



Git Branches



Objectives



- ▶ Branches
- ▶ Merges
- ▶ Conflicts



1

Recap- Git Workflow



▶ Recap-What is Git?

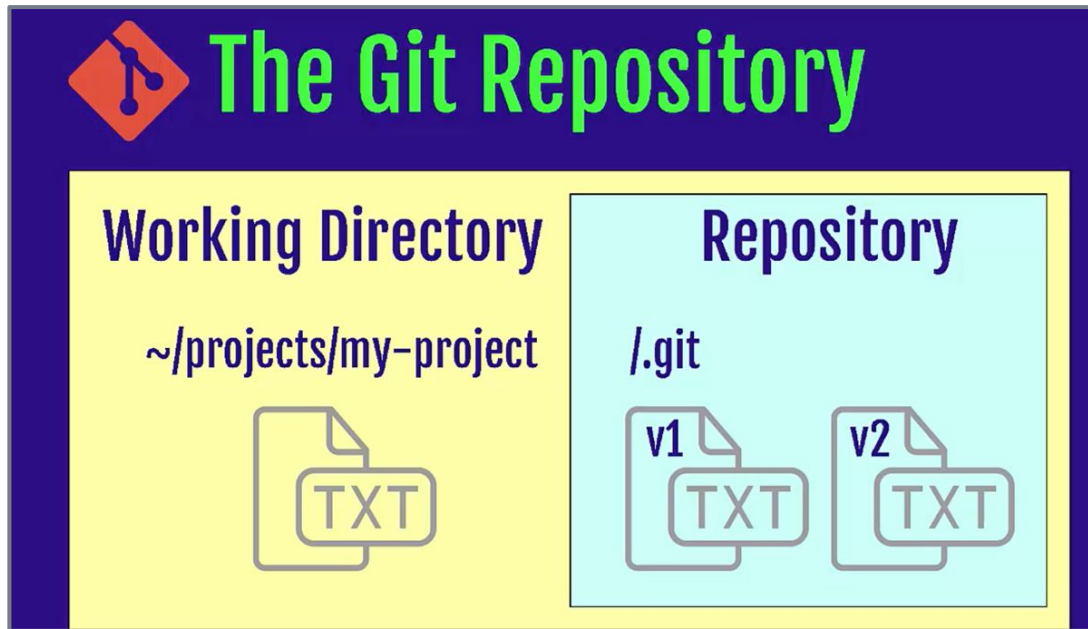
- **Git** is an **open source distributed version control system**
- **Tracks** and **records** changes to files over time (**versioning**)
- Can **retrieve** previous version of files at any time (**time travel**)
- Can be used **locally**, or **collaboratively** with others (**teamwork**)
- Contains extra information such as **date**, **author**, and **a message explaining the change**
- **Compare** and **Blame**
 - What changed
 - When it changed
 - Why it changed
 - Who changed it



Recap-Git Repository

What is a repository

- A directory or storage space where your projects can live.
- Local Repository
- Remote Repository (Central Repository)



Recap-Workflow-Git's "three trees"

Working Directory

Where you work. Create new files, edit files delete files etc.



Staging Area (Index)

Before taking a snapshot, you're taking the files to a stage. Ready files to be committed.



Repository (Commit Tree)

Committed snapshots of your project will be stored here with a full version history.





Recap-Git Config

→ Git needs your identity to mark/label changes / editor

```
git config --global user.name "Your Name"
```

```
git config --global user.email "Your Name"
```

```
git config --global core.editor "vim"
```

```
git config --list
```



Recap-Basic Commands

git help

git init

git status

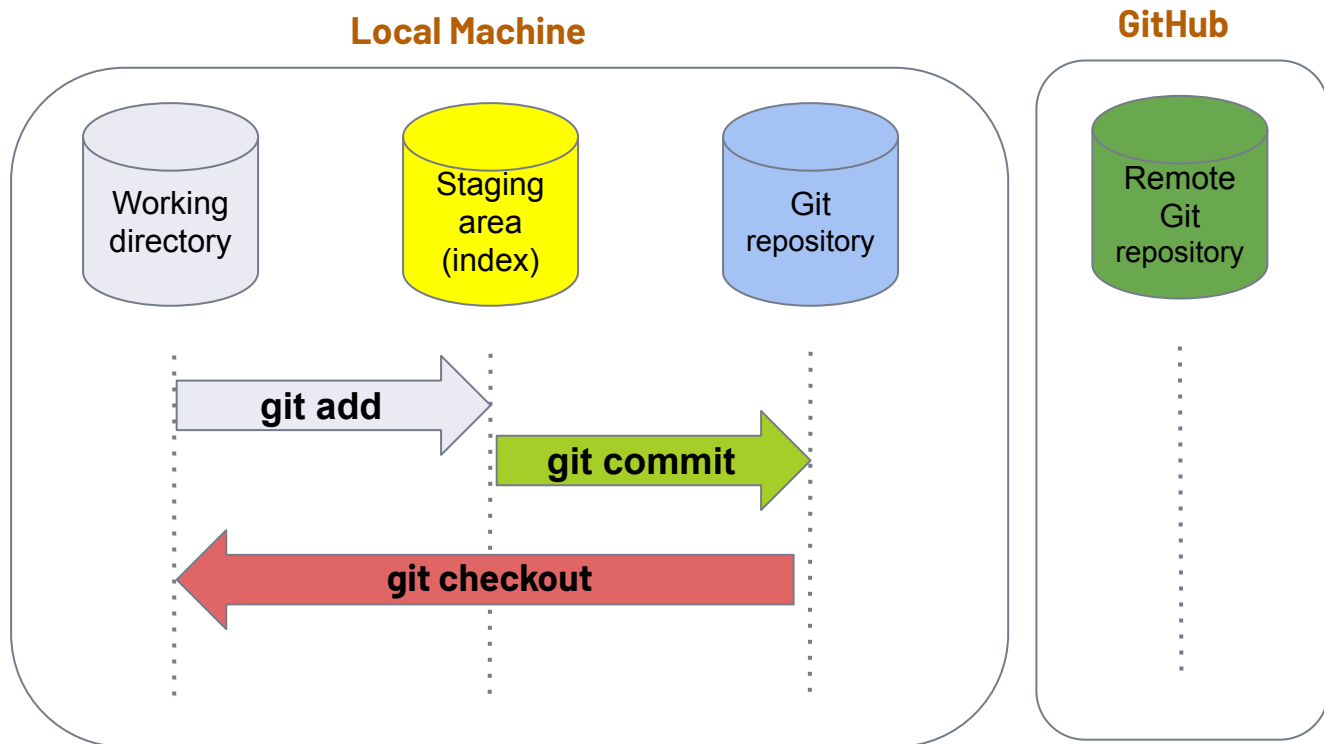
git add .

