DevOps basics: Introduction to Git

Outline

- Introduction to source control
  - History and fundamentals concepts behind source control
  - Centralized vs. distributed version control
- Introduction to Git
  - What is Git? Basic Git concepts and architecture
  - Git workflows: Creating a new repo (adding,committing code)
  - HEAD
  - Git commands (checking out code)
  - Master vs. branch concept
  - Creating a branch/ Switching between branches
  - Merging branches and resolving conflicts

What is a version control system?

- A way to manage files and directories
- Track changes over time
- Recall previous versions
- `source control` is a subset of a VCS (version control system)

Some history of source control

(1972) Source Code Control System (SCCS)

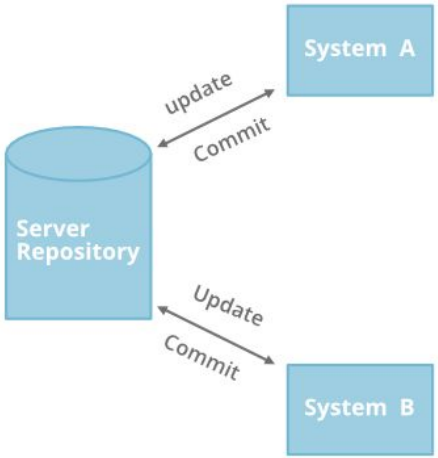
(1982)Revision Control System(RCS)

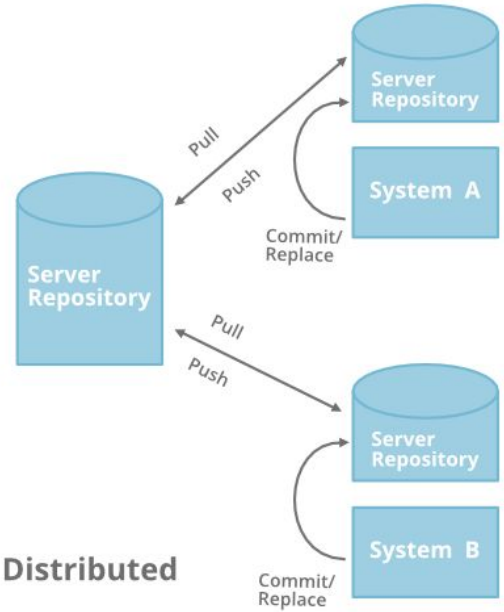
(1986)Concurrent Version System(CVS)

(2000)Apache Subversion(SVN)

Distributed version control

No central server
Every developer is a client, the server and the repository





What is git?

What is git?

- Created by Linus Torvalds, April 2005
- Replacement for BitKeeper to manage Linux kernel changes
- A command line version control program
- Uses checksums to ensure data integrity
- Distributed version control (like BitKeeper)
- cross-platform(including Windows!)
- Open source , free

Git distributed version control

- “If you're not distributed, you`re not worth using” -Linus Torvalds
- No need to connect to central server
- Can work without internet connection
- No single failure point
- Developers can work independently and merge their work later
- Every copy of a Git repository can serve either as the server or a client
- Git tracks **changes**, not versions
- Bunch of little **changes sets** floating around

