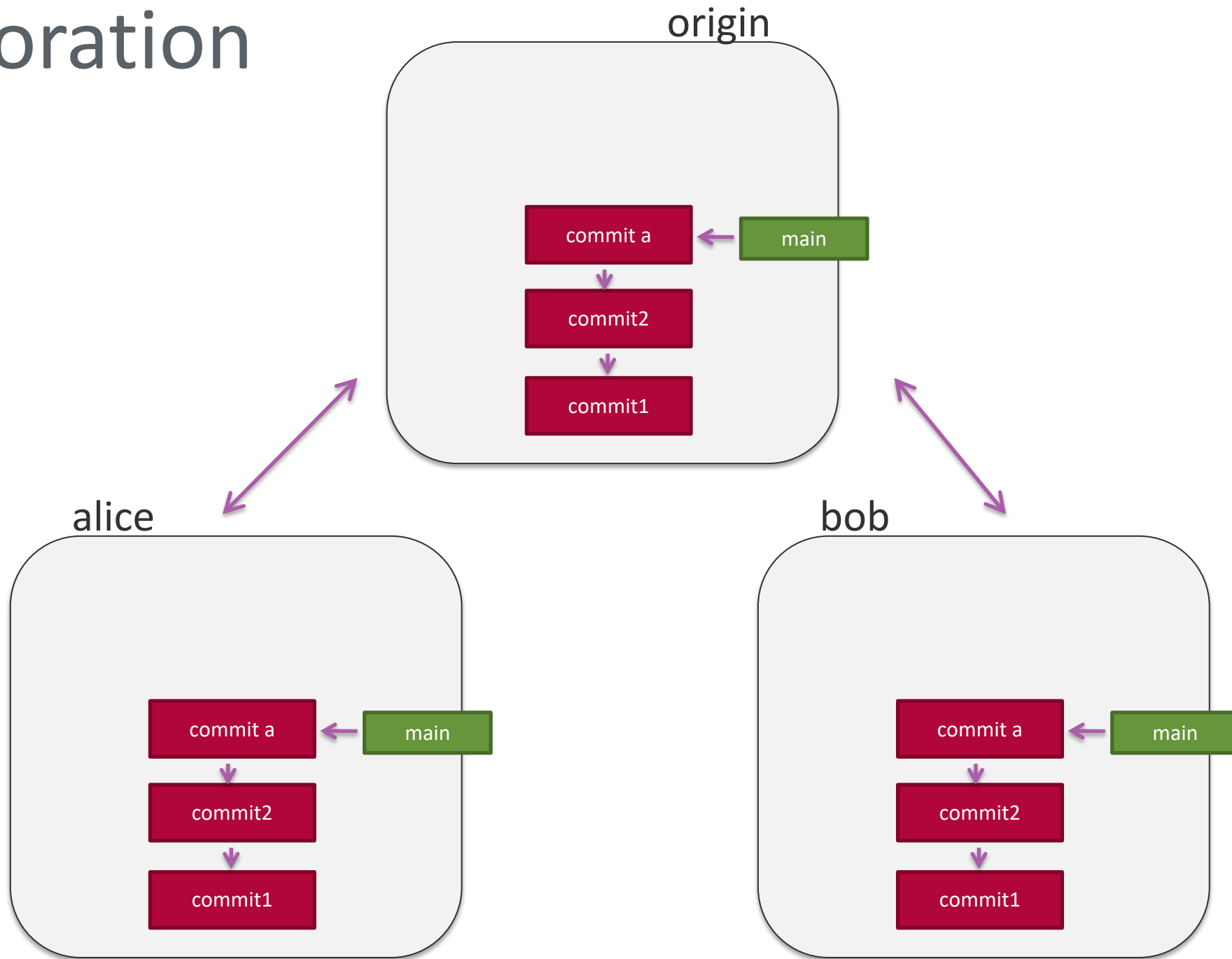


Collaboration using Git

Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