git rm --cached

git commit -m "abc"

git log

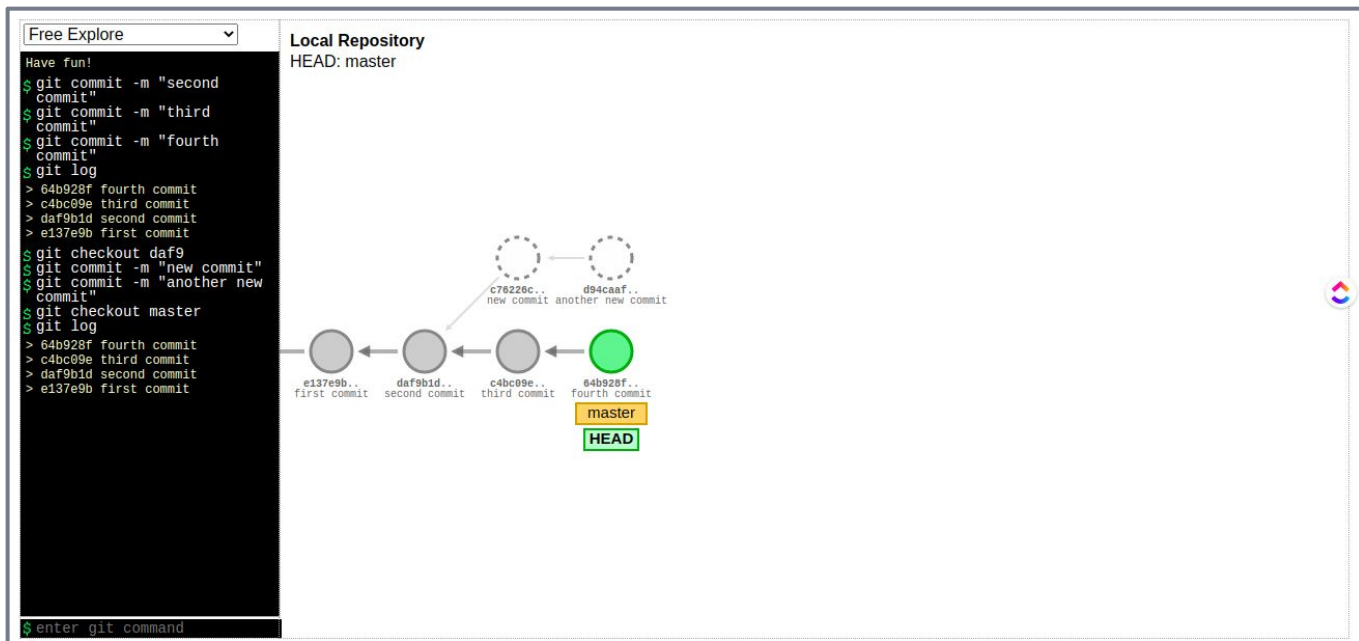
git checkout **commitID**





Recap-Timetravel

<https://git-school.github.io/visualizing-git/>





Recap-Tasks

Task-1



- Create a new repo under **my-second-project** folder
- Create a file named **file1.txt**
- Change the file
- Stage the file
- Commit the file to your repo

Task-2



- Create a file named **file2.txt**
- Edit **file2.txt**
- Stage
- Delete the file **file1.txt**
- Rename **file2.txt** >> **file3.txt**
- Stage **file3.txt**
- Unstage **file3.txt**
- Stage **file3.txt** again
- Commit the file to your repo
- Change the message of the commit
- Switch back to your first commit in **Task-1**





Recap-Solutions

- Create a new repo under **my-second-project** folder
- Create a file named **file1.txt**
- Change the file
- Stage the file
- Commit the file to your repo
- Create a file named **file2.txt**
- Edit **file2.txt**
- Stage
- Delete the file **file1.txt**
- Rename **file2.txt** >> **file3.txt**

```
git init
```

```
touch file1.txt
```

```
vim file1.txt
```

```
git add .
```

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

```
touch file2.txt
```

```
vim file2.txt
```

```
git add .
```

```
rm file1.txt
```

```
mv file2.txt file3.txt
```



Recap-Solutions Cntd.

