Git Branches

CLARUSWAY©
WAY TO REINVENT YOURSELF

Objectives

- Branches
- Merges
- Conflicts

1

Recap- Git Workflow

Recap-What is Git?

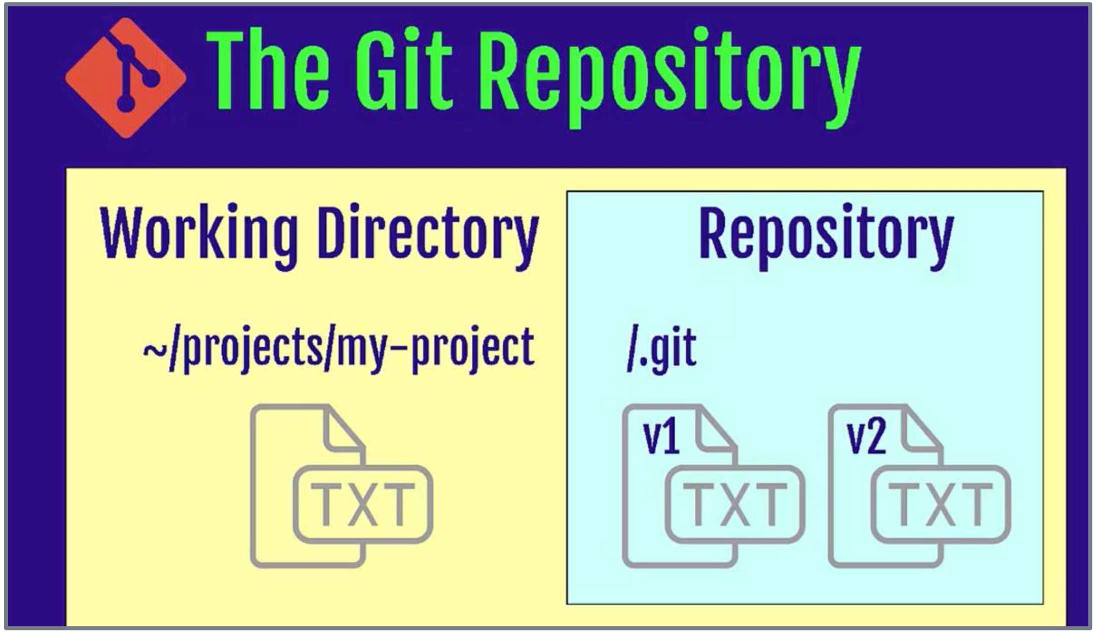
- **Git** is an **open source distributed version control system**
- **Tracks** and **records** changes to files over time (**versioning**)
- Can **retrieve** previous version of files at any time (**time travel**)
- Can be used **locally**, or **collaboratively** with others (**teamwork**)
- Contains 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-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-Workflow-Git’s “three trees”

Working Directory

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

Staging Area (Index)

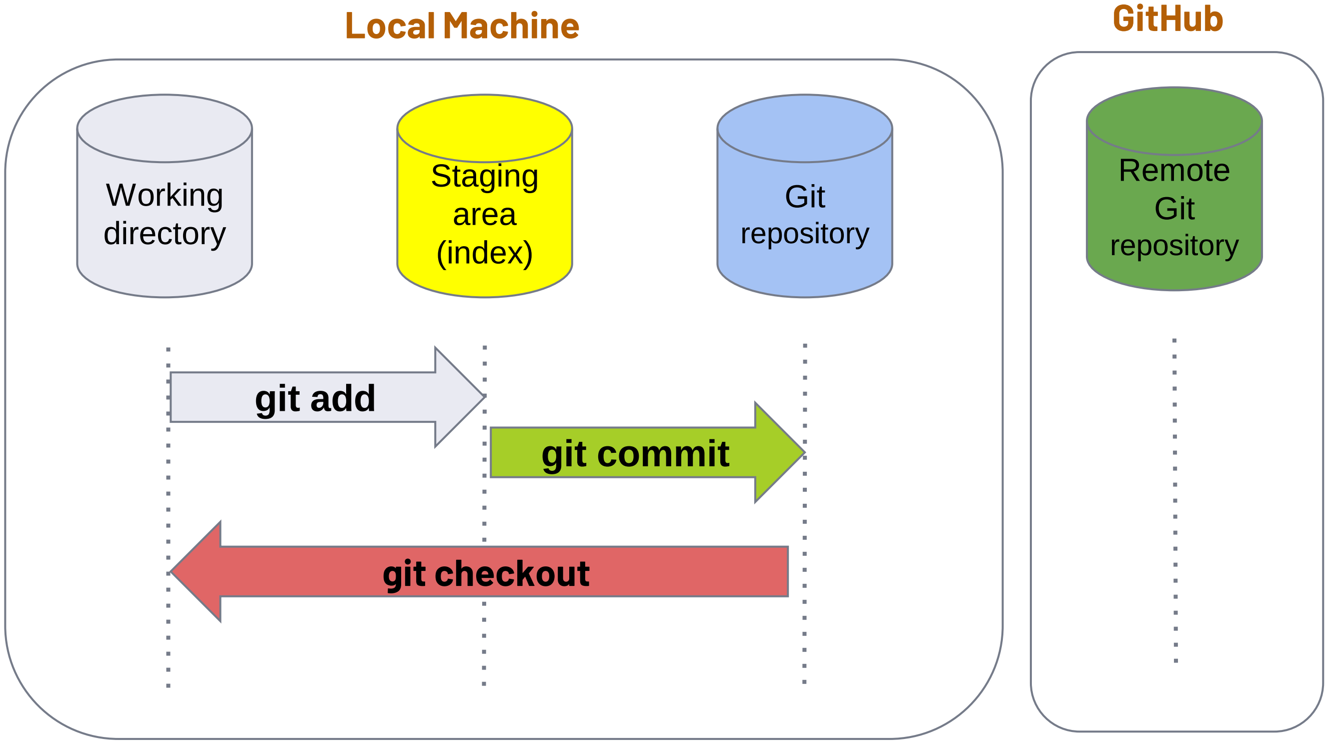
Before taking a snapshot, you’re taking the files to a stage. Ready files to be committed.

