Introduction to version control with Git

Day 2: Branching, Merging and collaboration workflows

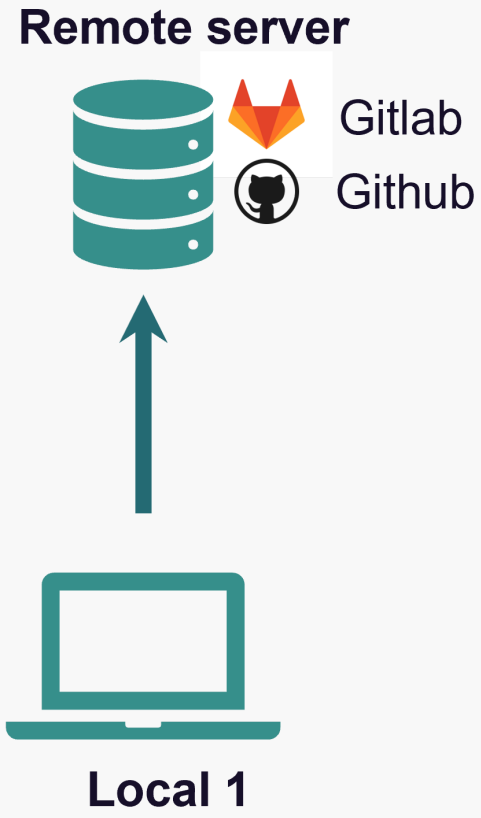
Selina Baldauf

September 16, 2023

Recap

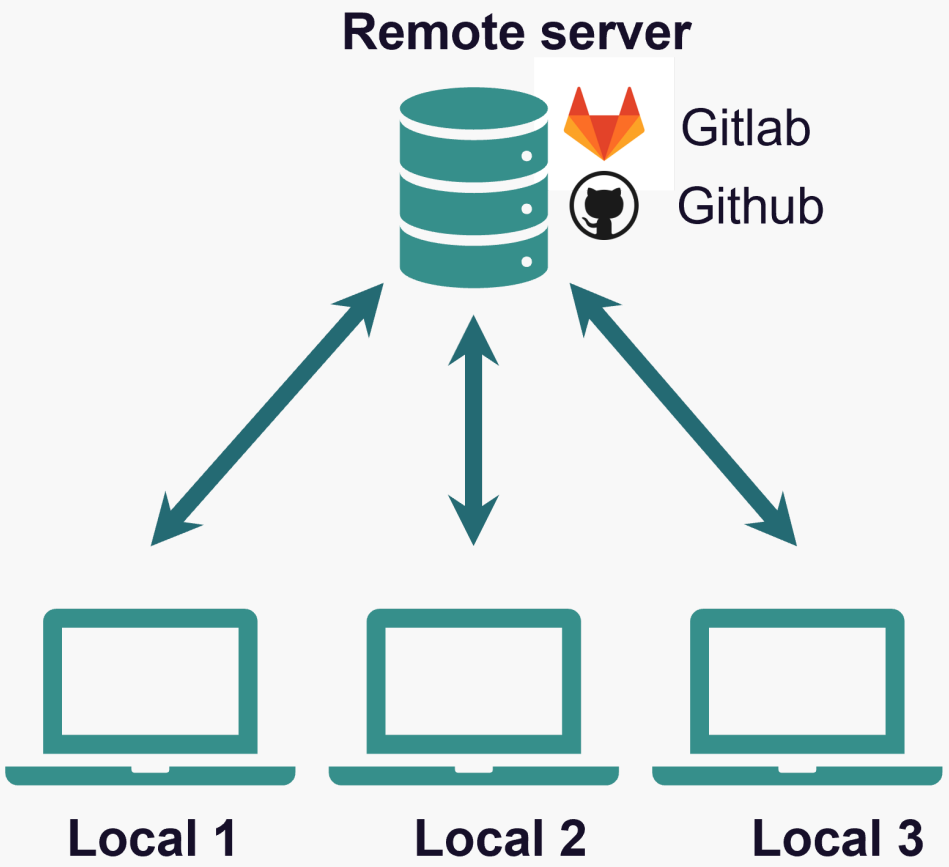
The basic Git workflow consists of the following steps:

1. **Initialize** a Git repository
2. **Work** on the project
3. **Stage** and **commit** files to the local repository
4. **Publish** the repository on a remote server
5. **Push** future changes to the remote repository