- Stage **file3.txt**
- Unstage **file3.txt**
- Stage **file3.txt** again
- Commit the file to your repo
- Change the message of the commit
- Switch back to your first commit in **Task-1**

```
git add .
```

```
git rm --cached file3.txt
```

```
git add .
```

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

```
git commit --amend
```

```
git log
```

```
git checkout "first commit ID"
```



Branch, Head

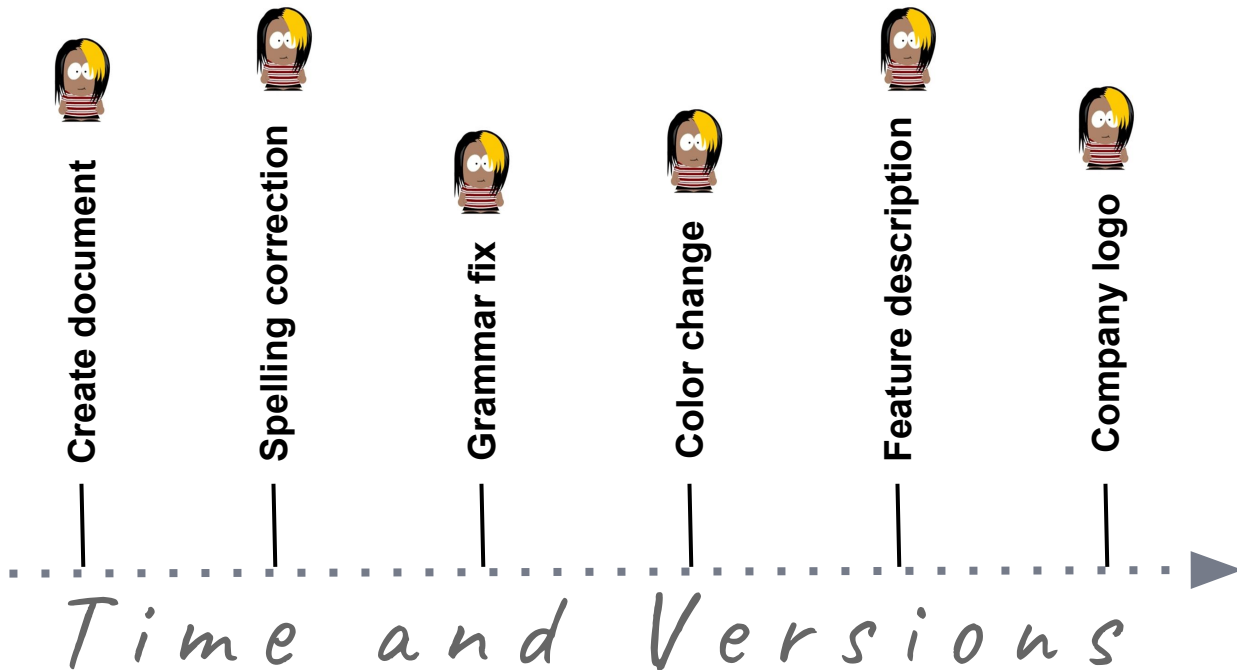
What comes to you your mind when you hear this?