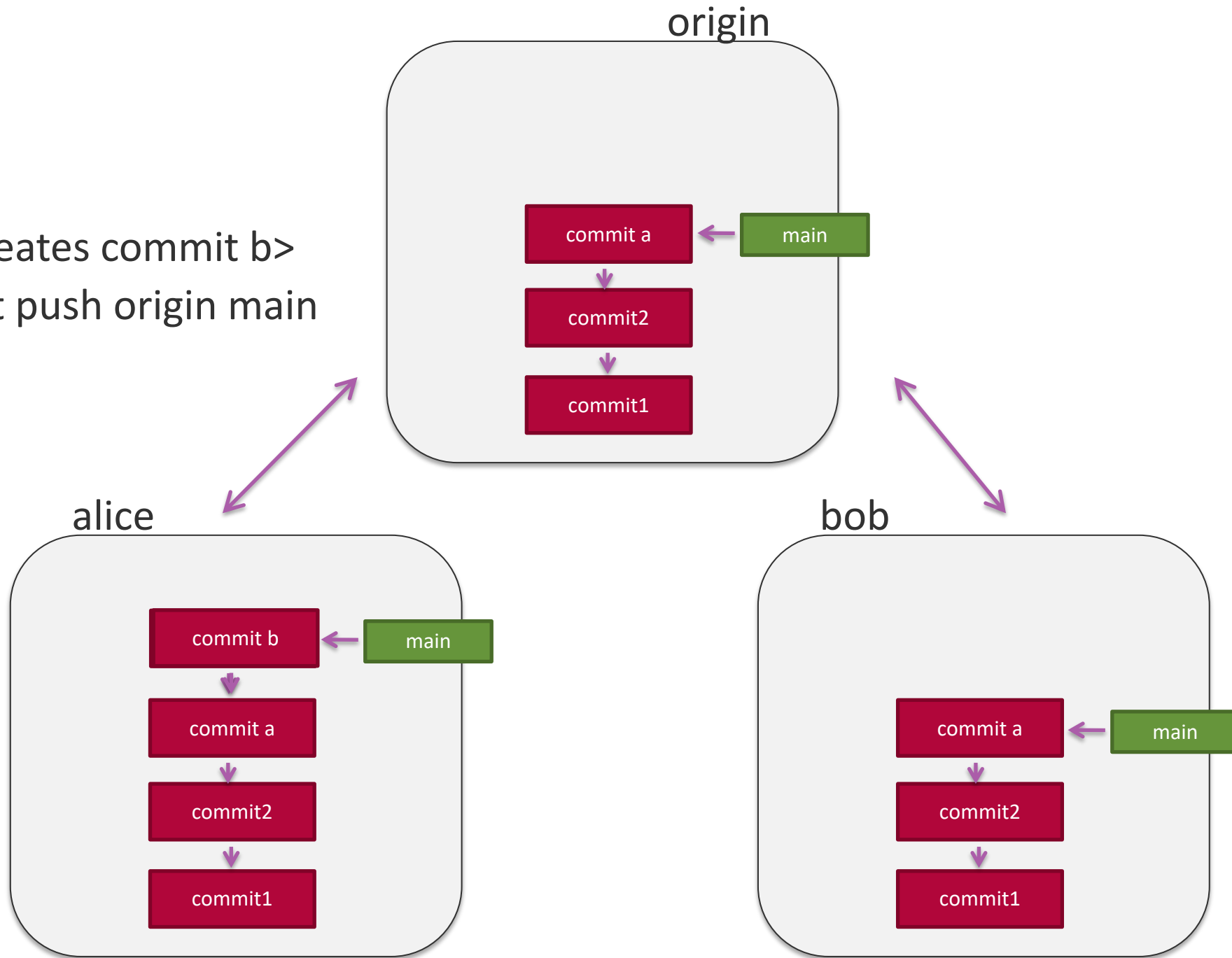
Collaboration



Push