Version control with Git

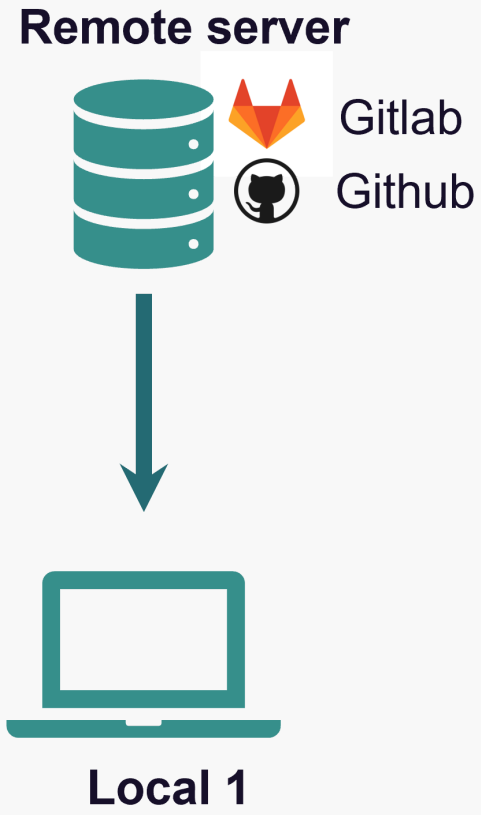
Git is a **distributed version control system**



- Idea: many *local* repositories synced via one *remote* repo
- What we still miss: From remote to local
  - This is essential for collaboration