Git Branches

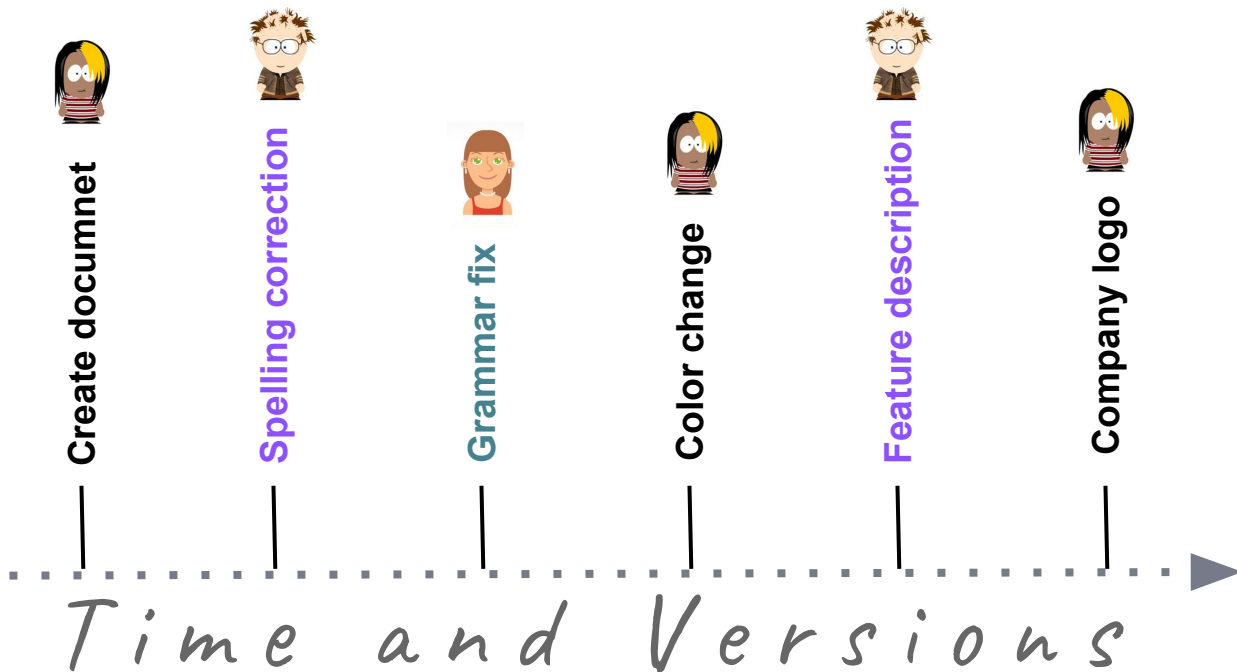
History Tracking





Git Branches

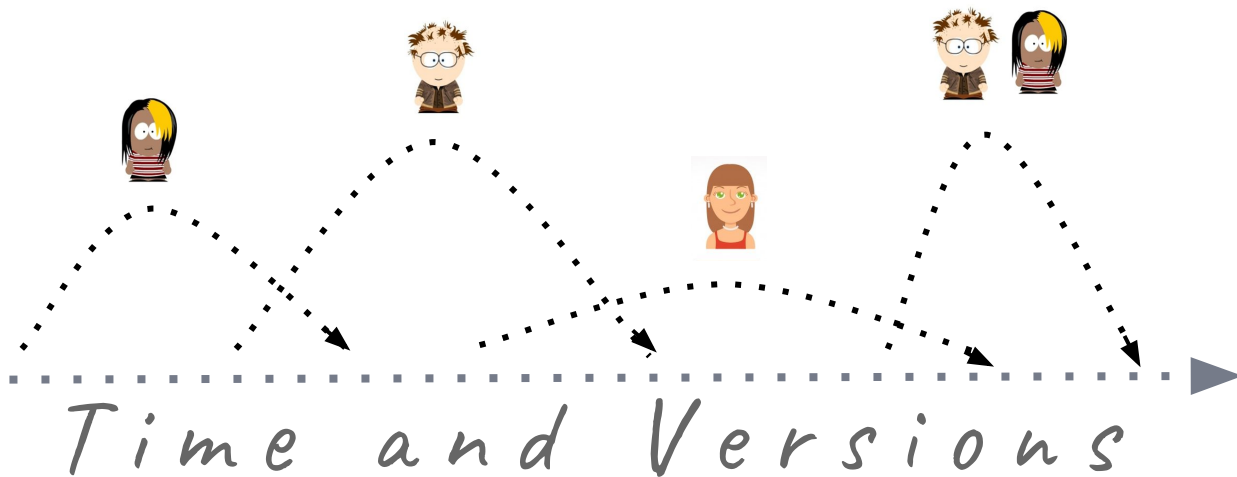
Collaborative History Tracking





Git Branches

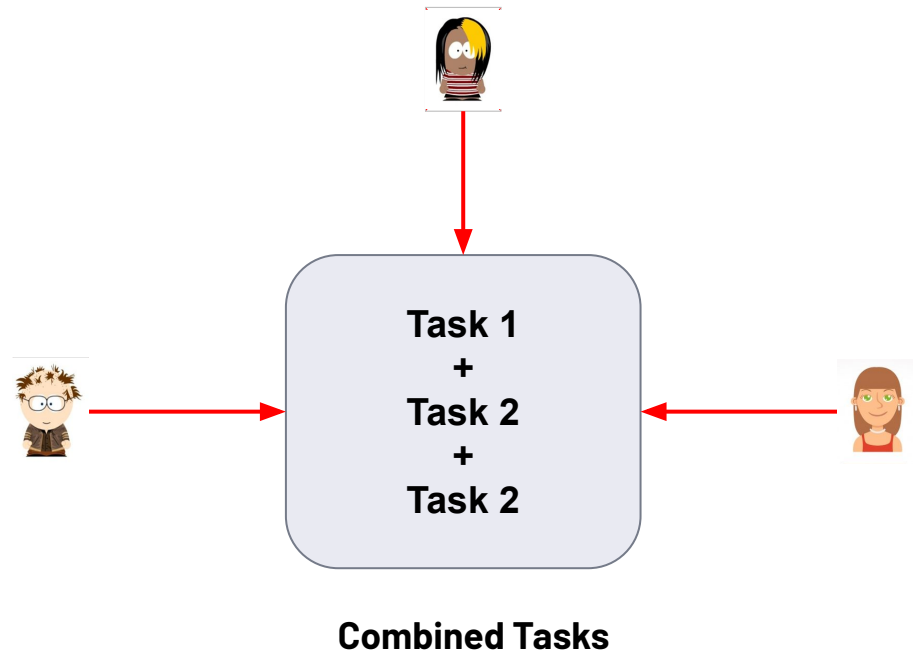
Collaborative History Tracking





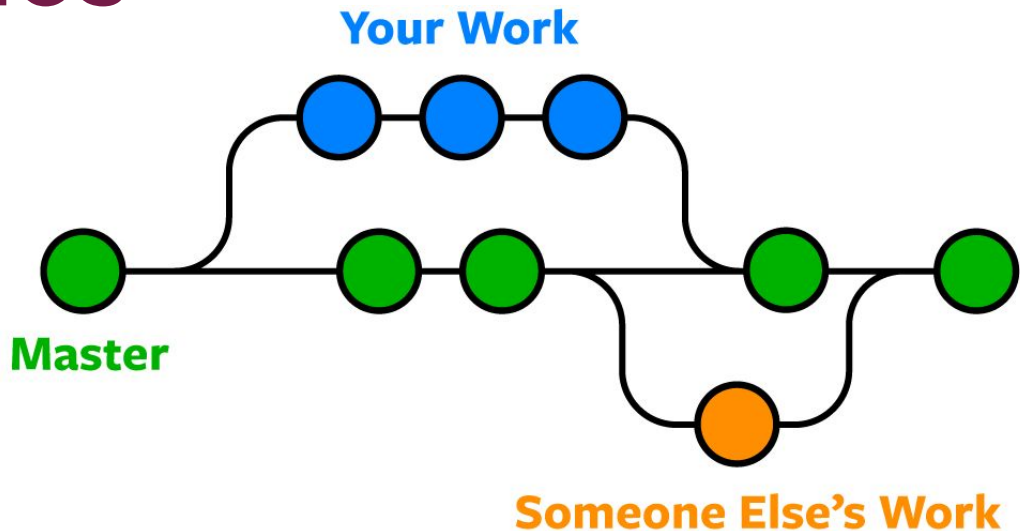
Git Branches

Collaboration





Branches



- Production of the project lives on master/main branch
- Branches are reference to a commit

```
Eric's-Mac:project eric$ git branch
* master
```



Branches