Repository (Commit Tree)

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

Recap-Basic Commands

```
git help
git init
git status
git add .
git rm --cached
git commit -m “abc”
git log
git checkout commitID
```



CLARUSWAY©
WAY TO REINVENT YOURSELF

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	git init
Create a file named **file1.txt**	touch **file1.txt**
Change the file	vim **file1.txt**
Stage the file	git add .
Commit the file to your repo	git commit -m **“message”**
Create a file named **file2.txt**	touch **file2.txt**
Edit **file2.txt**	vim **file2.txt**
Stage	git add .
Delete the file **file1.txt**	rm **file1.txt**
Rename **file2.txt** >> **file3.txt**	mv **file2.txt file3.txt**

Recap-Solutions Cntd.

→ Stage **file3.txt**	`git add .`
→ Unstage **file3.txt**	`git rm --cached file3.txt`
→ Stage **file3.txt** again	`git add .`
→ Commit the file to your repo	`git commit -m “message”`
→ Change the message of the commit	`git commit --amend`

→ Switch back to your first commit in **Task-1**	`git log`
	`git checkout “first commit ID”`

Git

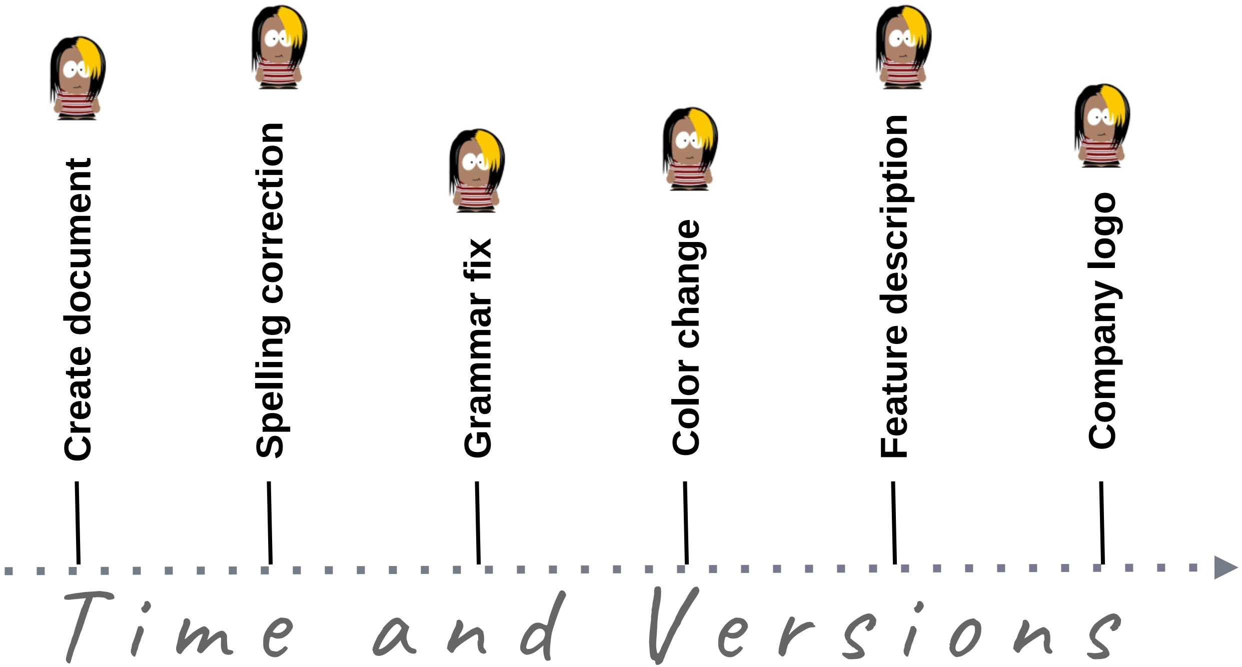
Branch, Head

What comes to you your mind when you hear this?

Students, write your response!