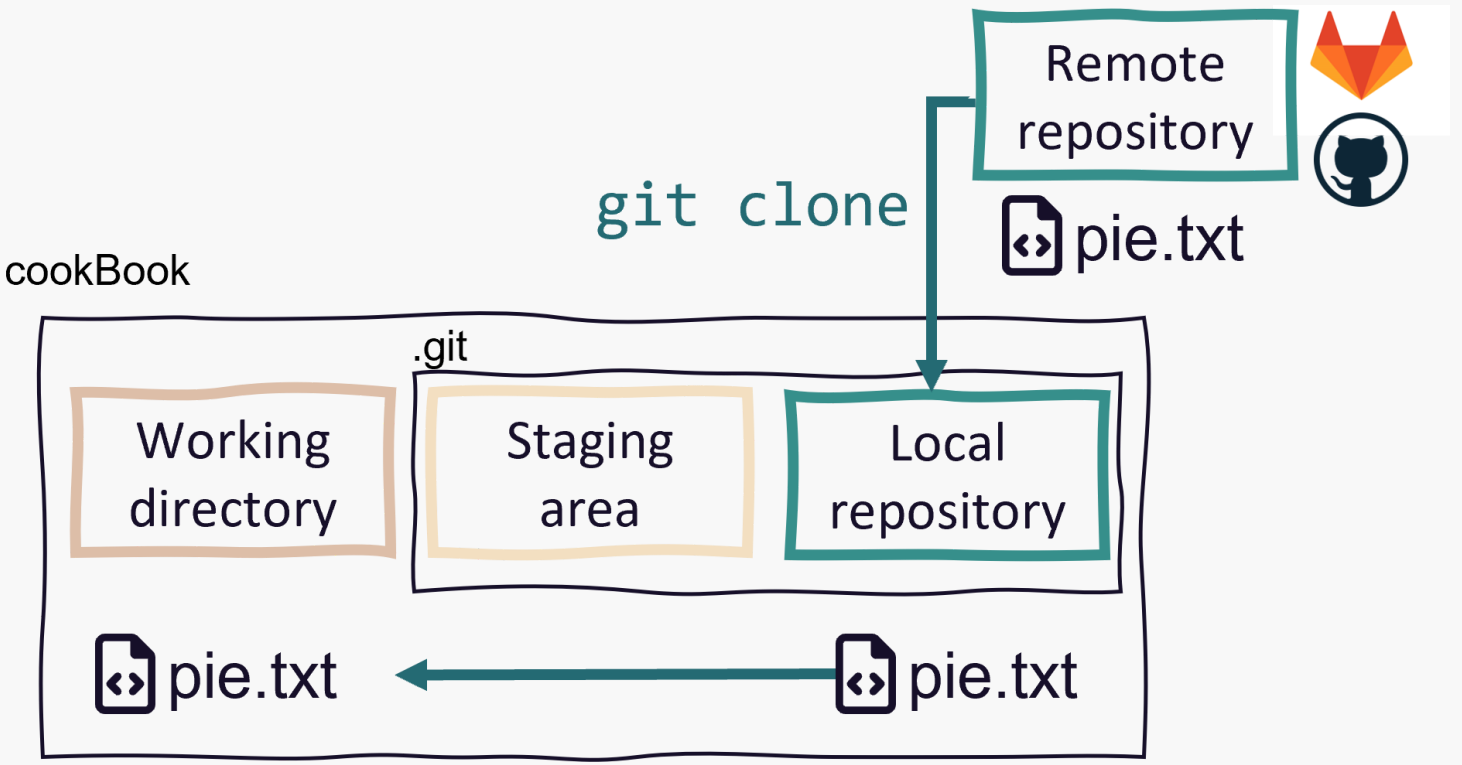
Get a repo from a remote

- In Git language, this is called **cloning**
- Get a copy of your own repository on a different machine
- Get the repository from somebody else to collaborate



Get a repo from a remote

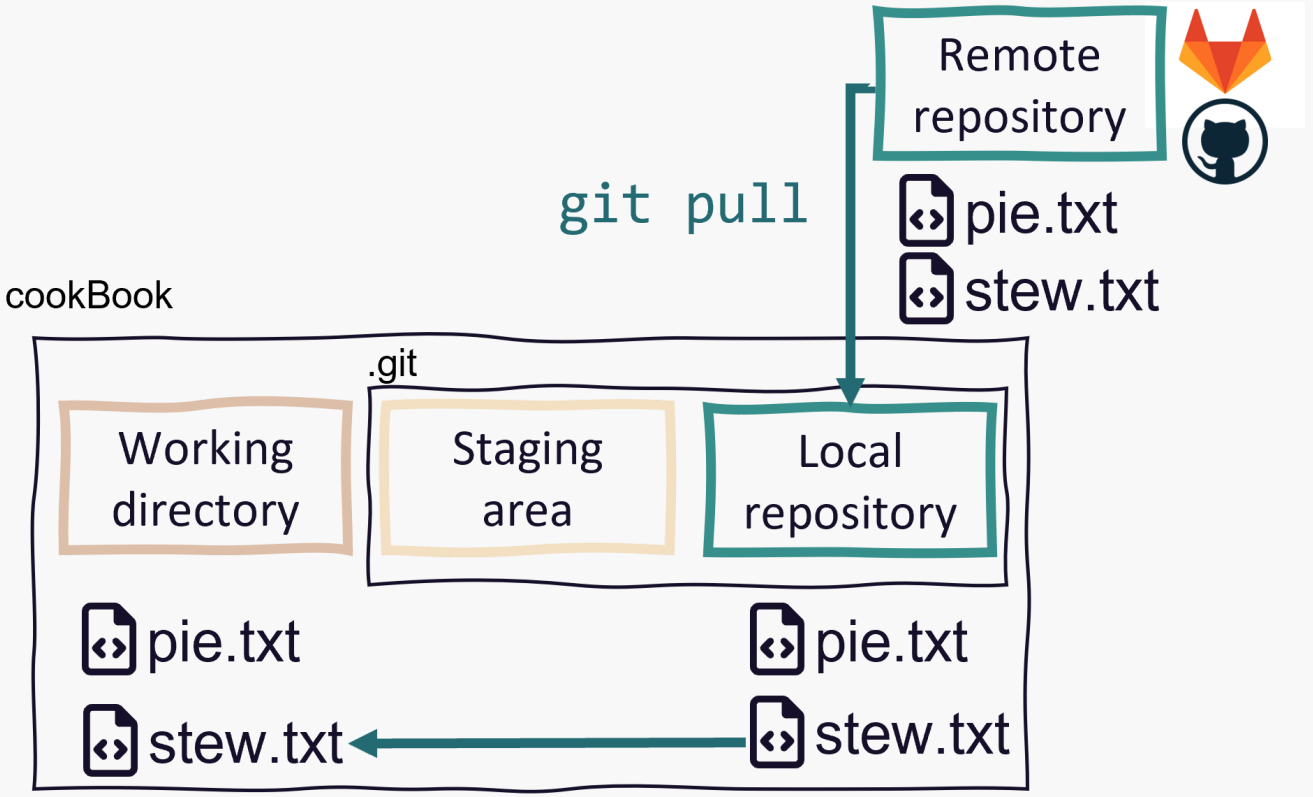
By cloning, you get a **full copy** of the repository on your machine.



