Introduction to version control with Git

Day 2: Branching, Merging and collaboration workflows

Selina Baldauf

September 18, 2023

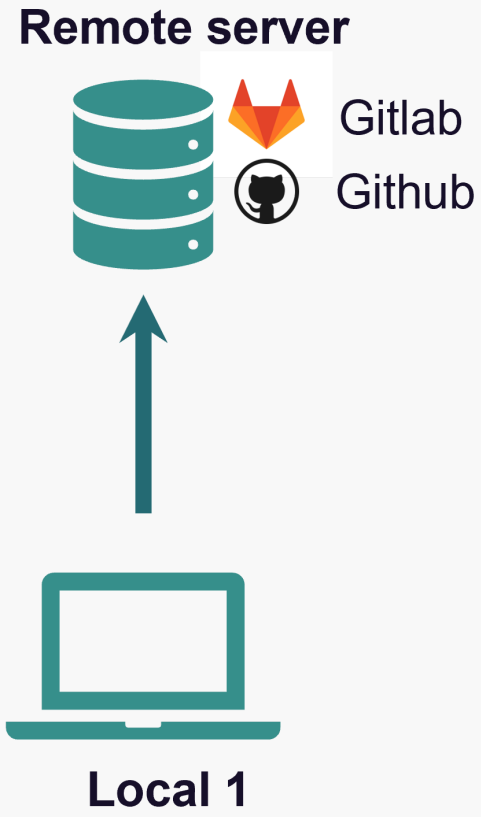
Before we start

Did everyone accept the invitation to the cook book of their partner?

Recap

Basic Git workflow:

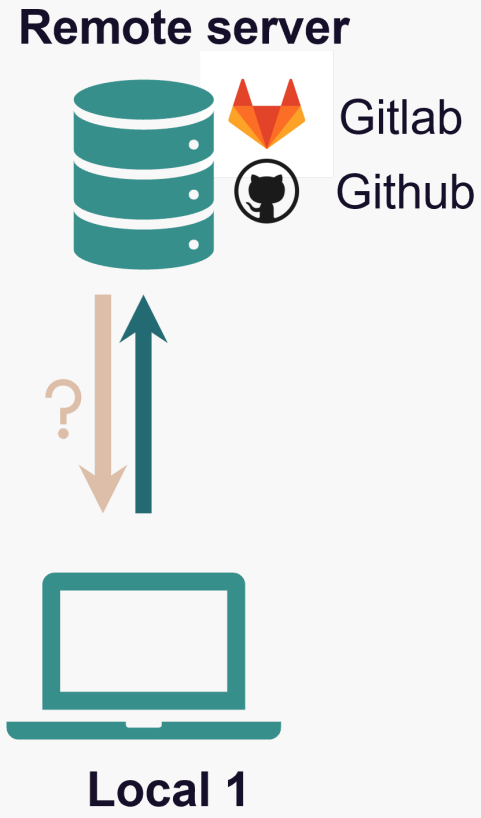
1. **Initialize** a Git repository
2. **Work** on the project
3. **Stage** and **commit** files to the local repository
4. **Push** future changes to the remote repository