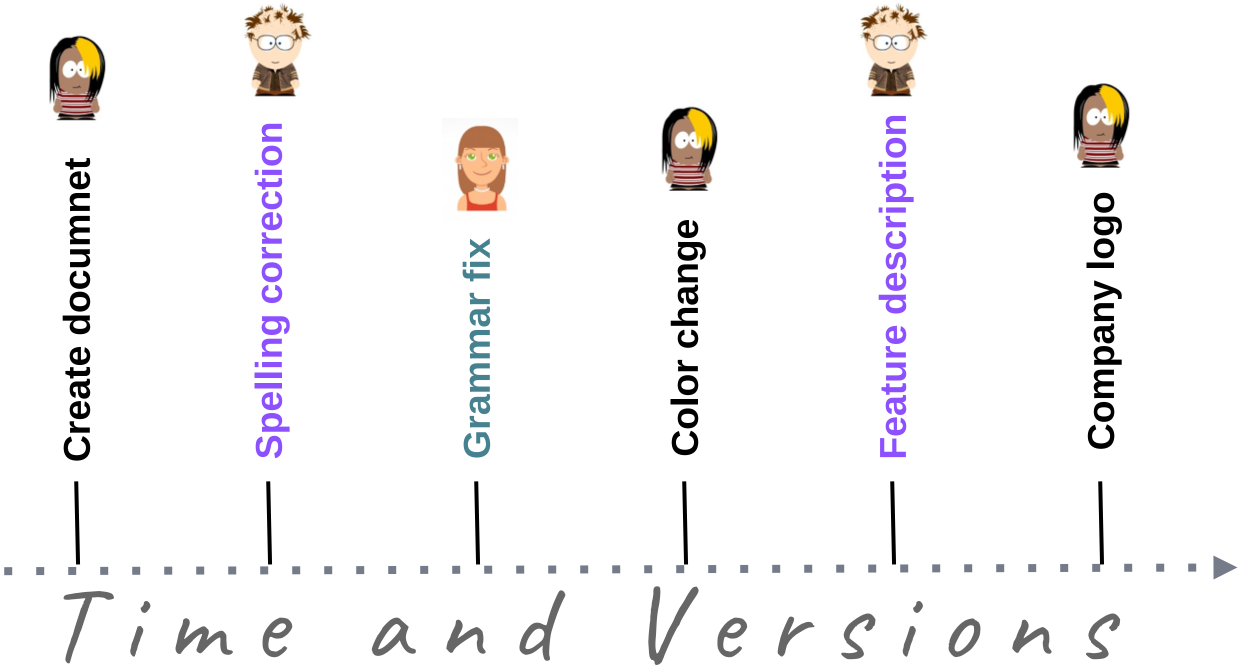
Git Branches

History Tracking



Git Branches

Collaborative History Tracking



Create documnet

Spelling correction

Grammar fix

Color change

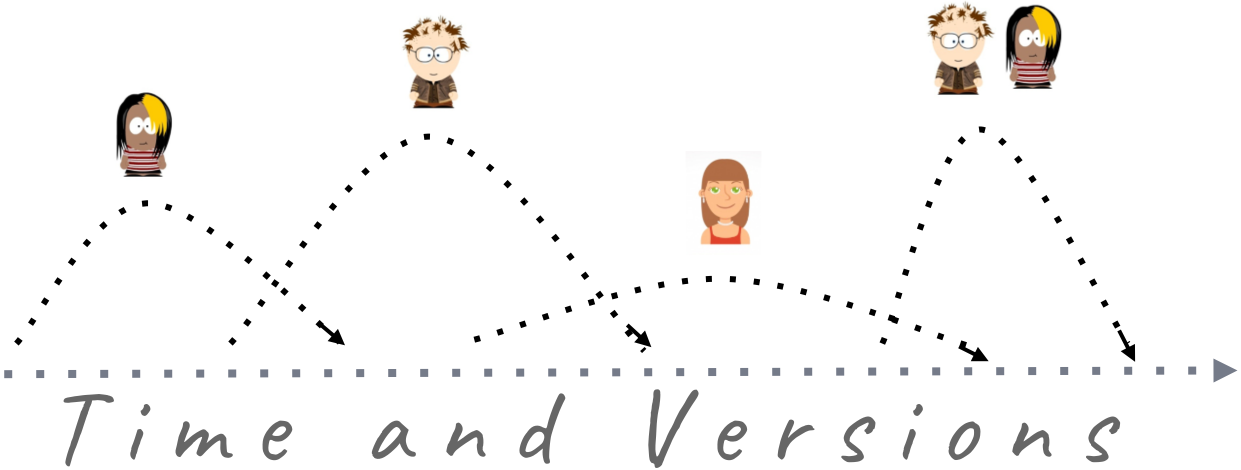
Feature description

Company logo

Time and Versions

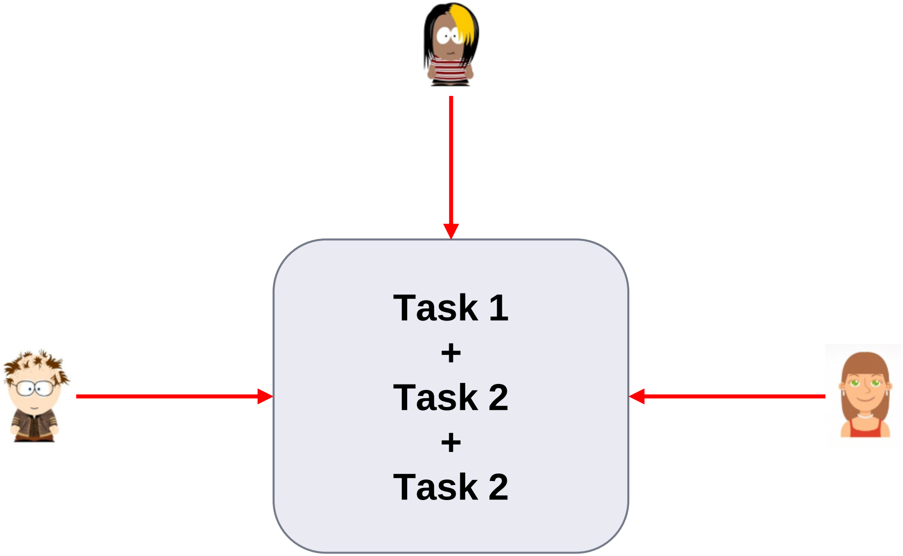
Git Branches

Collaborative History Tracking



Git Branches