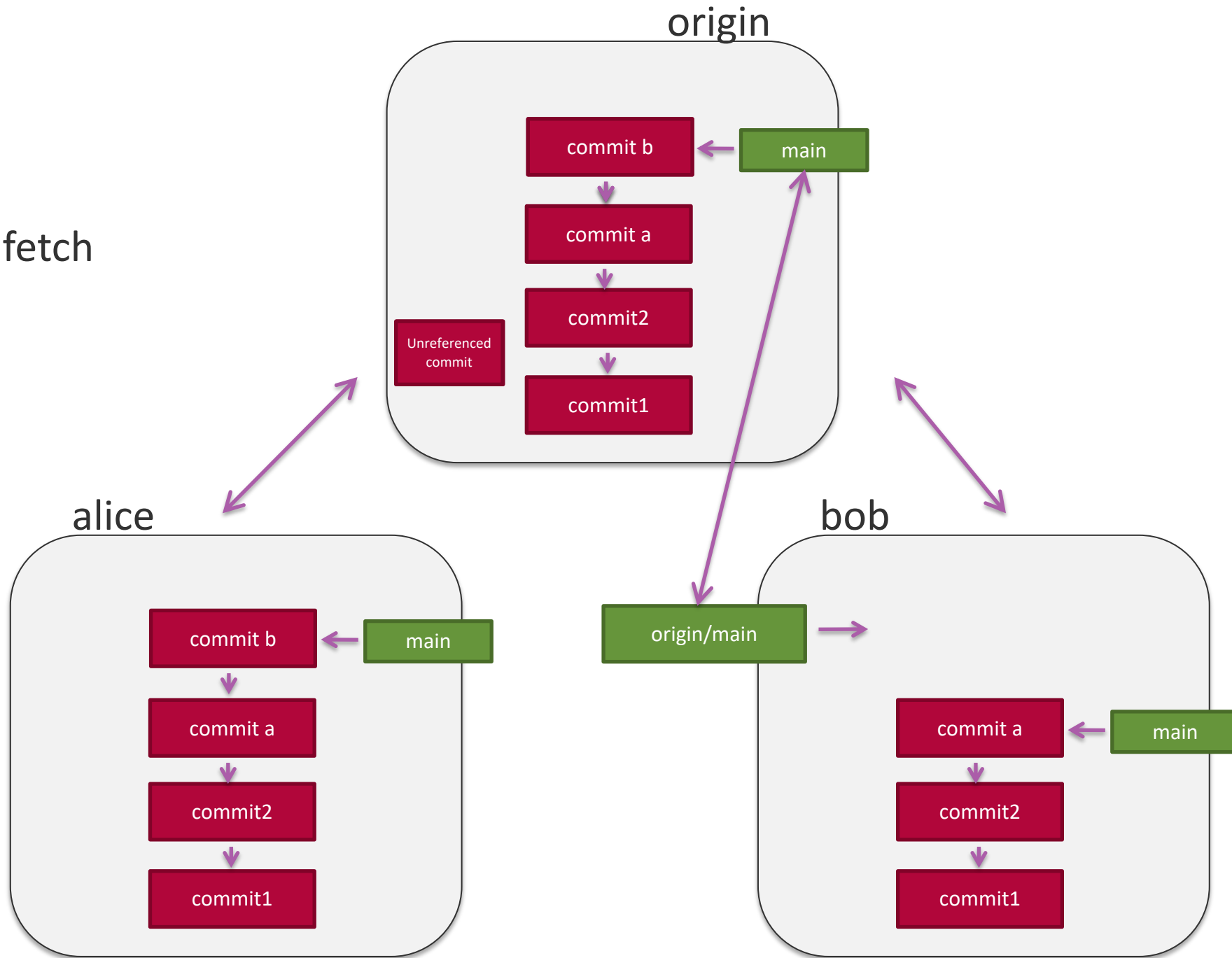
<alice creates commit b>
alice\$ git push origin main