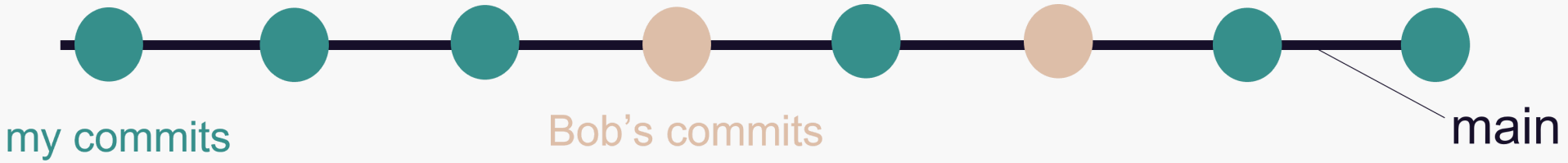
- If the clone is authorized it can also commit and push

Get changes from the remote

- Local changes, publish to remote: `git push`
- Remote changes, pull to local: `git pull`

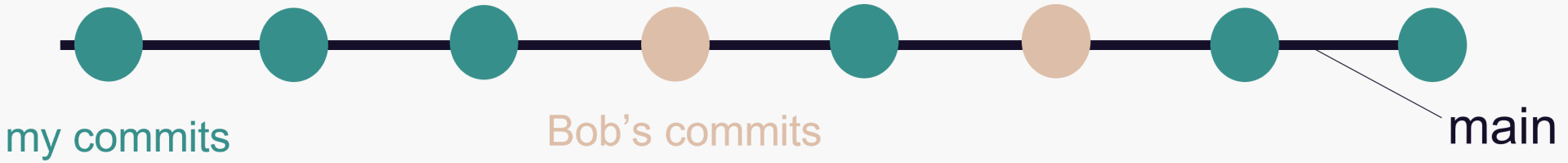


A simple collaboration workflow



- One remote repo on Github, multiple local repos
- Idea: Everyone works on the same branch
  - **Pull before** you start **working**
  - **Push after** you finished **working**

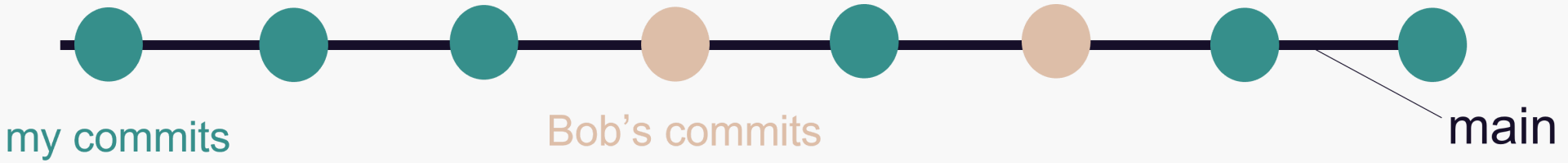
A simple collaboration workflow



This works well if

- Repo is not updated often
- You don't work on the same files simultaneously
- No need to discuss changes
  - Everything is directly integrated in main
- You collaborate with yourself