Collaboration

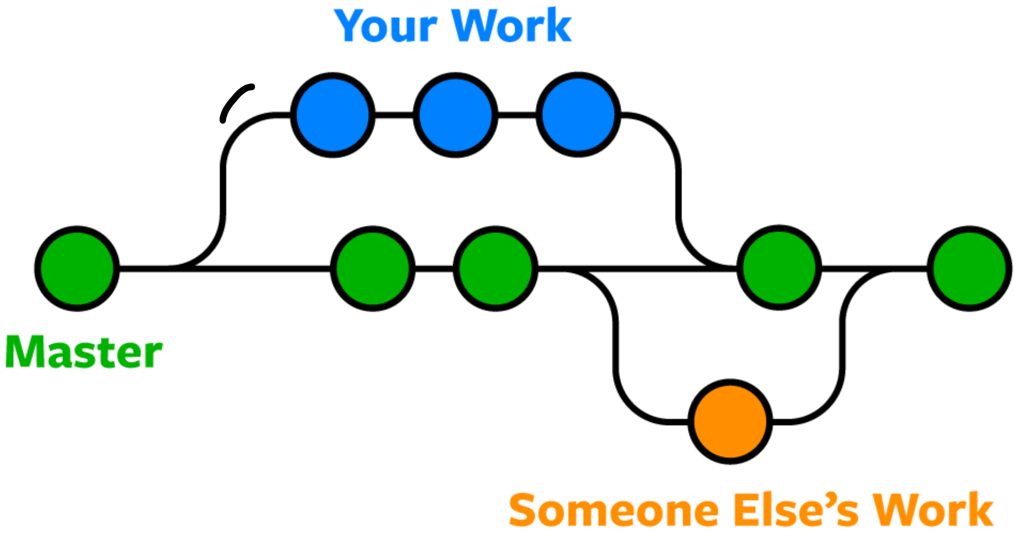


Task 1
+
Task 2
+
Task 2

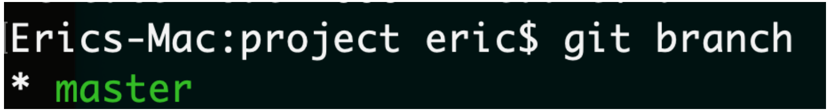
Combined Tasks

CLARUSWAY©
WAY TO REINVENT YOURSELF

Branches

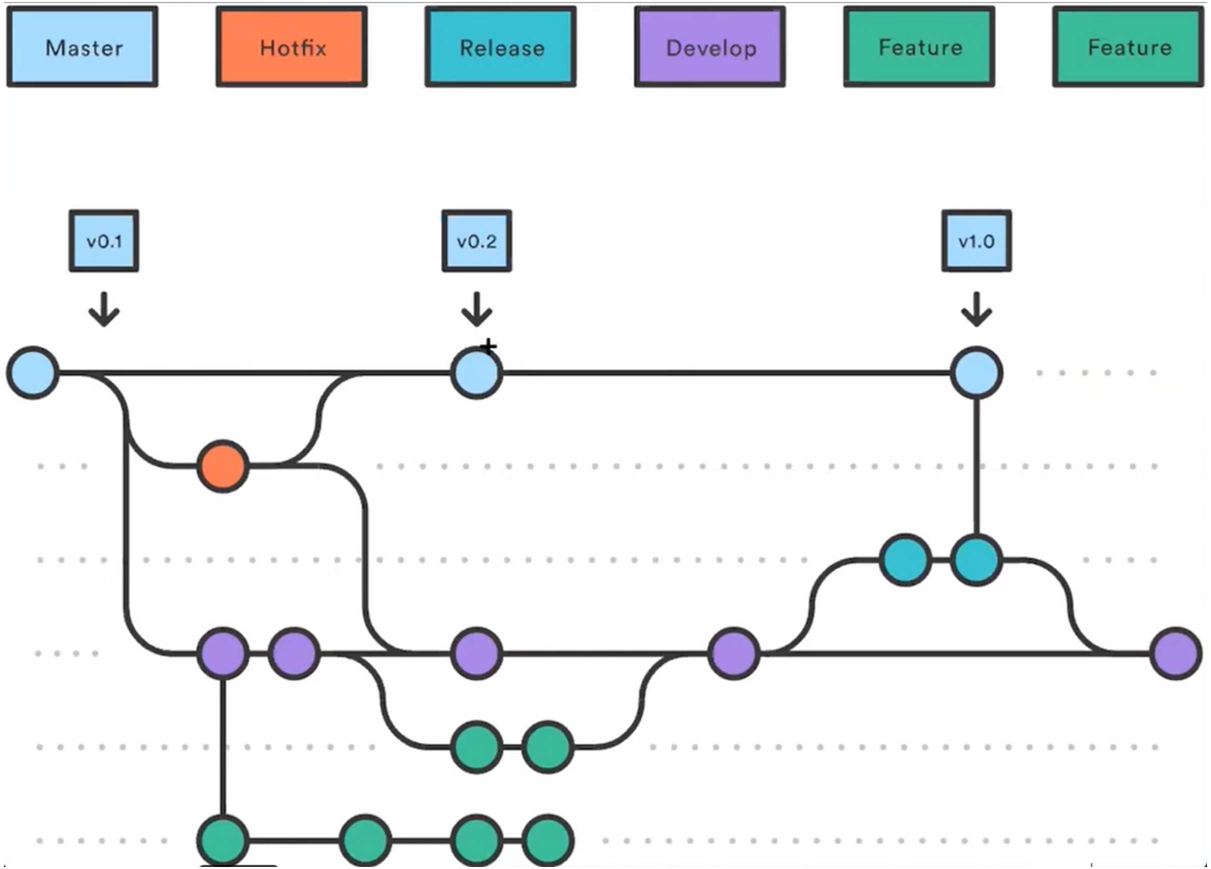


- → Production of the project lives on master/main branch
- → Branches are reference to a commit



```
[Erics-Mac:project eric$ git branch
* master
```