Recap

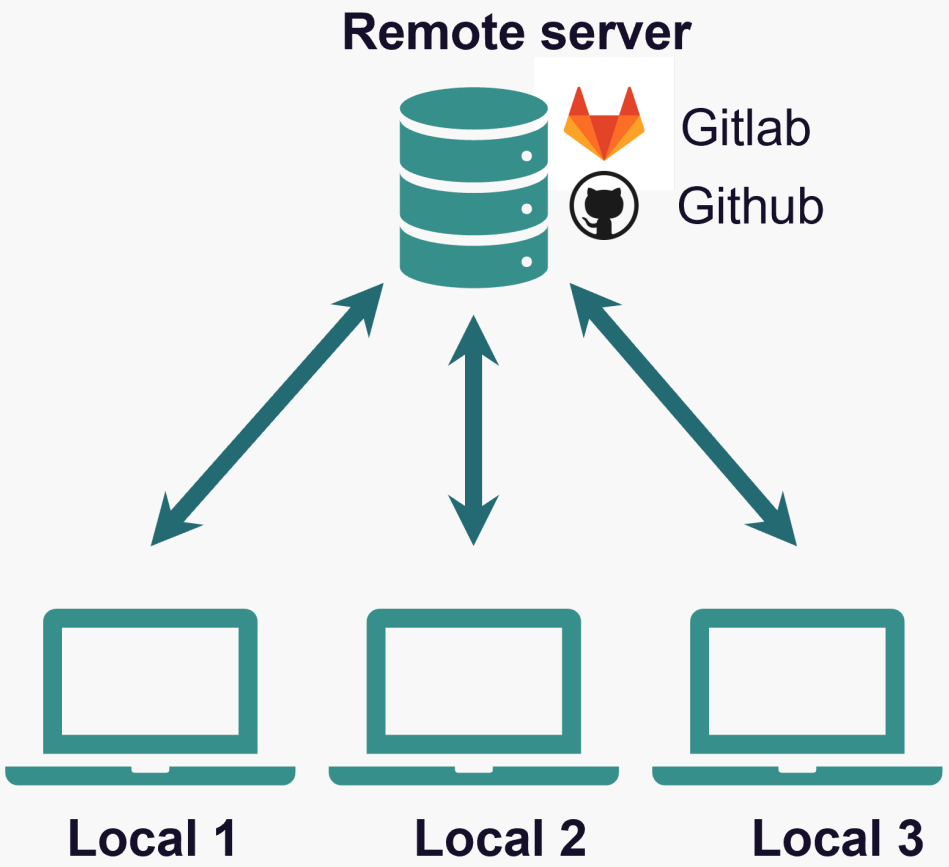
Basic Git workflow:

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