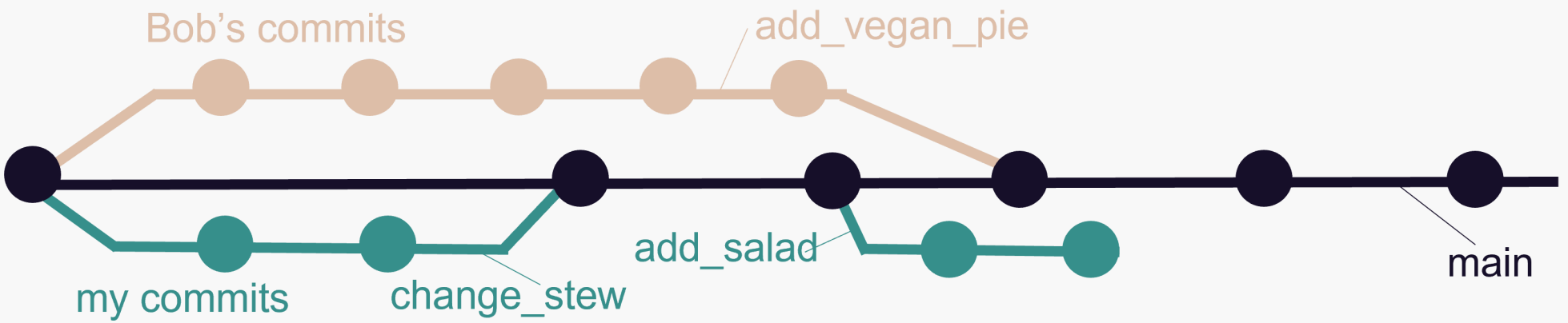
A simple collaboration workflow



This workflow starts to be problematic when

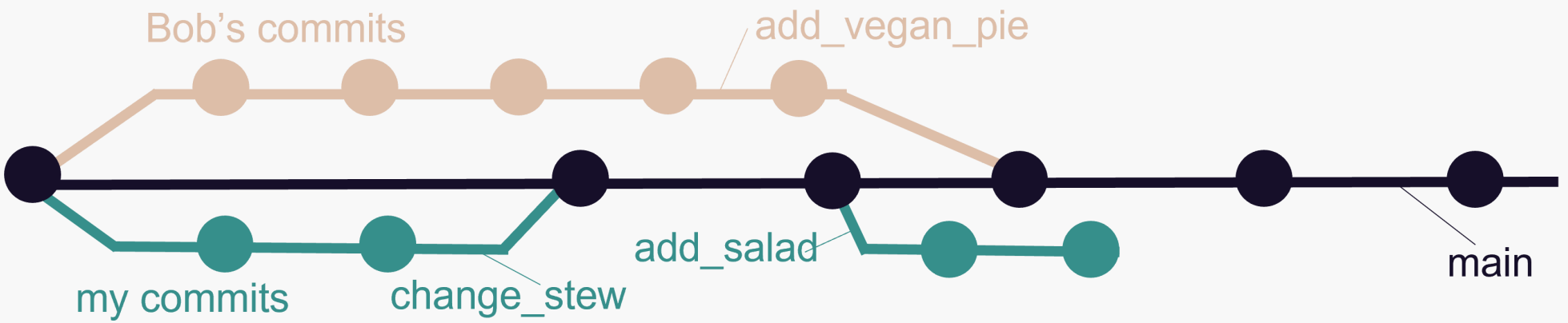
- People push often
  - Conflicts on main
- Not possible to discuss code before integration
- Difficult to just "try something out"
  - Everything goes directly to main

A branching-merging workflow



- One remote repo on Github, multiple local repos
- Idea: Everyone works on the their **separate branch**
  - **Merge** your branch with the main when you are finished

A branching-merging workflow



Advantages of this approach

- Guarantee that main always works
- Potential conflicts don't have to be solved on main

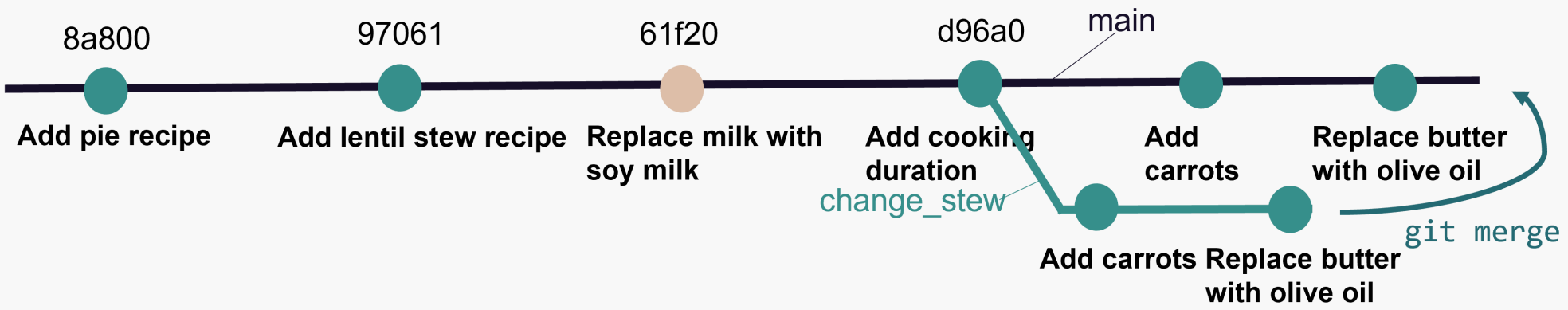
Working on a separate branch

The steps to create and work on a separate branch are easy:

- Create a local branch and switch to it
- Work on the branch like you are used to
  - Make changes, **stage** and **commit**, publish and **push**

Merging changes from a branch

To bring changes to the main branch you need to **merge** them.



Normally: Git merge brings the commits from the branch to main

Merging changes from a branch

To bring changes to the main branch you need to **merge** them.

If there was a commit on a common file in main, a *merge commit* is introduced.

Merging changes from a branch

To bring changes to the main branch you need to **merge** them.