Branches



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
```

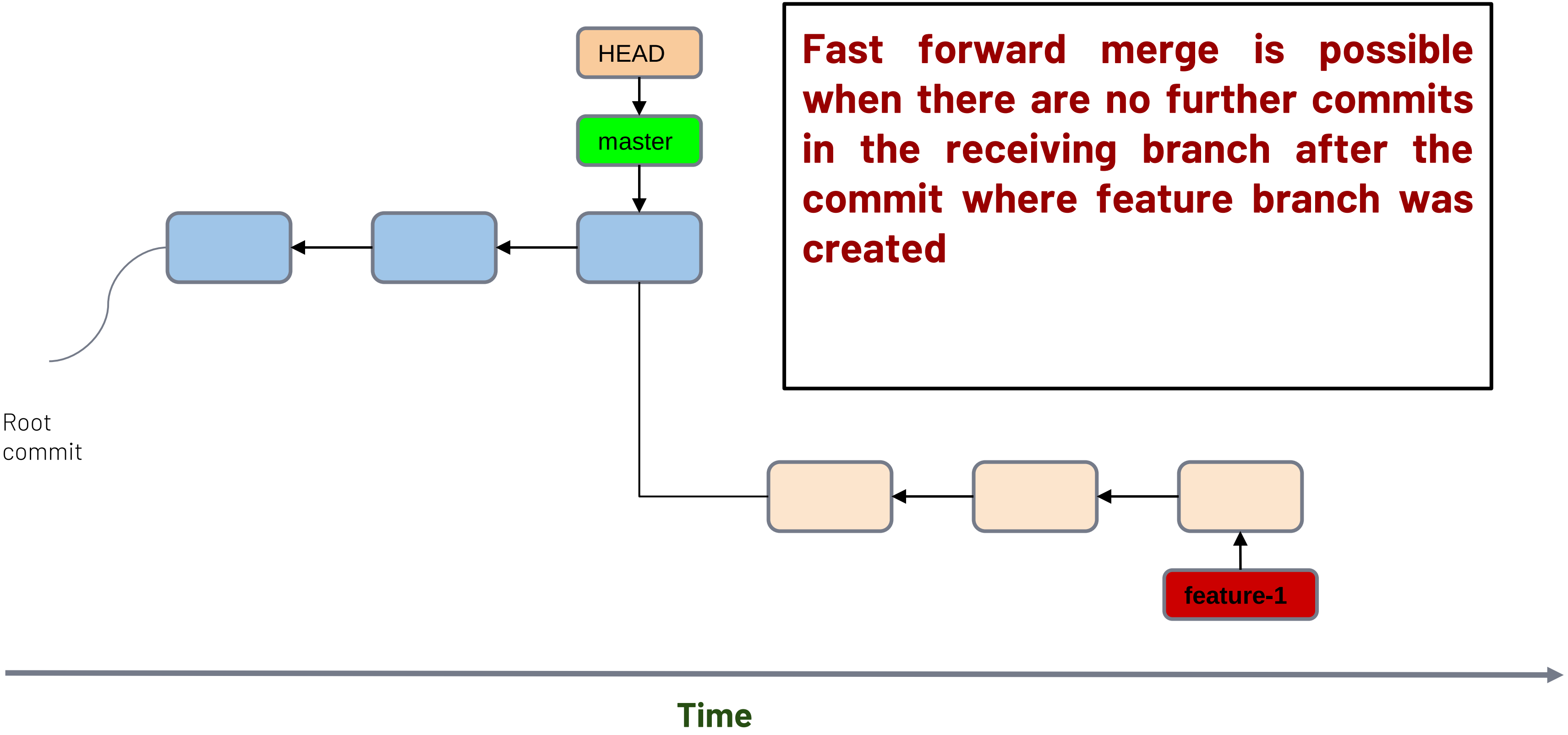
Deleting branches

→ delete a local branch

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