Fetch



bob\$ git fetch

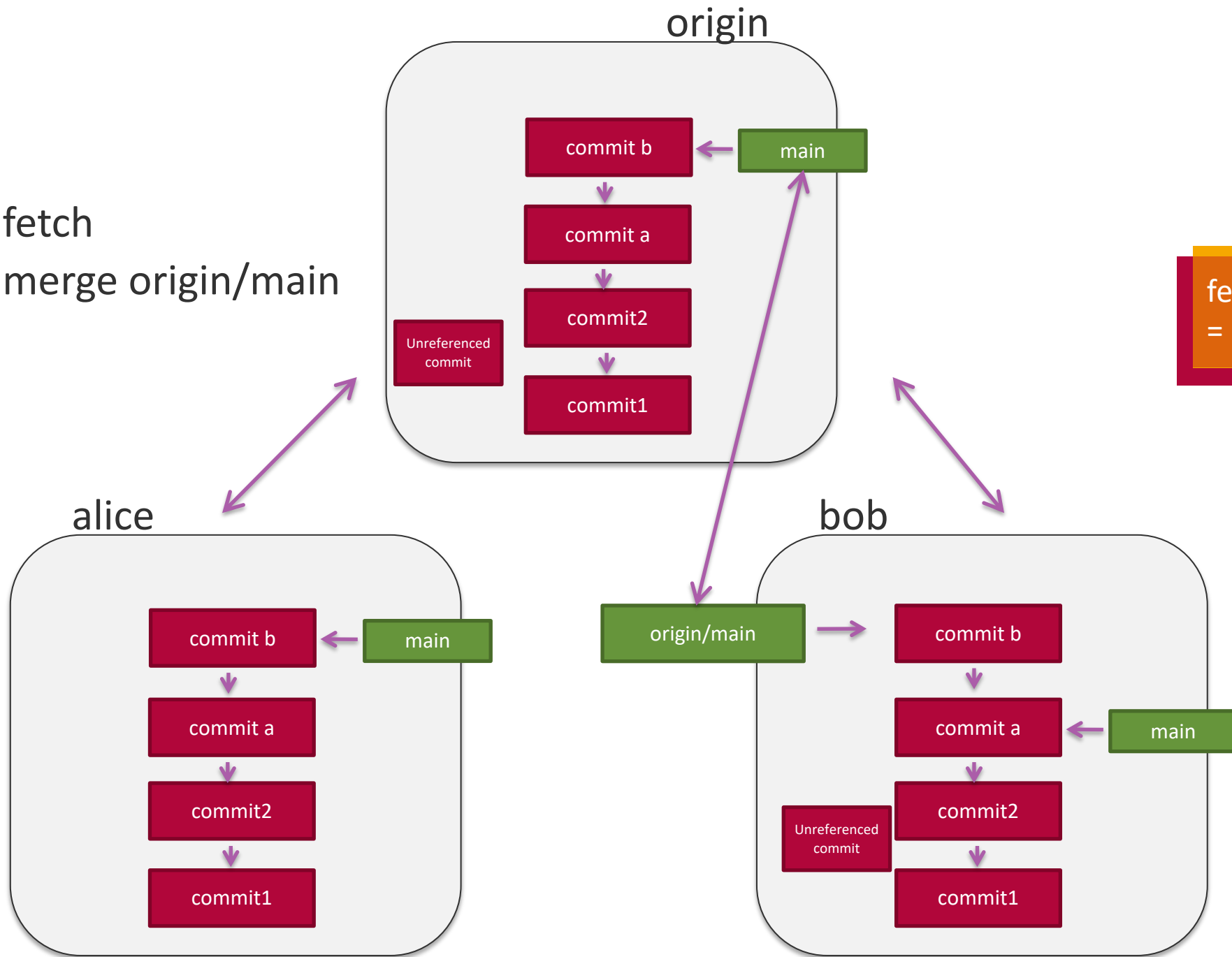


Pull



bob\$ git fetch
bob\$ git merge origin/main

fetch + merge
= pull

