DIS08 – Data Modeling

03 – git + GitHub – Tutorial Session

Mandy Neumann, Technische Hochschule Köln, Cologne, Germany

Version: SS 2020

Technology
Arts Sciences
TH Köln

Topics for Today

1. Q&A
2. Resolving conflicts
3. Undoing things

Topics for Today

1. Q&A

2. Resolving conflicts

3. Undoing things

Open Questions

Quiz!

What does `git init` do?

a) Commit your files to the repository
b) Delete a git project
c) Initialize a new git project
d) Add all files to the staging area

Quiz!

What does `git init` do?

a) Commit your files to the repository
b) Delete a git project
c) Initialize a new git project
d) Add all files to the staging area

Quiz!

The output below is typical of which command?

```
commit bda95786432d142bbff996ad32045fa4f32ec619
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date: on Nov 16 13:13:33 2015 -0500
First commit
```

a) git diff
b) git log
c) git status
d) git add filename

Quiz!

The output below is typical of which command?

```
commit bda95786432d142bbff996ad32045fa4f32ec619
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date: on Nov 16 13:13:33 2015 -0500
First commit
```

a) git diff
b) git log
c) git status
d) git add filename

Quiz!

The command `git status` shows

a) A git project's commit history
b) Untracked files only
c) Untracked files and file changes staged for commit
d) File changes staged for commit

Quiz!

The command `git status` shows

a) A git project's commit history
b) Untracked files only
c) Untracked files and file changes staged for commit
d) File changes staged for commit

Quiz!

What's wrong with the code below?

```
git commit -m Add new scene to screenplay
```

a) The commit message lacks quotation marks
b) The "-m" option is not necessary here
c) The commit message should be in all caps
d) The "-m" option goes before the word "commit"

Quiz!

What's wrong with the code below?

```
git commit -m Add new scene to screenplay
```

a) The commit message lacks quotation marks
b) The "-m" option is not necessary here
c) The commit message should be in all caps
d) The "-m" option goes before the word "commit"

Quiz!

The command below

```
git branch new_branch
```

a) Changes the name of a branch
b) Lists the commit history of the new branch
c) Switches you over to a new branch
d) Creates a new branch

Quiz!

The command below

```
git branch new_branch
```

a) Changes the name of a branch
b) Lists the commit history of the new branch
c) Switches you over to a new branch
d) Creates a new branch

Quiz!

A Git project has a branch “bug-fix”. How do you switch to it?

a) `git branch bug-fix`
b) `git switch master bug-fix`
c) `git checkout bug-fix`
d) `git switch bug-switch`

Quiz!

A Git project has a branch “bug-fix”. How do you switch to it?

a) `git branch bug-fix`
b) `git switch master bug-fix`
c) `git checkout bug-fix`
d) `git switch bug-switch`

Quiz!

Why is the branch name “my branch” invalid?

a) The word “my” cannot be used
b) Valid branch names must contain a dash
c) Branch names cannot contain whitespace
d) Branch names must be capitalized

Quiz!

Why is the branch name “my branch” invalid?

a) The word “my” cannot be used
b) Valid branch names must contain a dash
c) Branch names cannot contain whitespace
d) Branch names must be capitalized