Recap

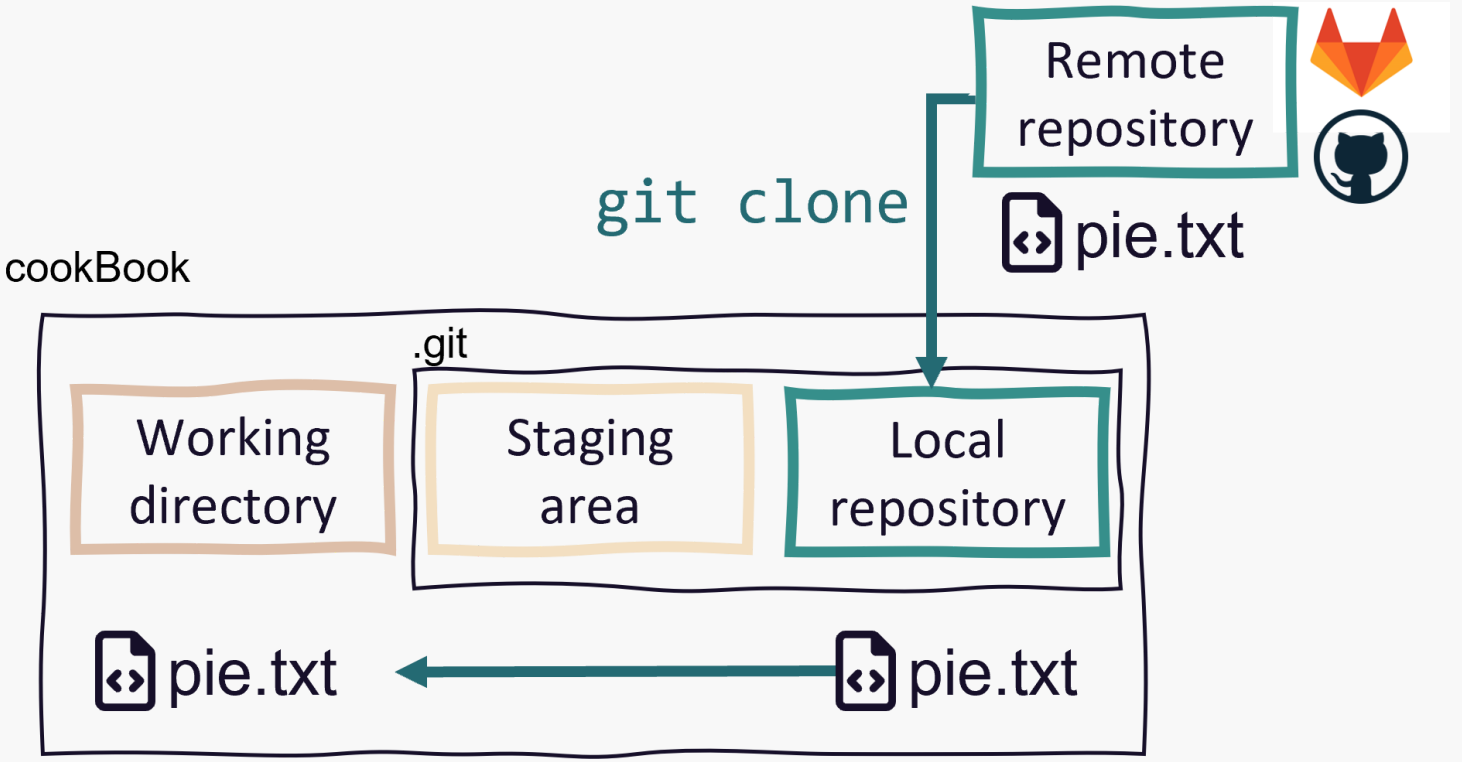
Git is a **distributed version control system**