→ to see local branches

```
git branch
```

→ to see remote branches

```
git branch -r
```

→ to see all branches

```
git branch -a
```



Creating/switching branches

- create a new branch

```
git branch Branch name
```

- switch to a branch

```
git checkout Branch name
```

- create a new branch and switch to that branch

```
git checkout -b Branch name
```



Deleting branches

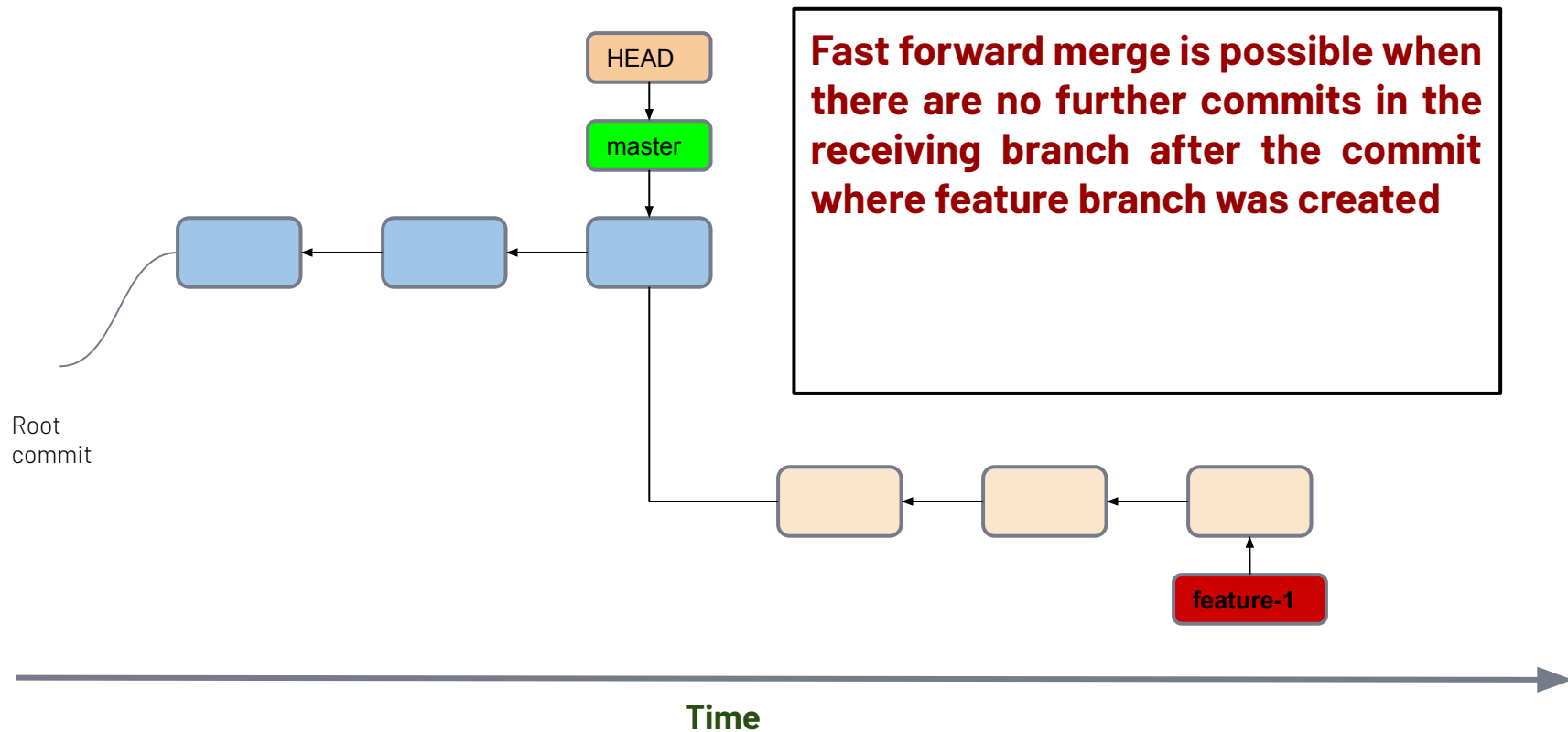
→ delete a local branch

```
git branch -d Branch name
```

```
git branch -D Branch name
```