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

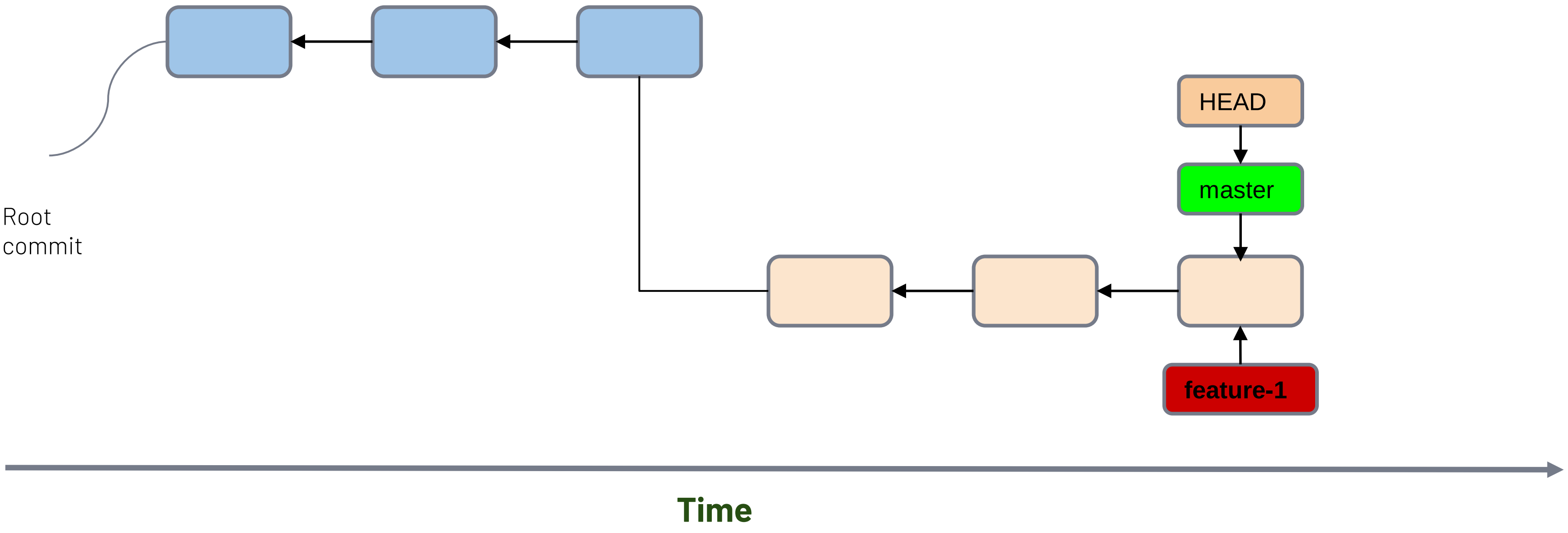
Fast forward merge



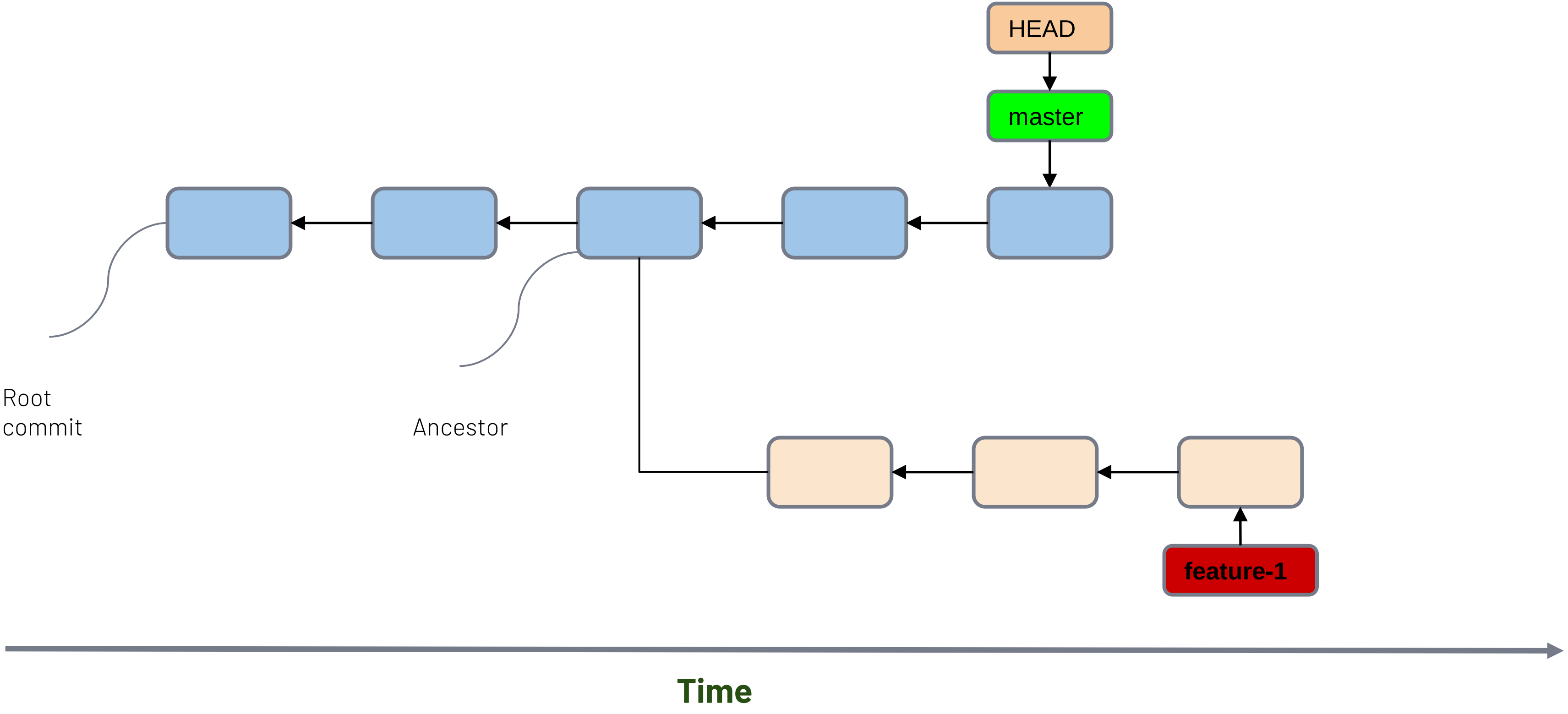
Fast forward merge is possible when there are no further