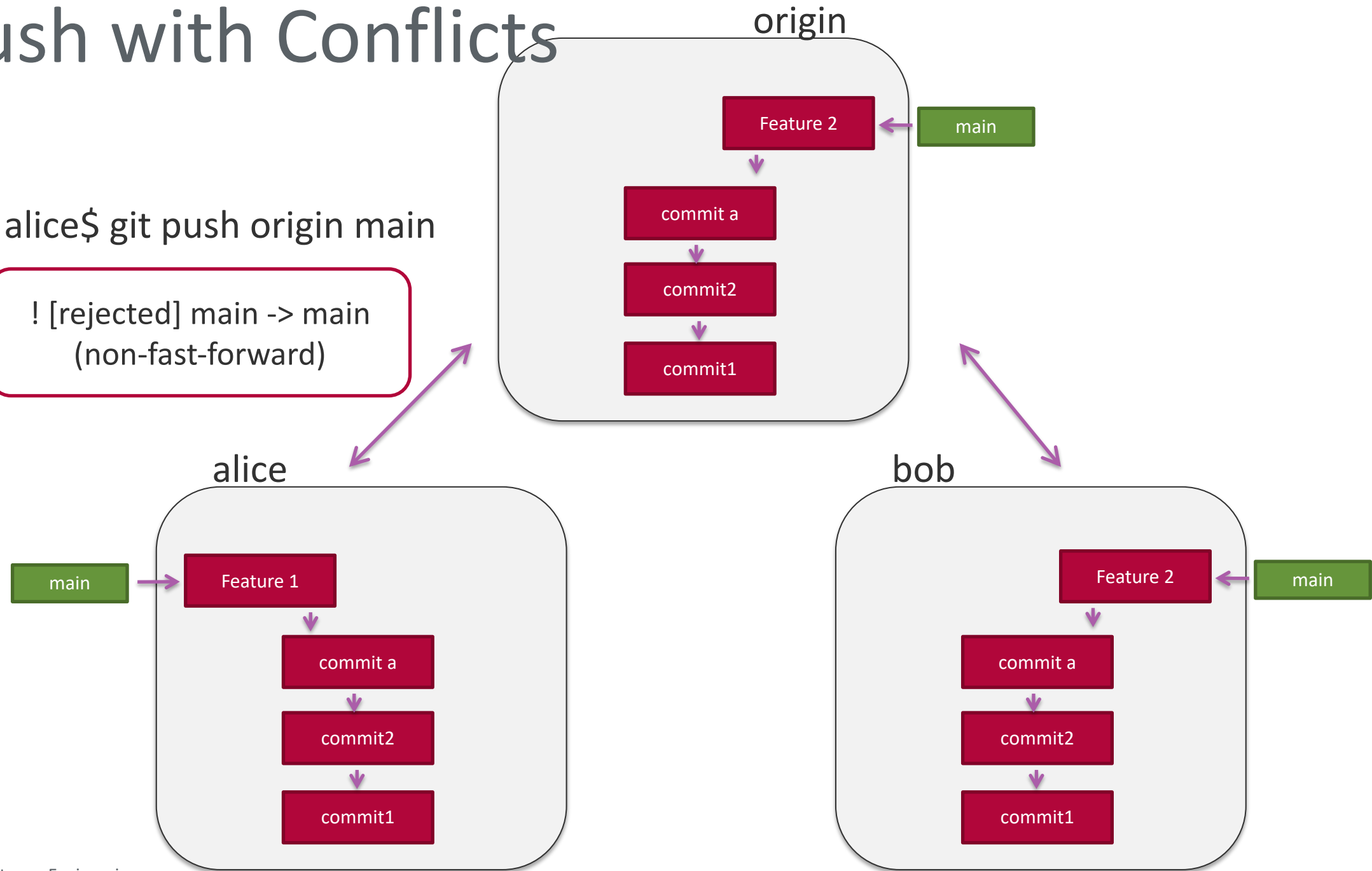


Push with Conflicts



alice\$ git push origin main

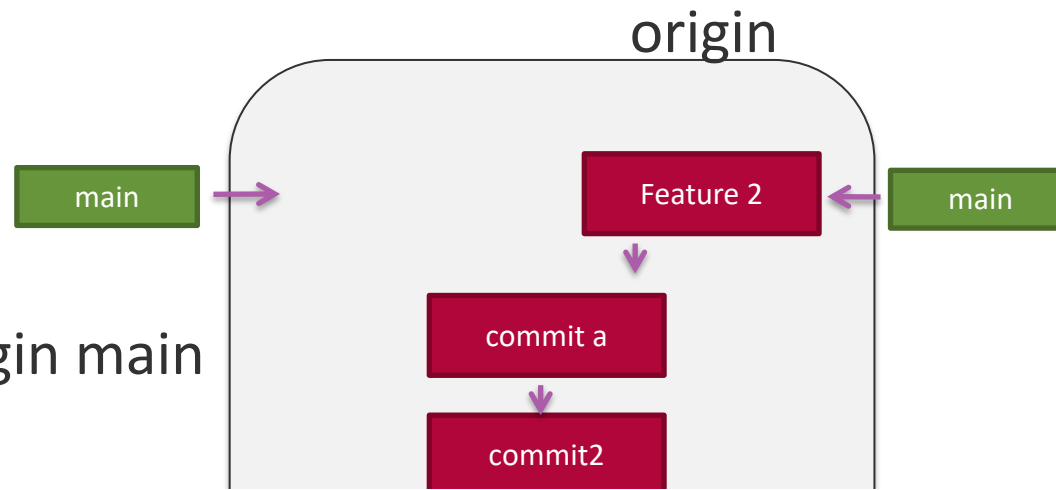