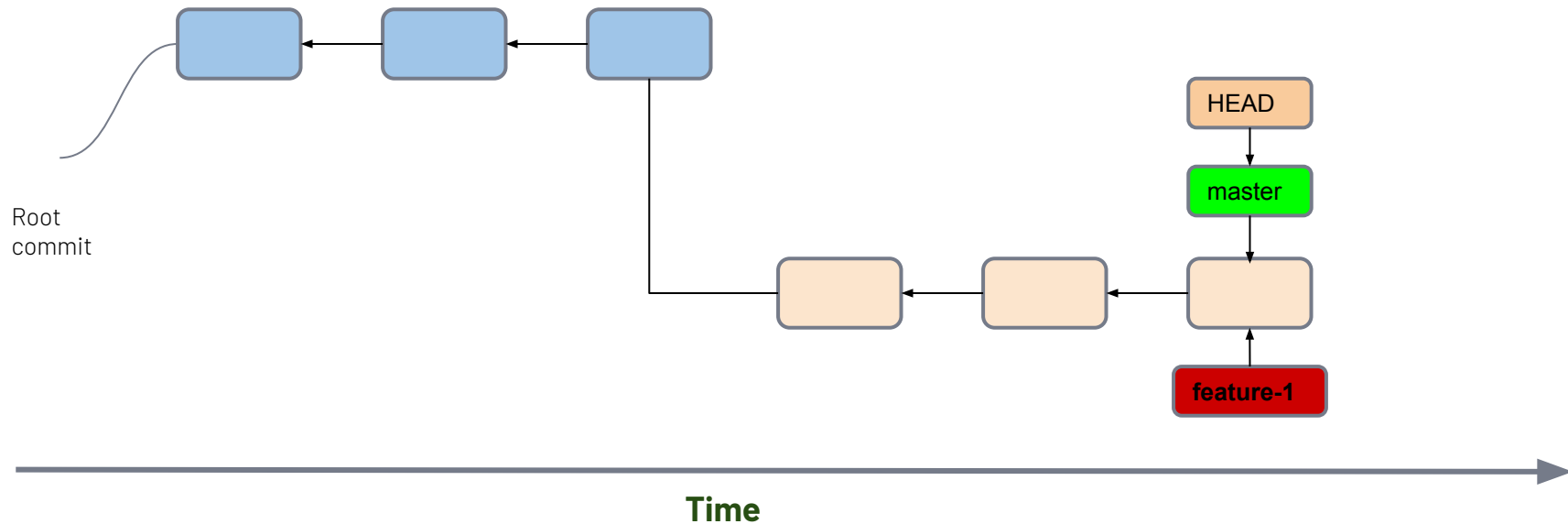
Fast forward merge





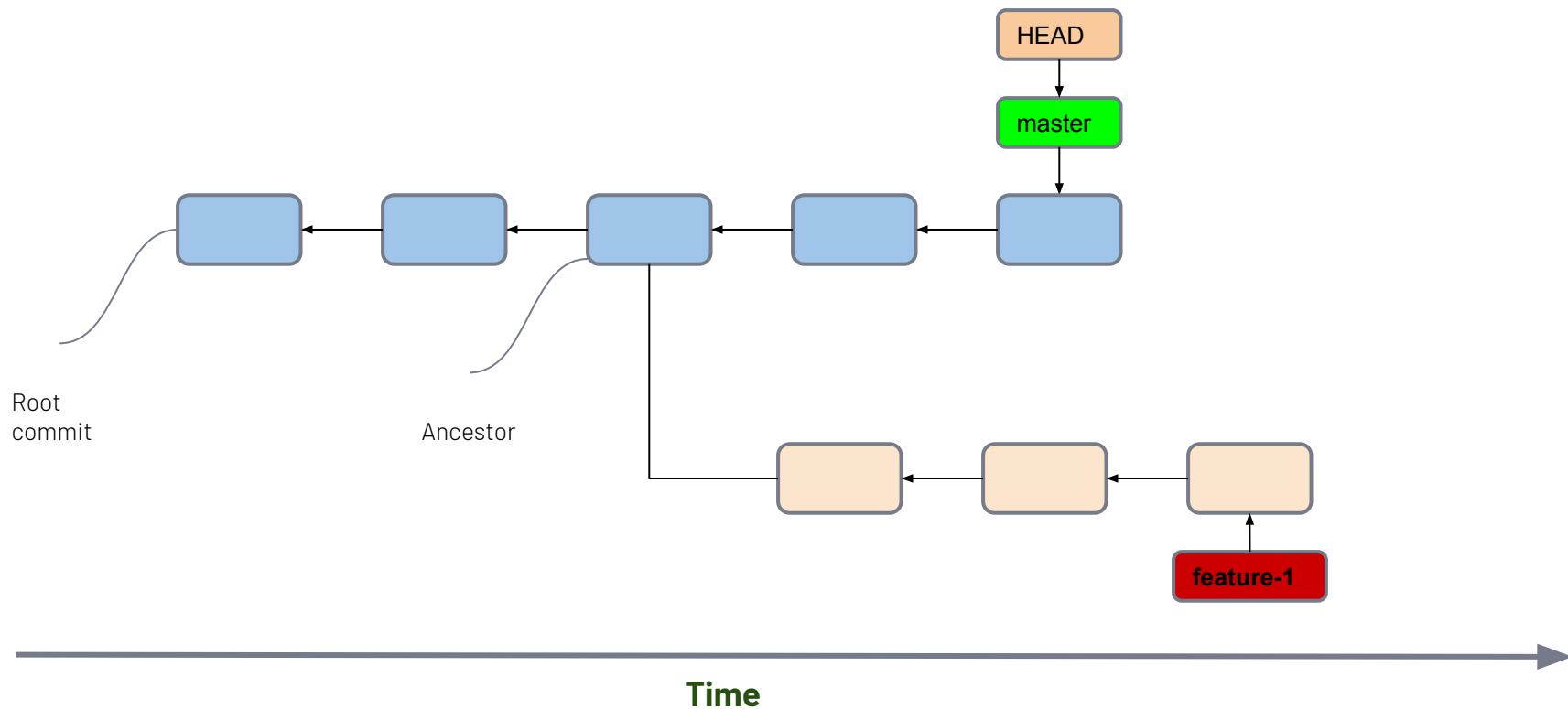
Fast forward merge

git merge <feature-branch>



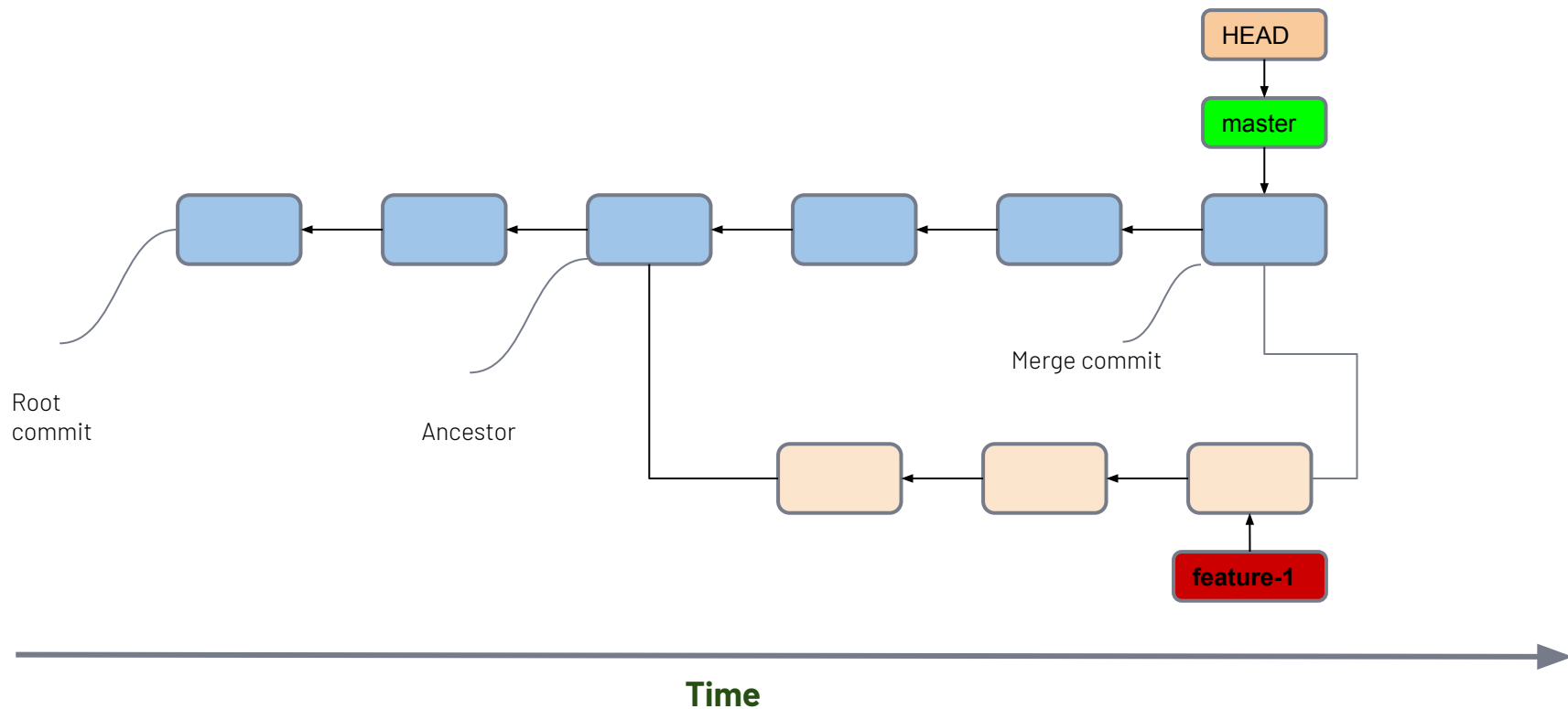


3-way merge