- Idea: many *local* repositories synced via one *remote* repo
- Collaborate with
  - yourself on different machines
  - your colleagues and friends
  - strangers on open source projects

Get a repo from a remote

- In Git language, this is called **cloning**
  - Get a **full copy** of the remote repo



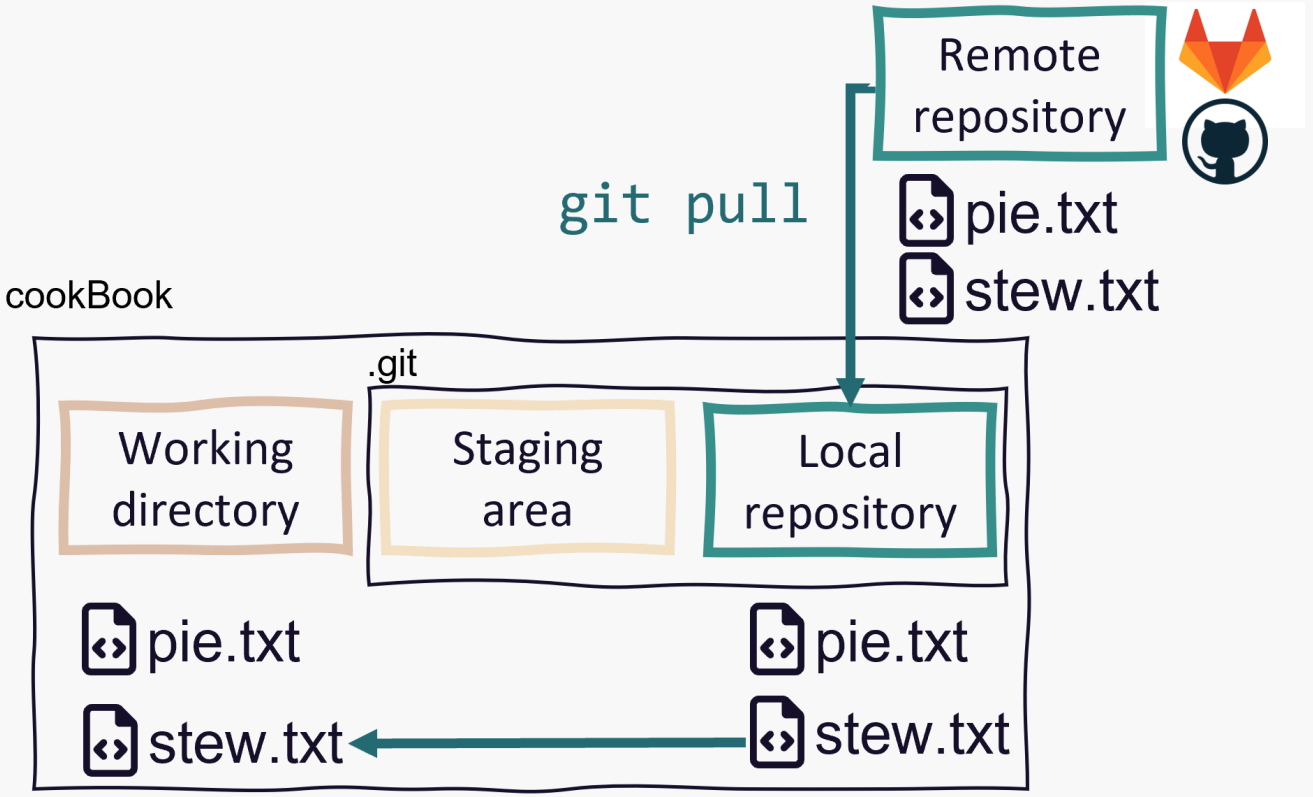
- If the clone is authorized it can also commit and push