Topics for Today

1. Q&A

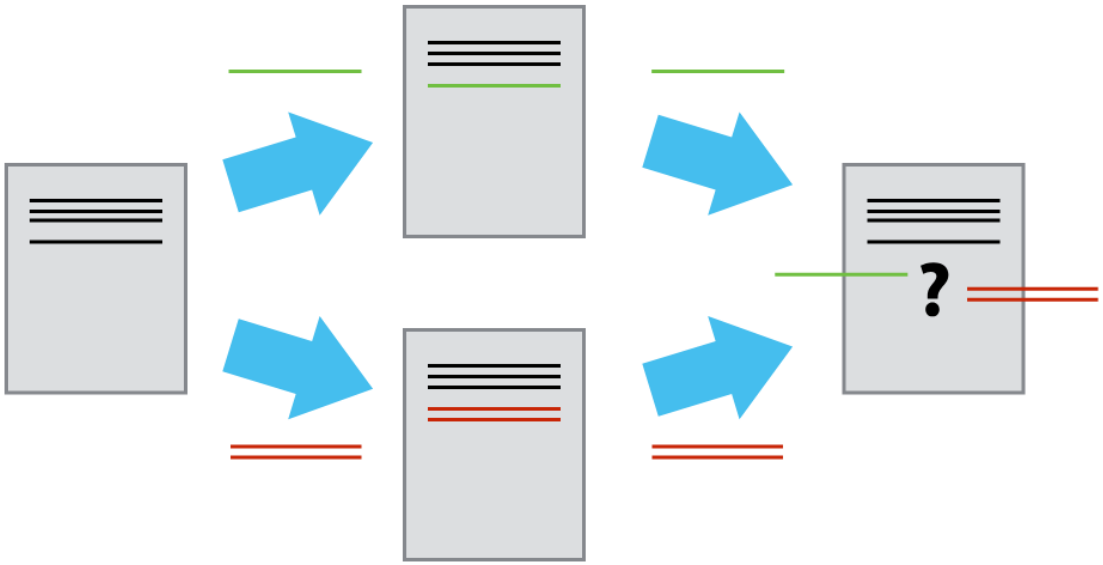
2. Resolving conflicts

3. Undoing things


GIT MERGE WITHOUT CONFLICTS
WHAT
DID I DO WRONG?
quickmeme.com

The Feared Merge Conflict

- Conflicts occur when..
  - .. two or more people change the same file(s) at the same time
  - .. one person changes the same file(s) from several machines
  - .. one changes the same file(s) on different branches.
- git does not allow people to overwrite each other's changes blindly, but highlights conflicts so that they can be resolved.



The Feared Merge Conflict

Merge conflicts are git-speak for "I need a human to make a decision".

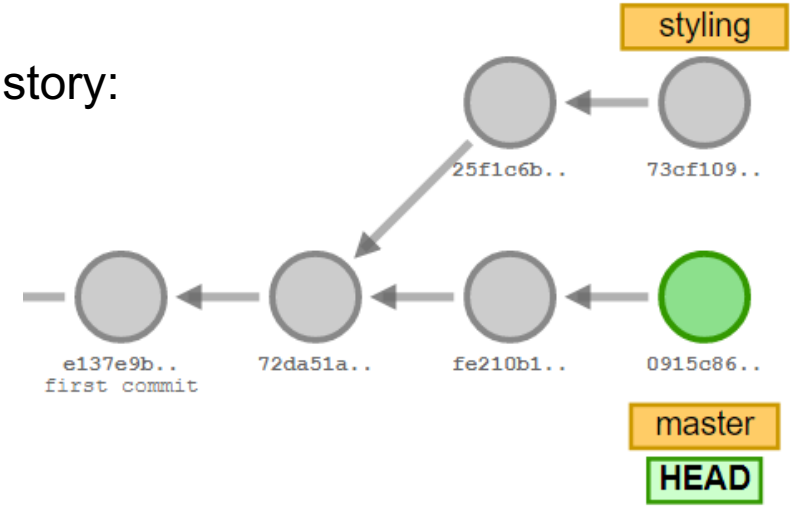
There's nothing to worry about them.

git will indicate the conflicting lines in your files.

git's status output will also guide you through the process.

Example

- Suppose you have this project history:



- There are two lines of work with one common anchestor (72da51a..).

- You happen to have modified the same file in both branches.

- You want to integrate your changes from `styling` into `master`.

Example

```
Mandy@DESKTOP-N81SCAQ MINGW64 ~/dis08/testproject (master)
$ git checkout styling
Switched to branch 'styling'

Mandy@DESKTOP-N81SCAQ MINGW64 ~/dis08/testproject (styling)
$ git log -n 2
commit 22bb113e7633c65c5dd9f6e43cb472457e893422 (HEAD -> styling)
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 11:03:52 2019 +0200

    Introduce some CSS styling

commit a938ad039397b875c2ae4bb9b3a008a0ce19abf3
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 10:56:01 2019 +0200

    Add index.html

Mandy@DESKTOP-N81SCAQ MINGW64 ~/dis08/testproject (styling)
$ git checkout master
Switched to branch 'master'

Mandy@DESKTOP-N81SCAQ MINGW64 ~/dis08/testproject (master)
$ git log -n 2
commit 5bf1fbfa7052f540ab4ecace9bec50a0b403772d (HEAD -> master)
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 11:04:41 2019 +0200

    Change page title

commit a938ad039397b875c2ae4bb9b3a008a0ce19abf3
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 10:56:01 2019 +0200

    Add index.html
```

Commit a938a… is the last one that both branches share.

