Code and data management with Git
Combining changes by merging

Setting the stage

In this practical, you'll work with the repository your neighbour created during the previous practical. We'll refer to this repository as **N**, and to your own repository from the previous practical as **Y**.

Clone his or her repository to your local machine. Make sure to do this in a separate directory.

After you and your neighbour both cloned eachother's repositories:

1. Add a new commit to repository **Y** on GitLab (so the one your neighbour just cloned).
2. Wait for eachother before continuing.
3. Update your clone of **N** with `git fetch`.

Question: What does the commit graph of your repository look like? Can you draw it on paper?

Hint: Use `git log` with the appropriate arguments.

As an alternative to `git log`, you can also try a graphical viewer such as `gitg` (installed on the course laptops) or `gitk`.

Fast-forward merging

Of course we want the nice commit your neighbour just made in our `master` branch.

Merge the remote `master` branch into `master`.

- *Question:* What does the commit graph of you repository look like now?

A three-way merge

The previous section produced a fast-forward merge.

- *Question:* Can you think of a way to get a three-way merge?

Together with your neighbour, manipulate your repository in such a way that the commit graph contains a merge commit (created by a three-way merge).

- *Question:* What does the commit graph of you repository look like now?

Resolving a merge conflict

Now do the same thing again (a three-way merge), but in such a way the merge step will result in a merge conflict. Resolve the conflict in a way you feel is appropriate.