Now you

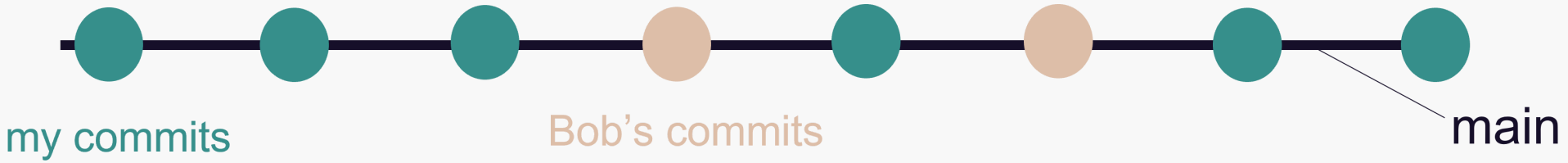
> Clone your partners cookBook repo Complete task 1 “Clone” (5 min)

Get changes from the remote

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

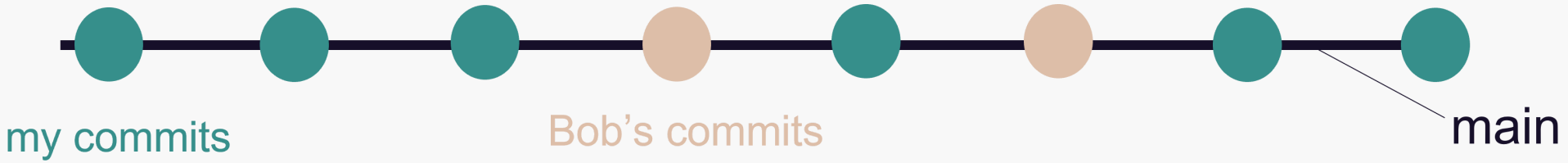


A simple collaboration workflow



- 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**

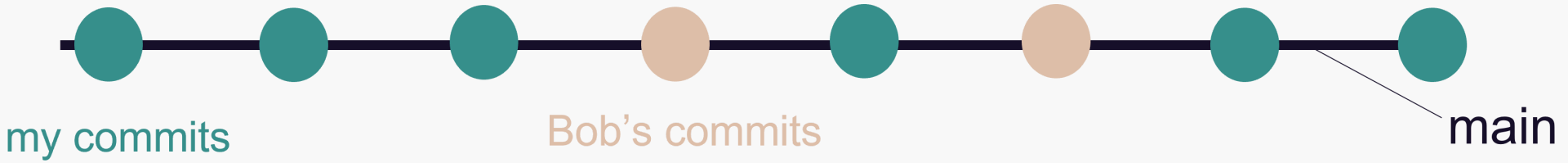
A simple collaboration workflow



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

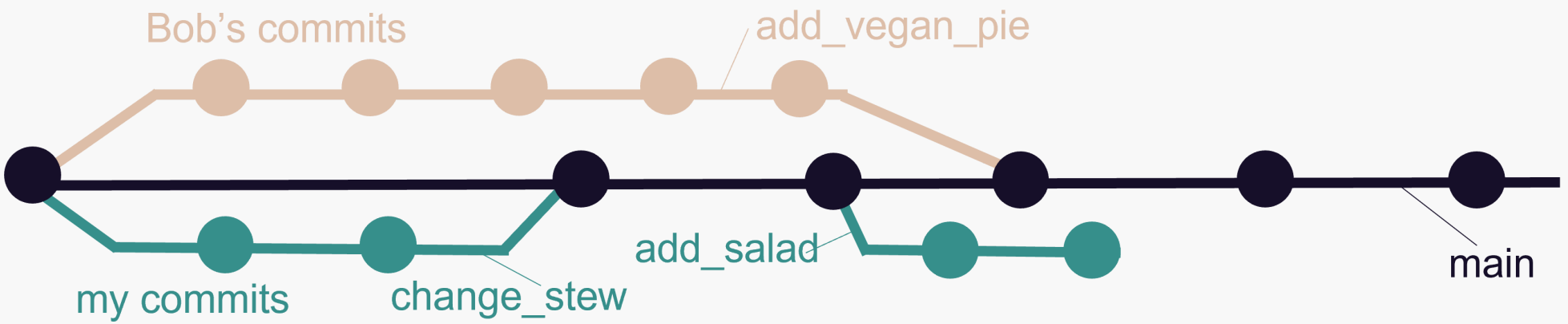
- People push often
  - Conflicts on main
- Not possible to discuss code before integration
- Difficult to just "try something out"
  - Everything goes directly to main

Let's give it a try

- Open a recipe in the cook book of your partner
- Change something in there
- Commit the change and push it

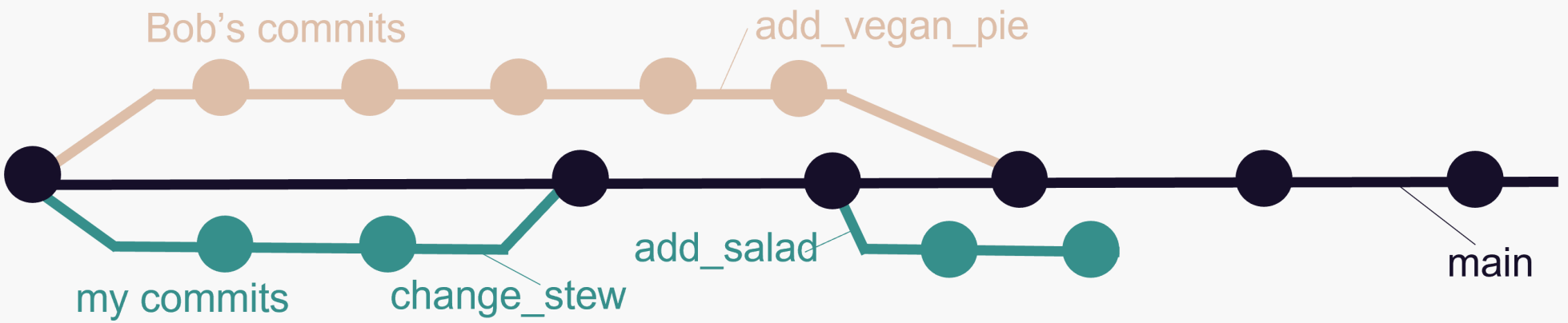
- Now switch to your own cook book
- Pull the change that your partner just did
- Checkout what they changed in the history tab