! [rejected] main -> main
(non-fast-forward)



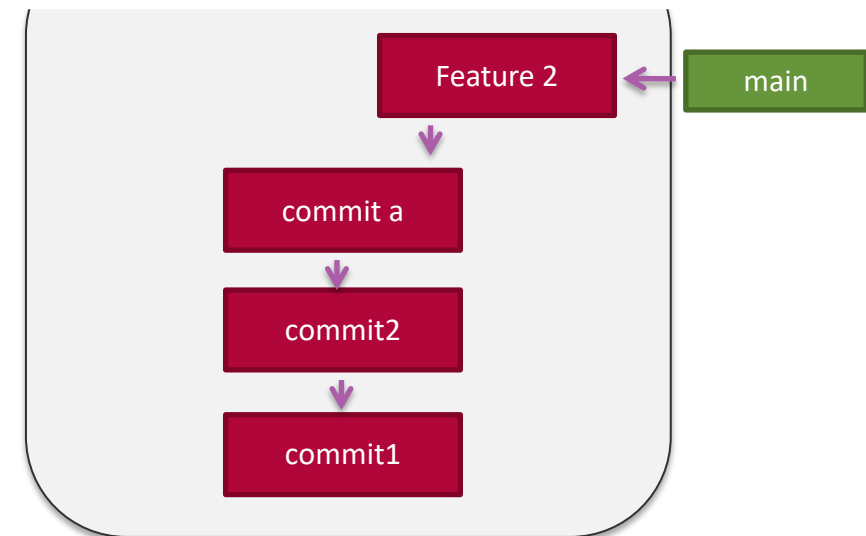
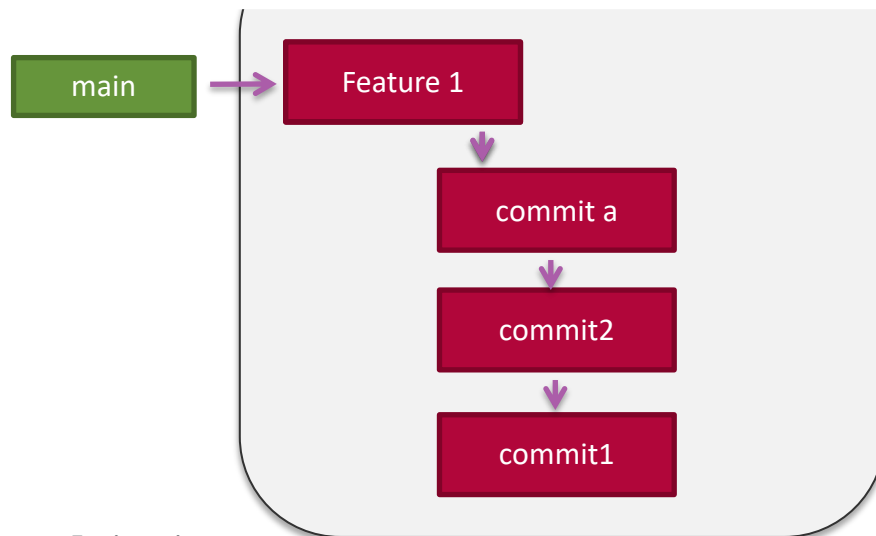
Push --force