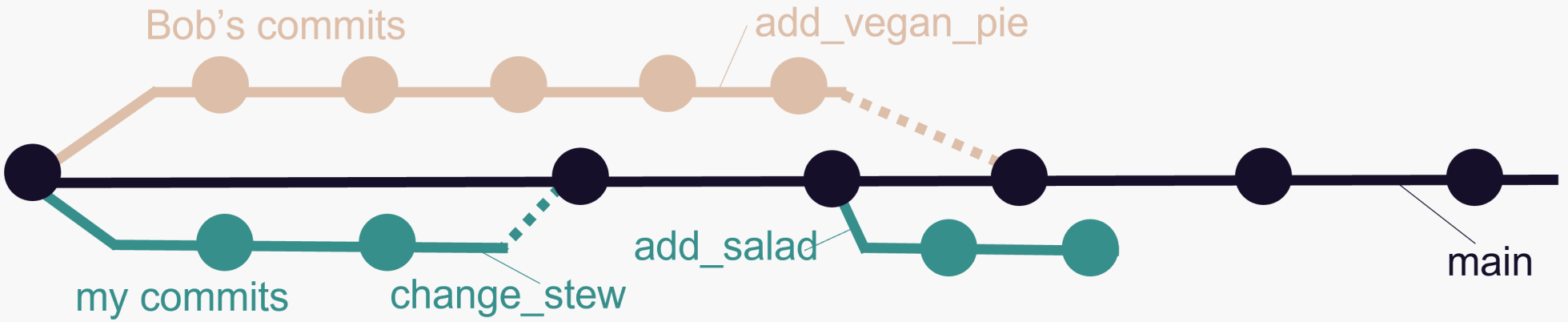
- Mostly merging happens without problems, but…
- … if the same line was edited on separate branches…
- … there will be a merge conflict 😱

Merge conflicts need to be solved manually. You need to chose which of the conflicting versions you want to keep.

Now you

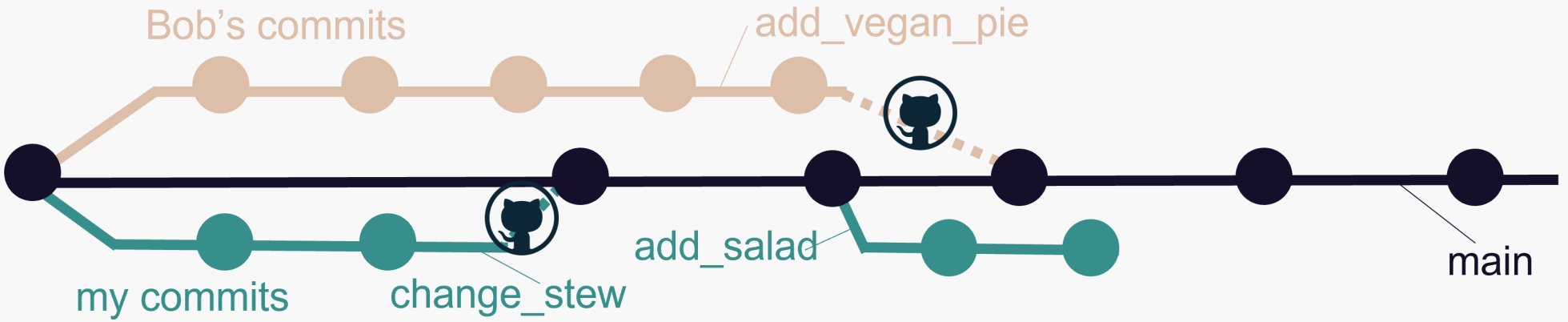
> Try out branching and merging in your cook book (10 min)

A branching-merging workflow with Github



- One remote repo on Github, multiple local repos
- Idea: Everyone works on the their separate branch
  - ~~Merge your branch with the main when you are finished~~

A branching-merging workflow with Github



- One remote repo on Github, multiple local repos
- Idea: Everyone works on the their separate branch
  - ~~Merge your branch with the main when you are finished~~
  - Create a pull request on Github to ask for a merge

A branching-merging workflow with Github

A pull request is basically asking your collaborators:

> What do you think of my changes? Can we integrate them in main or do we still need to change something?

Github has nice features for pull requests:

- **Describe your changes** in detail
- Collaborators can easily **compare versions**
- Collaborators can **discuss and comment** on your changes
- …

Now you

> Add a recipe to your partner's repo and create a pull request Answer to the pull request of you partner

Thanks for your attention

> Questions?

Next week