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
- You can just "try something out"

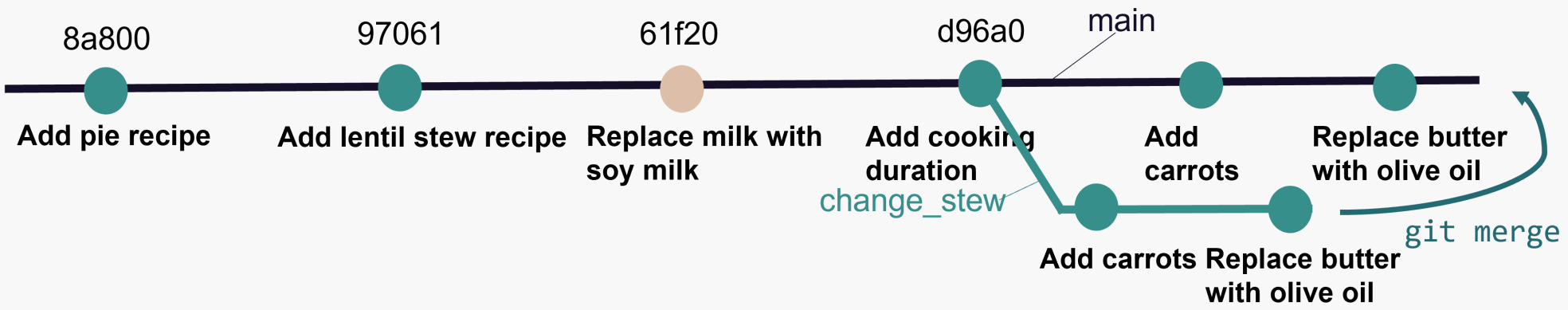
Working on a separate branch

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

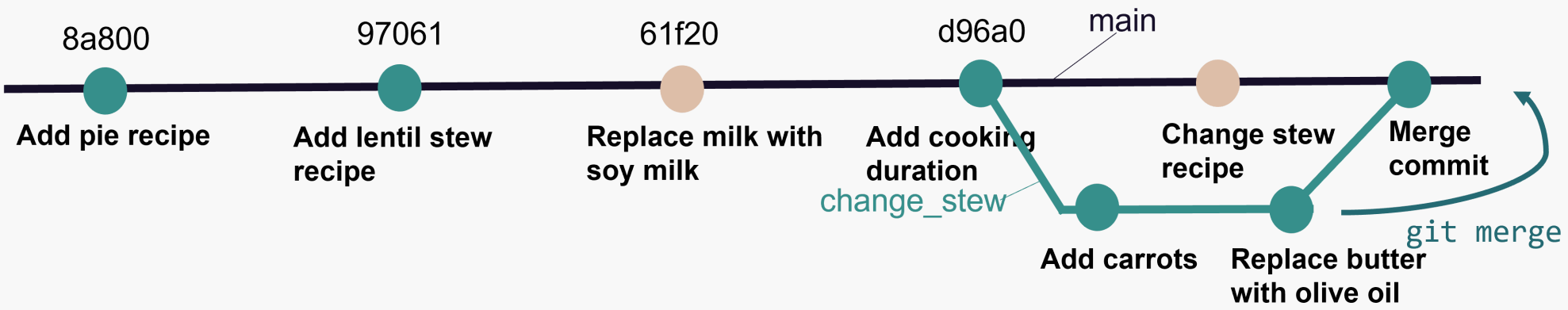
To bring changes to the main branch you need to **merge** them.