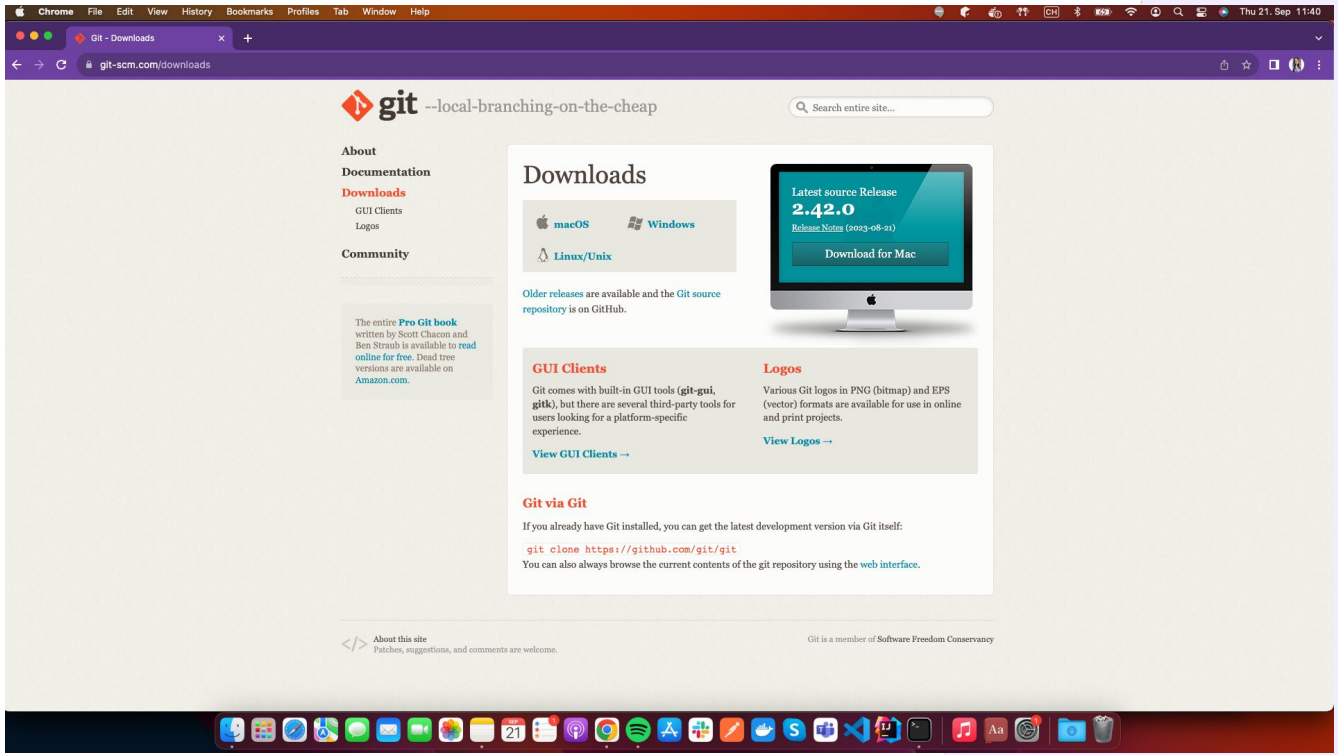
Is Git for me?

- People primarily working with source code
- Anyone wanting to track edits (especially changes to text files)
  -review history of changes
  -anyone wanting to share, merge changes

- Anyone not afraid of command line tool

How do I get it?