alice\$ git push --f origin main



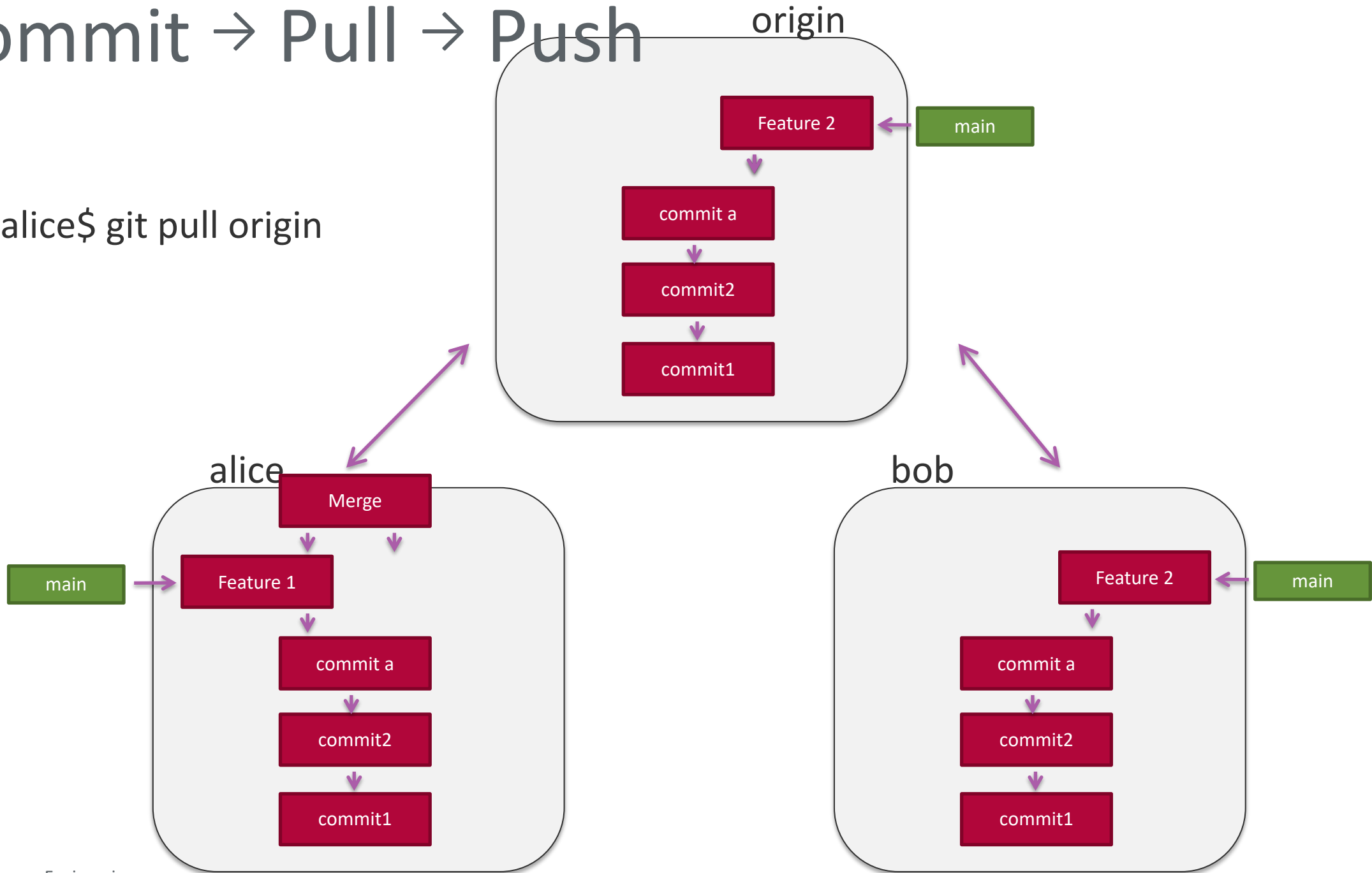
Never push --force



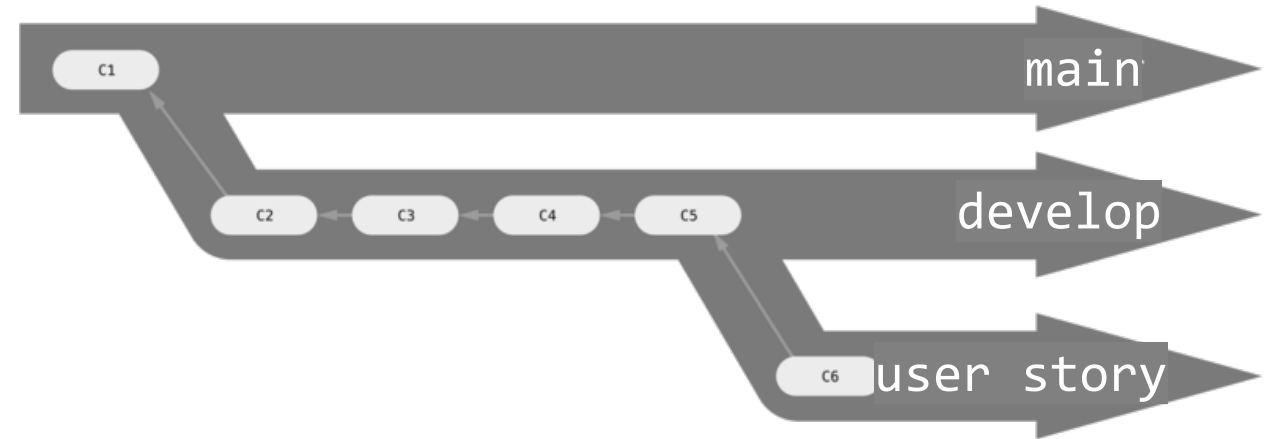
Commit → Pull → Push



alice\$ git pull origin



Branching Models



Structuring your code and development

- Many ways to structure branches
- Branching model should be agreed upon by participants
- **Merging efforts increase with increased code divergence**

Recommendations:

- Never merge in main or release branches
- Try to not commit unfinished/broken code in shared branches

Understanding Changes



■ git log

- List commits reachable by following the parent links from the current commit
- Make it visual: `git log --graph --decorate --oneline`

■ git diff

- Show line changes between commits
- Show changes in staged files: `git diff --cached`

■ git blame