After that, each branch created a commit separately.

Both changed the same file though!

Example

- Automatic merge will fail, because git cannot know which changes you want to keep in the merged version:

```
Mandy@DESKTOP-N81SCAQ MINGW64 ~/dis08/testproject (master)
$ git merge styling
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

- Conflicting lines are indicated by `<<<<<<<`, `=======` and `>>>>>>>`:

```
15    </head>
16    <body>
17  <<<<<<< HEAD
18      <h1>All about cats</h1>
19  =======
20      <h1 class="red-text">The best page ever</h1>
21  >>>>>>> styling
22      <p>Cat ipsum dolor sit amet, jump launch to pou
23  litter box until treats are fed. Go into a room to
```

Example

- Now you need to decide which lines / changes to keep – modify the file accordingly and also delete the conflict markers.
- Tell git that you have resolved the conflict by adding the file.
- Then commit – git will automatically create a commit message in this case.

```
Mandy@DESKTOP-N81SCAQ MINGW64 ~/dis08/testproject (master)
$ git log -n 4
commit bfcf261f9e39de70ca256a781a57966bed8a02fb (HEAD -> master)
Merge: 5bf1fbf 22bb113
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 11:19:28 2019 +0200

    Merge branch 'styling'

commit 5bf1fbfa7052f540ab4ecace9bec50a0b403772d
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 11:04:41 2019 +0200

    Change page title

commit 22bb113e7633c65c5dd9f6e43cb472457e893422 (styling)
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 11:03:52 2019 +0200

    Introduce some CSS styling

commit a938ad039397b875c2ae4bb9b3a008a0ce19abf3
Author: Mandy Neumann <Mandy.Neumann@th-koeln.de>
Date:   Wed Apr 17 10:56:01 2019 +0200

    Add index.html
```

Now it‘s your turn:

- Create a conflict on purpose: Modify the same part of a file on two different branches. Try to merge one branch into the other. Resolve the conflict.
- Here‘s a refresher on basic git commands:

git command	Action
`git init <name>`	Initialize new repository with a name
`git add <file>`	Add file(s) to staging area
`git status`	Check status
`git commit`	Create a snapshot of staging area (opens configured editor for commit message)
`git commit -m`	Same as before, but enter commit message directly
`git log`	Show commit history
`git branch <newbranch>`	Create a branch named „newbranch“
`git checkout <branch>`	Checkout a branch
`git merge <branch>`	Merge „branch“ into current branch

Topics for Today

1. Q&A

2. Resolving conflicts

3. Undoing things

Scenarios to undo Things

- You forgot to include a file/change in your recent snapshot.
- You want to change the commit message of a snapshot.
- You accidentally staged a file you didn‘t want to include.
- You really messed something up and need to go back in time…

Change your most recent commit

- The --amend option allows to redo the most recent commit.

```
$ git commit --amend
```

- It can be used to both include additional changes, or just edit the commit message.

- This command is most useful if you only need to make minor improvements to your commit, without cluttering your history.

Un-stage files

- You accidentally staged a change you need to un-stage before doing a commit.

- Use reset to remove that change from the staging area:

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

Reset files

- You realize you made some changes that you need to revert, maybe you broke your code.
- You want to get the version of the file back as it was in the previous commit.
- Make use of the checkout command to get back a specific version from the history:

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

- Beware that git will replace your file, so use this command with caution!

Now it‘s your turn again:

- Try to do and undo something..
- Here‘s a refresher on related git commands:

git command	Action
`git commit --amend`	Replace the most recent commit
`git reset HEAD <file>`	Un-stage a file
`git checkout -- <file>`	Undo all local changes to a file by getting its version from the HEAD commit
`git checkout HEAD <file>`	(same as above)

References

- Official docs:
  https://git-scm.com/doc
- More details on undoing things with git reset:
  https://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified
- Some useful advice on how to get out of messy situations:
  http://ohshitgit.com