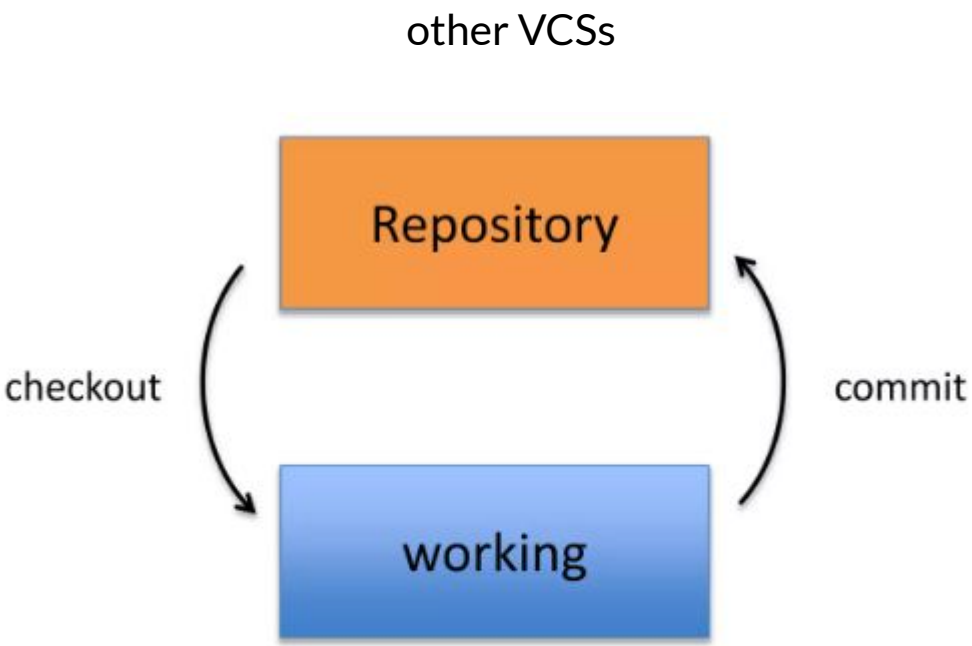
https://git-scm.com/downloads



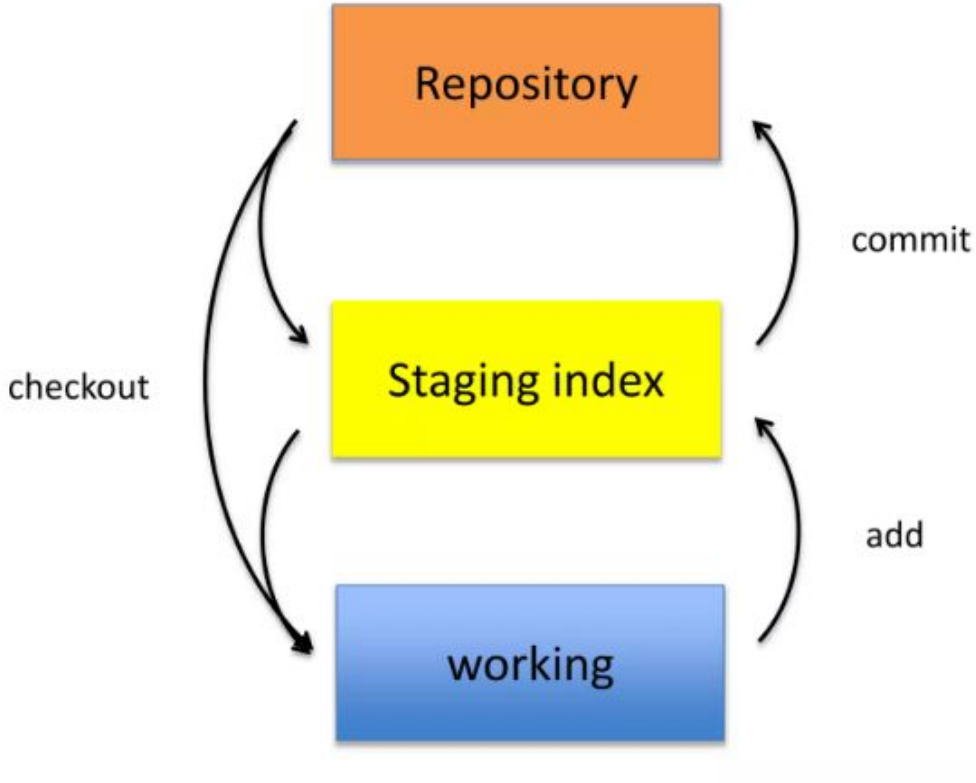
What is Repository?

- “Repo” = repository
- Usually used to organize a single project
- Repos can contain folders and files, images, videos, spreadsheets, and data sets- anything your project needs