- Show what revision and author last modified each line of a file

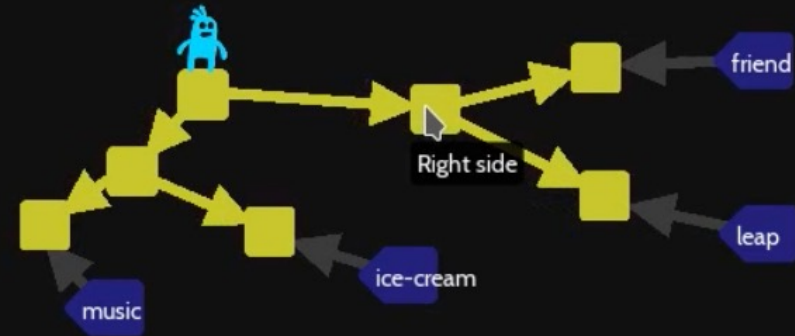


Learn some more

Learn & practice Git

```
Read the README.md for instructions or view them in browser:  
http://gitexercises.fracz.com/e/commit-one-file
```

```
~/exercises (commit-one-file u=)  
$ git status -s  
?? A.txt  
?? B.txt  
  
~/exercises (commit-one-file u=)  
$ git add A.txt
```



Interactive git exercises and training (go find some more), e.g.

- <https://gitexercises.fracz.com/>
- <https://github.com/benthayer/git-gud>
- <https://ohmygit.org>
- https://learngitbranching.js.org/?locale=de_DE

Summary



Types of VCS

- (De)centralized
- Usage
- (Dis)advantages

Git internals

- Git Objects
- File Lifecycle
- Detached Head
- Fetching & Merging

Collaboration

- Synchronizing repos
- Push –force
- Branching models
- Tracing Changes