commits in the receiving branch after the commit where feature branch was created

Fast forward merge

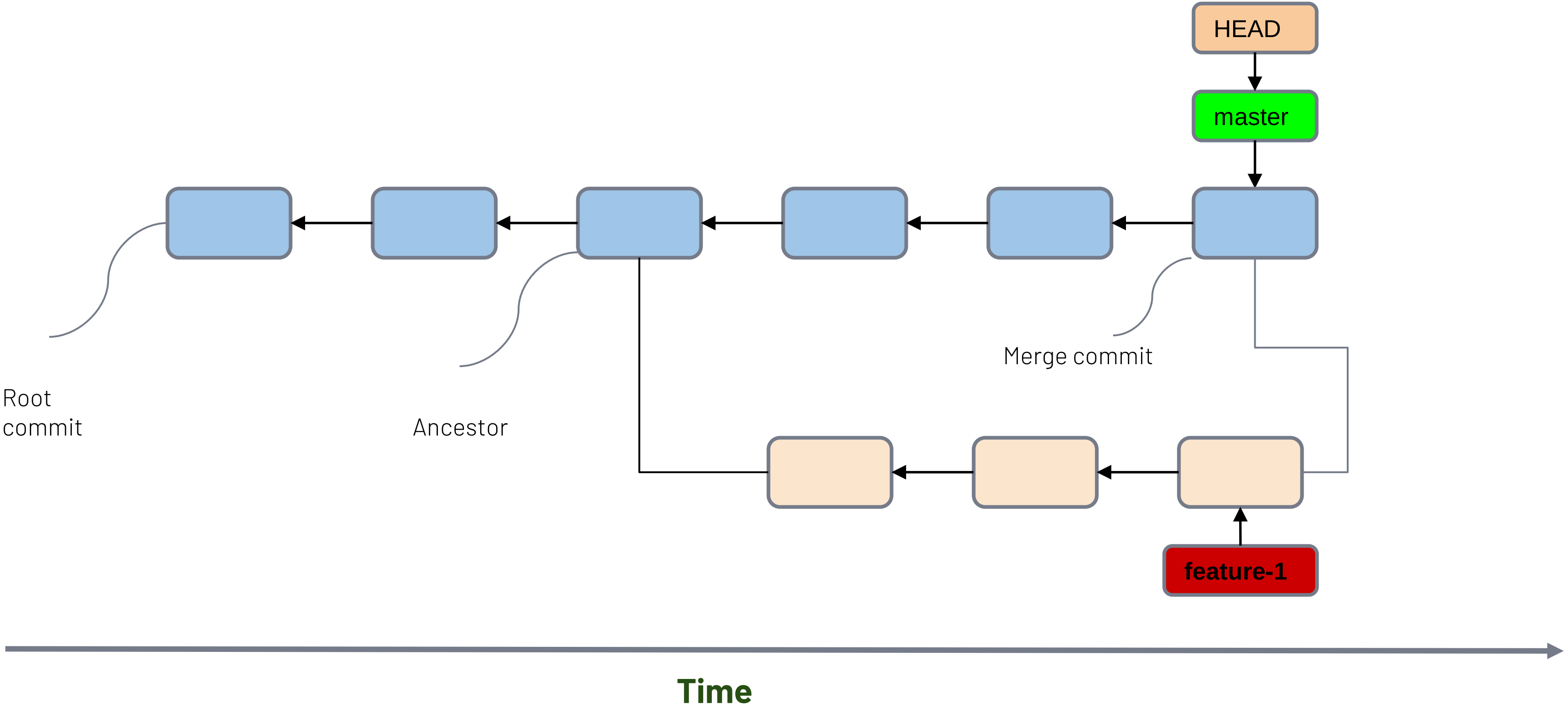
```
git merge <feature-branch>
```



3-way merge