Two-tree architecture



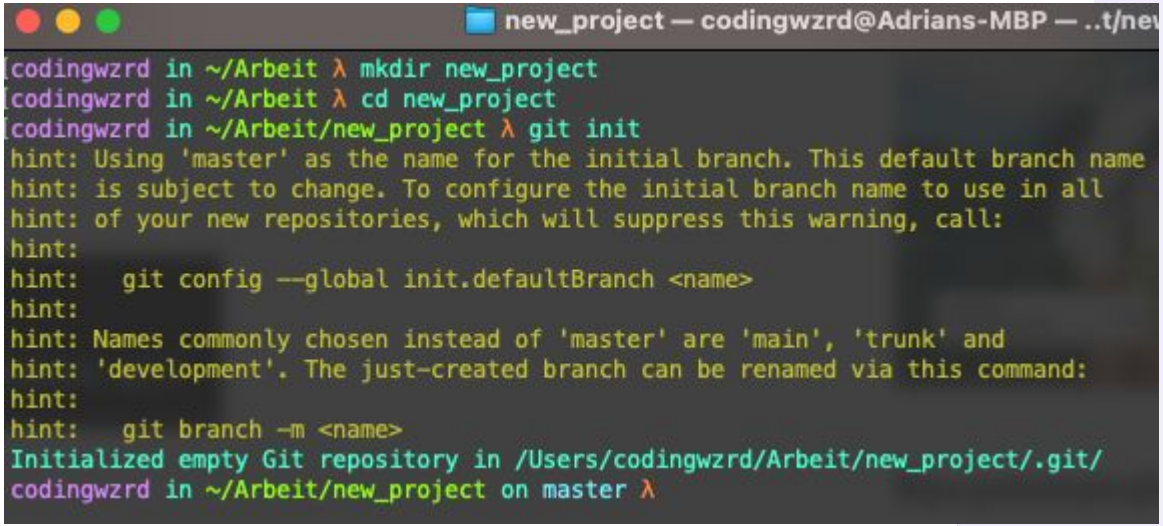
Git uses a three-tree architecture



A simple Git workflow

1. Initialize a new project

```
git init
```



A simple Git workflow

2.Add a file using a text editor to the directory
3. Add every changes that has been made to the directory:

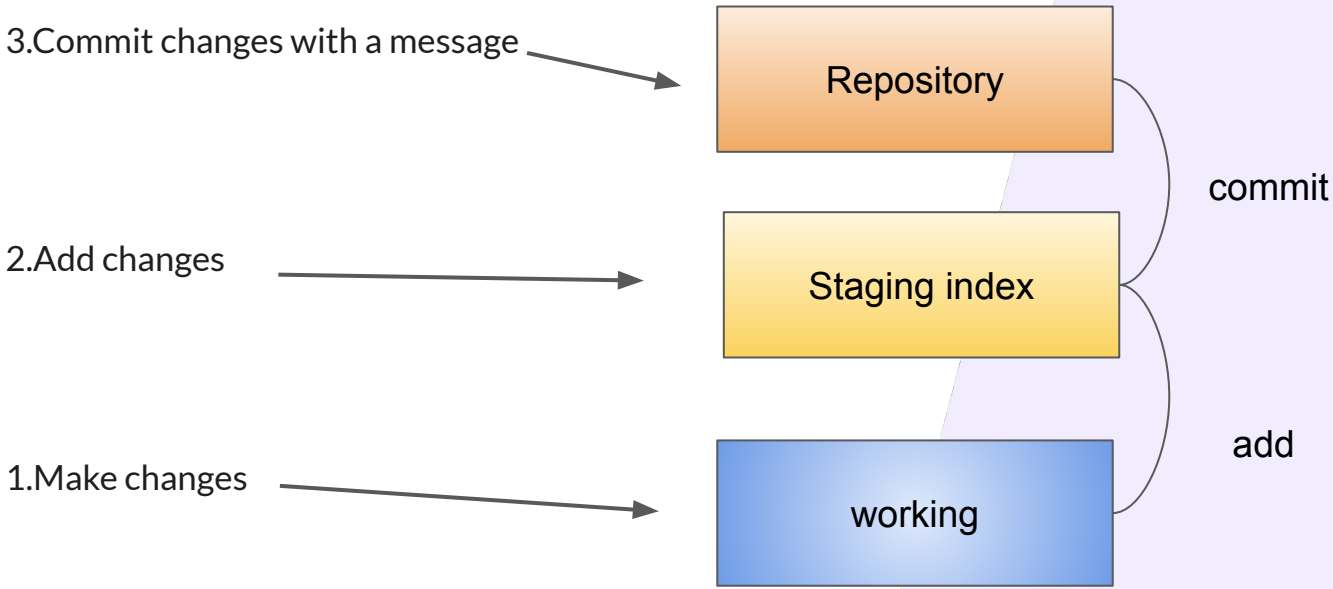
```
git add .
```

4. Commit the change to the repo:

```
git commit -m “Here is your message”
```

```
[codingwzrd in ~/Arbeit/new_project on master λ git add .
[codingwzrd in ~/Arbeit/new_project on master λ git commit -m "Your message here"
On branch master
```

After initializing a new git repo



A note about commit messages

- Tell what it does (present tense)
- Single line summary followed by blank space followed by more complete description
- Ticket or bug number helps

A note about commit messages

- Tell what it does (present tense)
- Single line summary followed by blank space followed by more complete description
- Ticket or bug number helps

Good and bad examples

Bad: “Typo fix”
Good: “Add missing / in CSS section”

Bad: “Updates the table. We will discuss next Week”

The HEAD pointer

- Points to a specific commit in repo
- As new commits are made, the pointer changes
- HEAD always point to the “tip” of the currently checked-out branch in the repo
- (not the working directory or staging index)
- Last state of repo
- HEAD points to parent of next commit (when writing the next commit takes place)