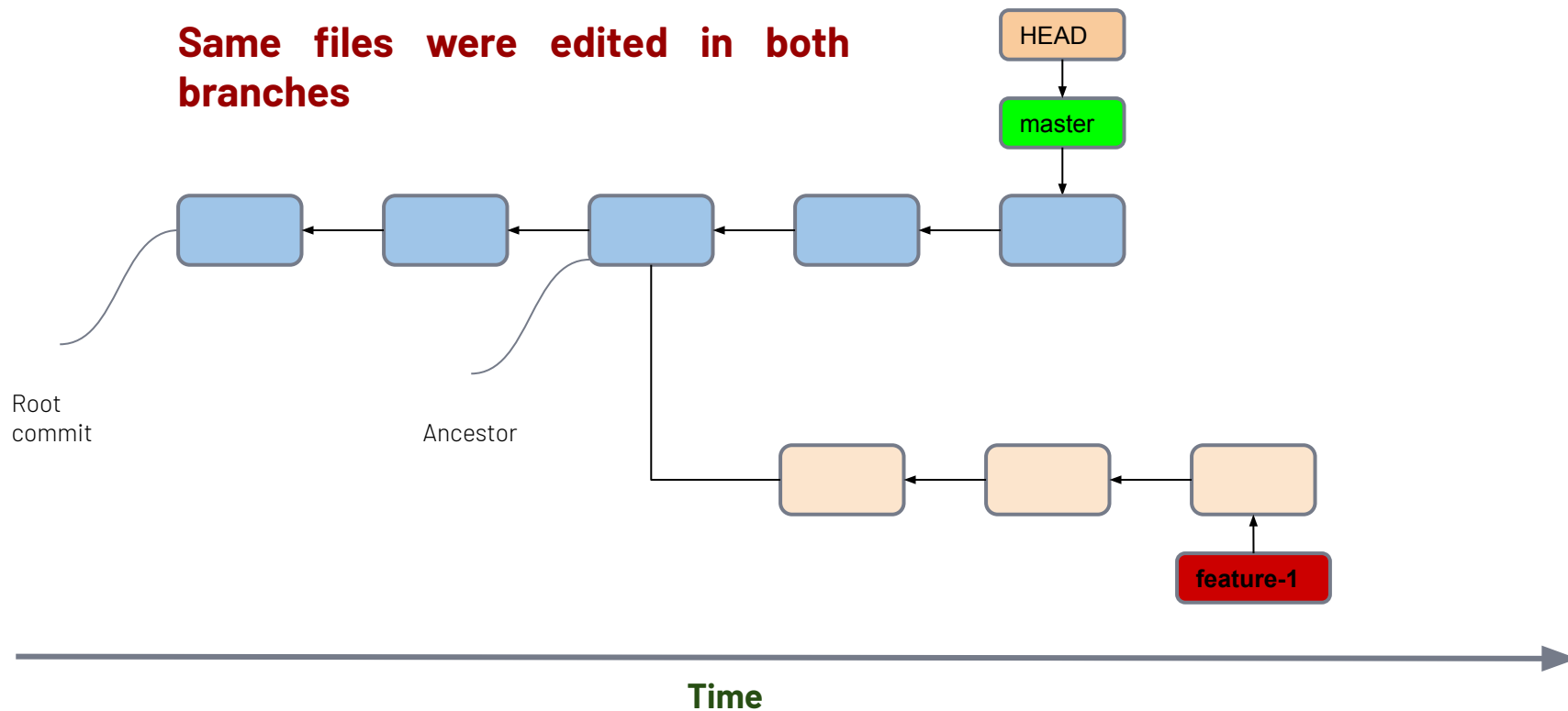
3-way merge





Merge Conflicts

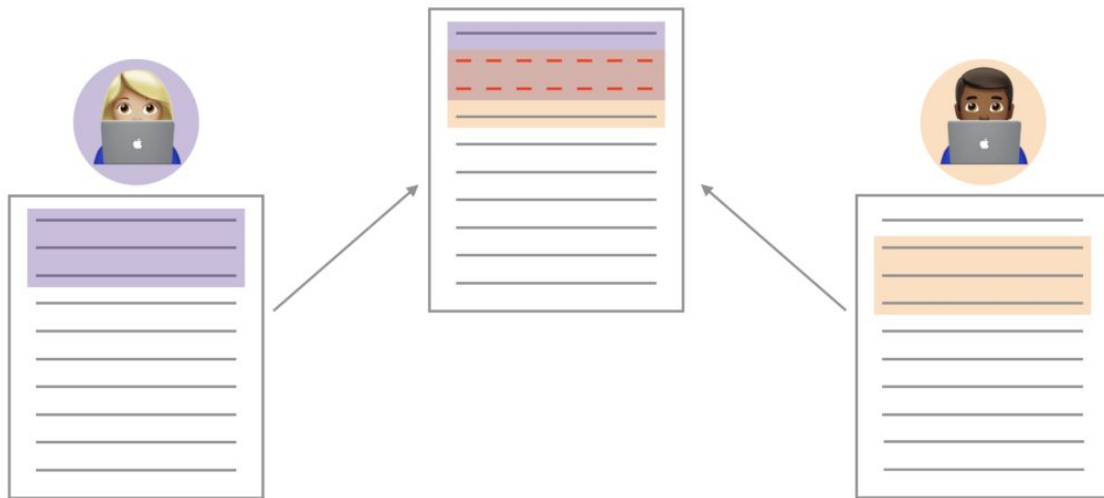
Same files were edited in both branches





Github - Merge Conflict

- **Merge conflicts** happen when you merge branches that have competing commits, and Git needs your help to decide which changes to incorporate in the final merge.





THANKS!

Any questions?

You can find me at:

- ▶ martin_fade@clarusway.com
- ▶ tyler@clarusway.com