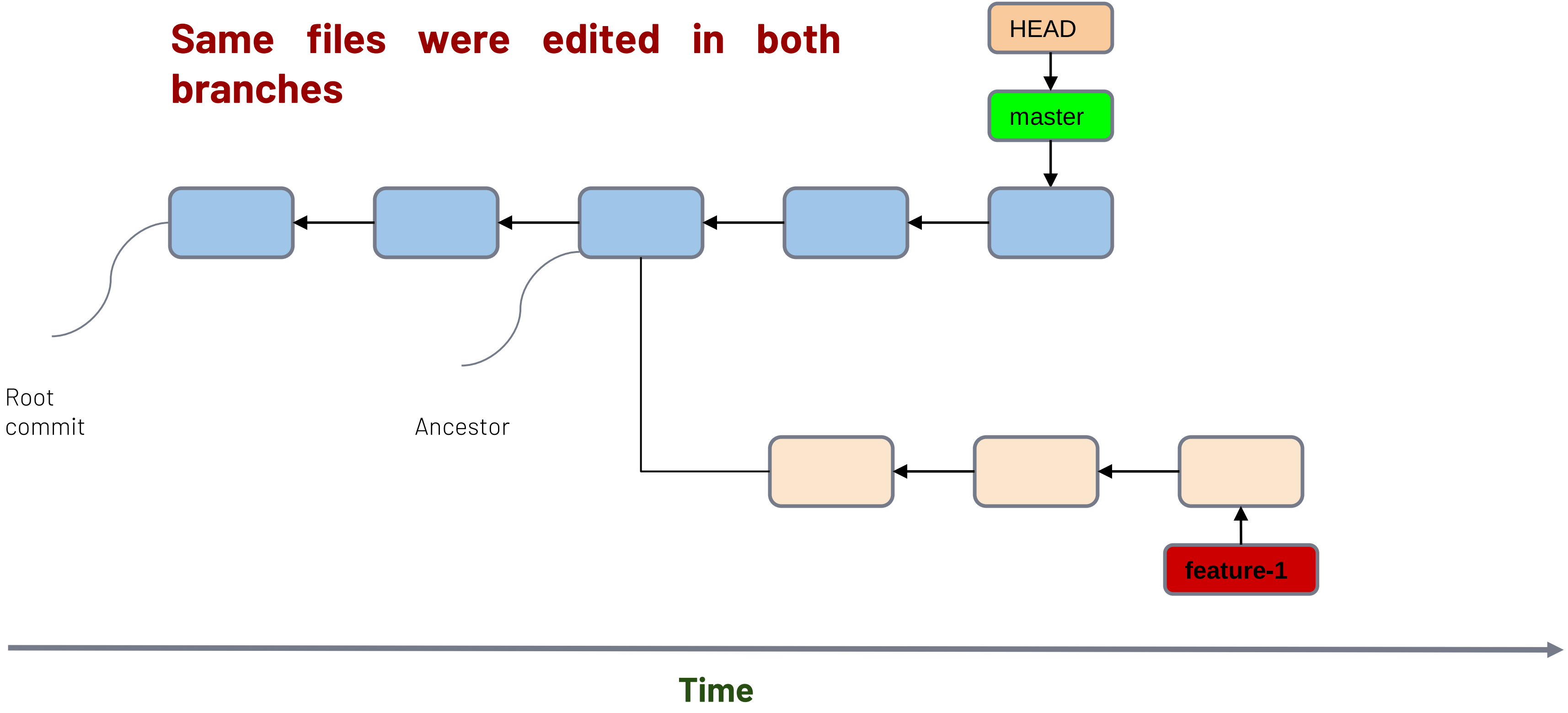


3-way merge


HEAD
master
Root commit
Ancestor
Merge commit
feature-1
Time

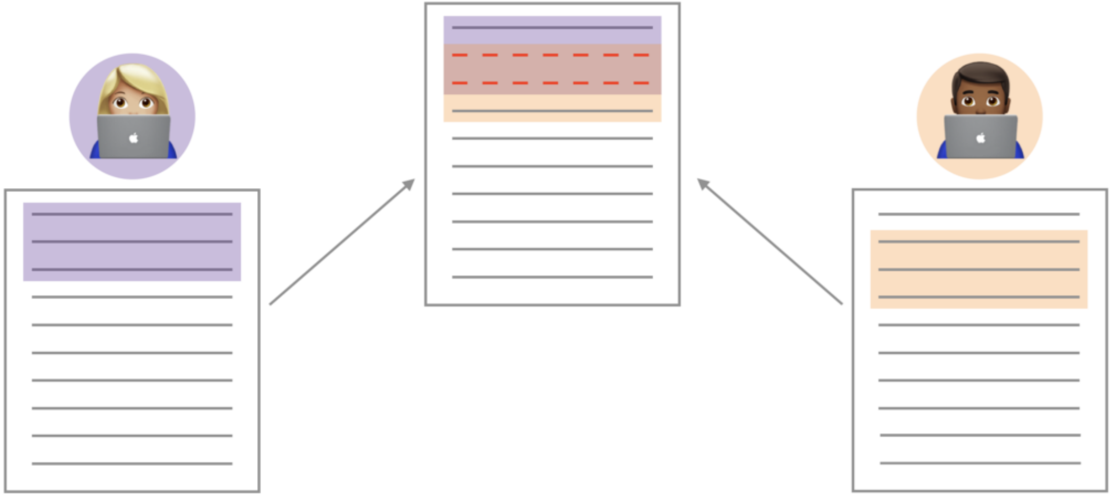
Merge Conflicts

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?