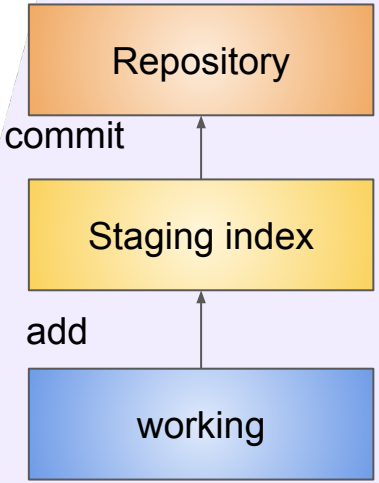
Which files were changed?

git status

Allows one to see where files are in the three tree scheme

```
codingwzrd in ~/Arbeit/new_project on master λ git status
On branch master

No commits yet
```



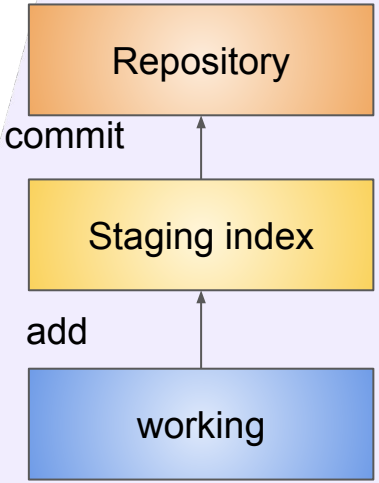
What changes in working directory?

```
git diff
```

Compares changes to file between repo and working directory

Note: git diff- - staged - compares staging index to repo

Note: git diff filename can be used as well

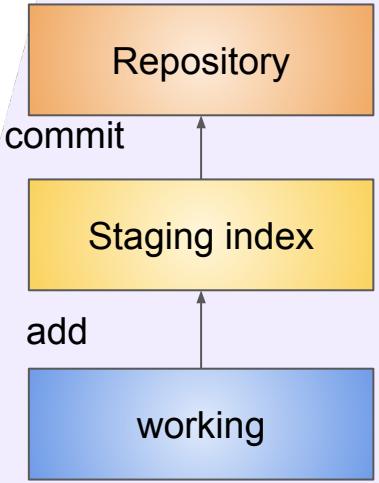


Deleting files from the repo

```
git rm filename.java
```

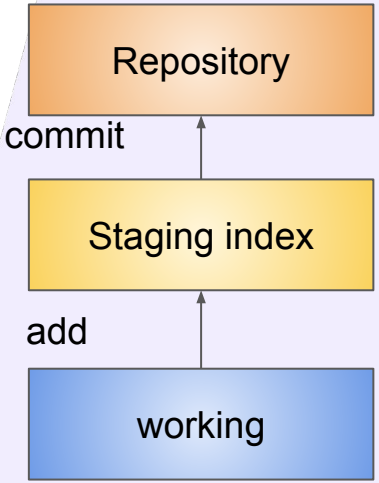
Moves deleted file change to staging area

It is not enough to delete the file in your working directory.
You must commit the change.



Moving (renaming) files

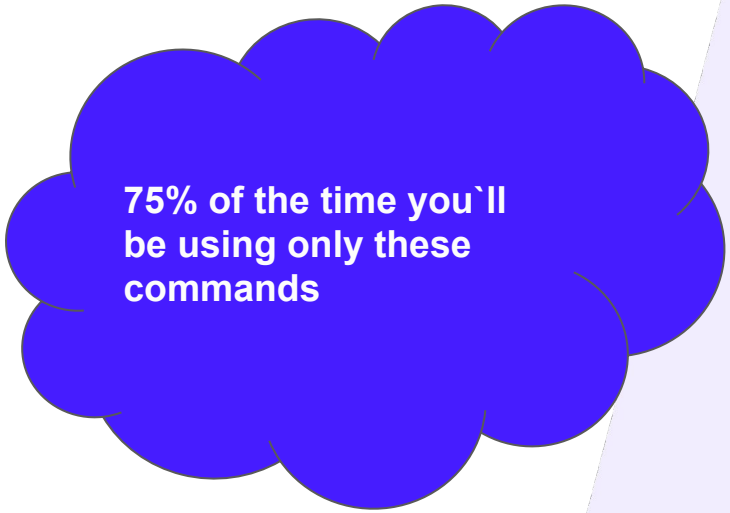
```
git mv filename1.java    filename2.java
```