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

> Create a branch in your partner's cook book Complete task 2 "Branch and merge" (10 min)

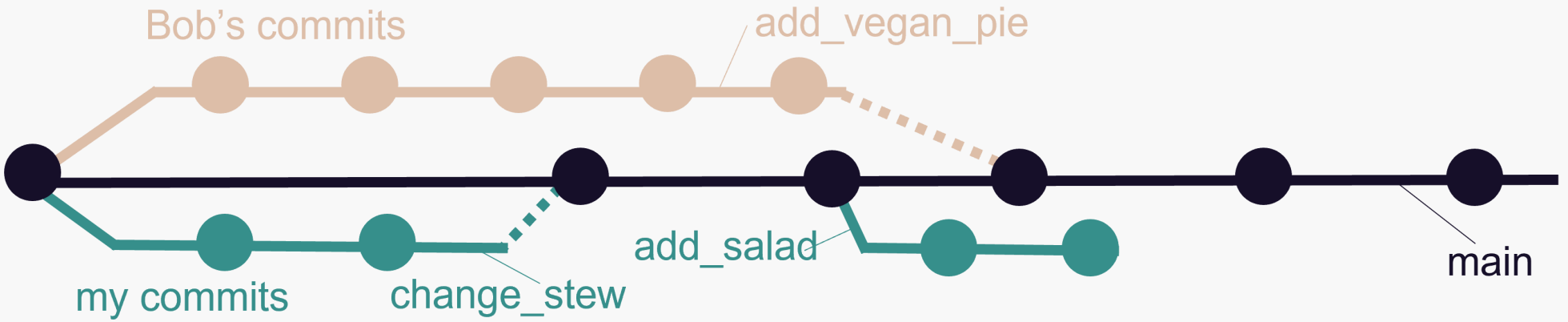
Before we continue

You partner now has pushed some changes to your cook book.

Get the changes from the remote. In Github Desktop

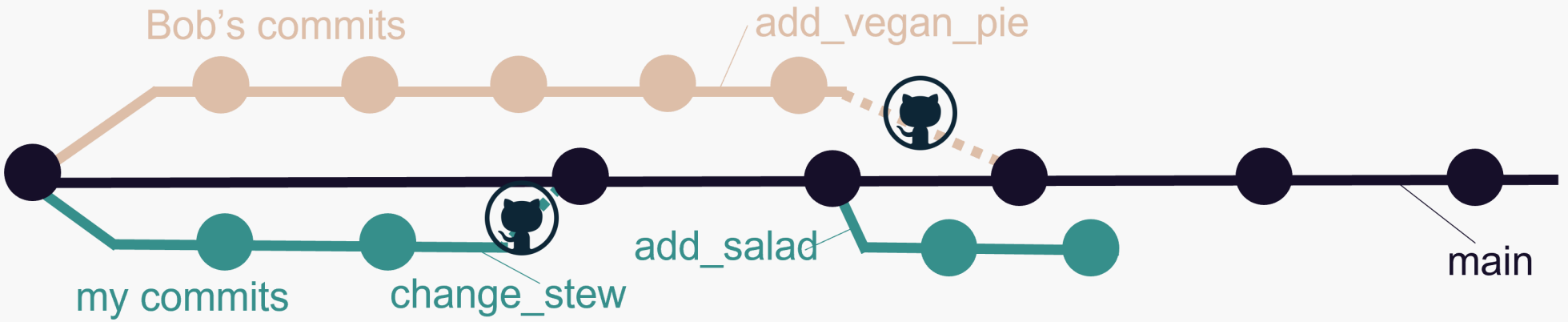
- Switch to your own cook book repository
- Click the pull button (same as push)
- Have a look at the commit history to see your partner's changes

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
- …

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?

A pull request is merged on Github when **everyone agreed on the code**.

Now you

> Create a pull request on your partners repo Review the pull request by your partner on your own repo Complete task “Pull requests” (20 min)

Thanks for your attention

> Questions?

Next week

Idea: Questions and problems often arise when working with the tool.

Until then

Work with Git (~ 2 h), e.g.

- Start using version control for you own projects
- Create a cook book project using the terminal
- Try to use Git from R (see How-To)
- ...

Write down your problems/questions/other Git things you are interested in.

Next week we will - Discuss your questions - Look at other interesting Git things todo - e.g. Use Git from R, VS Code - Use Github to publish