Good news!!!

git init

git commit



git status

git add .

git diff

git log

git mv

git rm

What if I want to undo changes made to working directory?

```
git checkout something
```

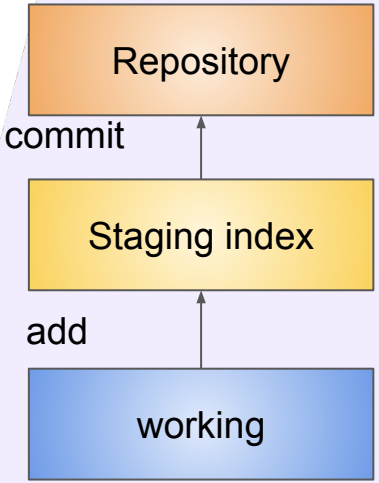
(where “something” is a file or an entire branch)

git checkout will grab the file from repo

Example:

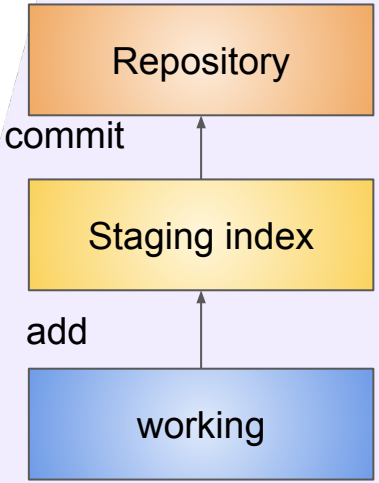
```
git checkout --file1.java
```

(“checkout file `file1.java` from the current branch”)



What if I want to undo changes added to staging area?

```
git reset HEAD filename.java
```

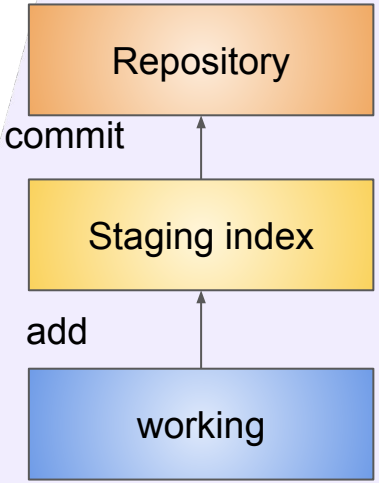


What if I want to undo changes committed to the repo?

```
git commit --amend -m “message”
```

Allows one to amend a change to the last commit

Anything in staging area will be amended to the last commit



Branching

Allows one to try new ideas

If the idea doesn't work , throw away the branch.

Don't have to undo many changes to master branch.

If it doesn't work , merge ideas into master branch.

There is only one working directory.

In which branch am I?

```
git branch
```

How do I create a new branch?

```
git branch new_branch_name
```

How do I switch to new branch?

```
git checkout new_branch_name
```

At this point, one can switch between branches, making commits etc. in either branch, while the two stay separate from one to another.

Note: In order to switch to another branch, your current working directory must be clean(no conflicts, resulting in data loss).

How do I merge a branch?

```
git merge branch_to_merge
```

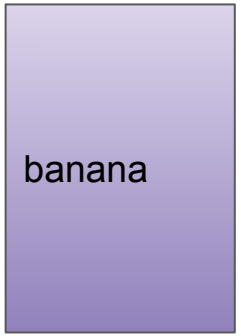
From the branch into which you want to merge another branch…

Note: always have a clean working directory when merging

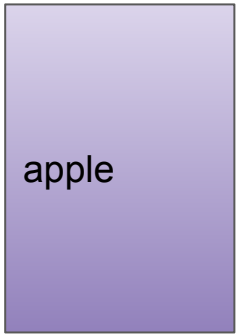
Merge conflicts

What if there are two changes to same line in two different commits?

File1.java



File2.java



master

new_feature

Resolving merge conflicts

Git will notate the conflict in the files!

Solution:

1. Abort the merge using **git merge -abort**
2. Manually fix the conflict
3. Use a merge tool (there are many out there)

Tip to reduce merge pain

- Merge often
- Keep commits small/focused
- Bring changes occurring to master into your branch frequently(tracking!!)

Break 10 minutes

Questions?

Thank you for your attention!