Gitting Started

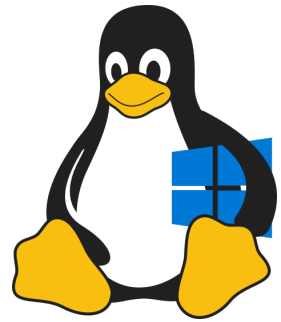
An Introduction to Git

K. Müller[1] J. Verzijden[2]

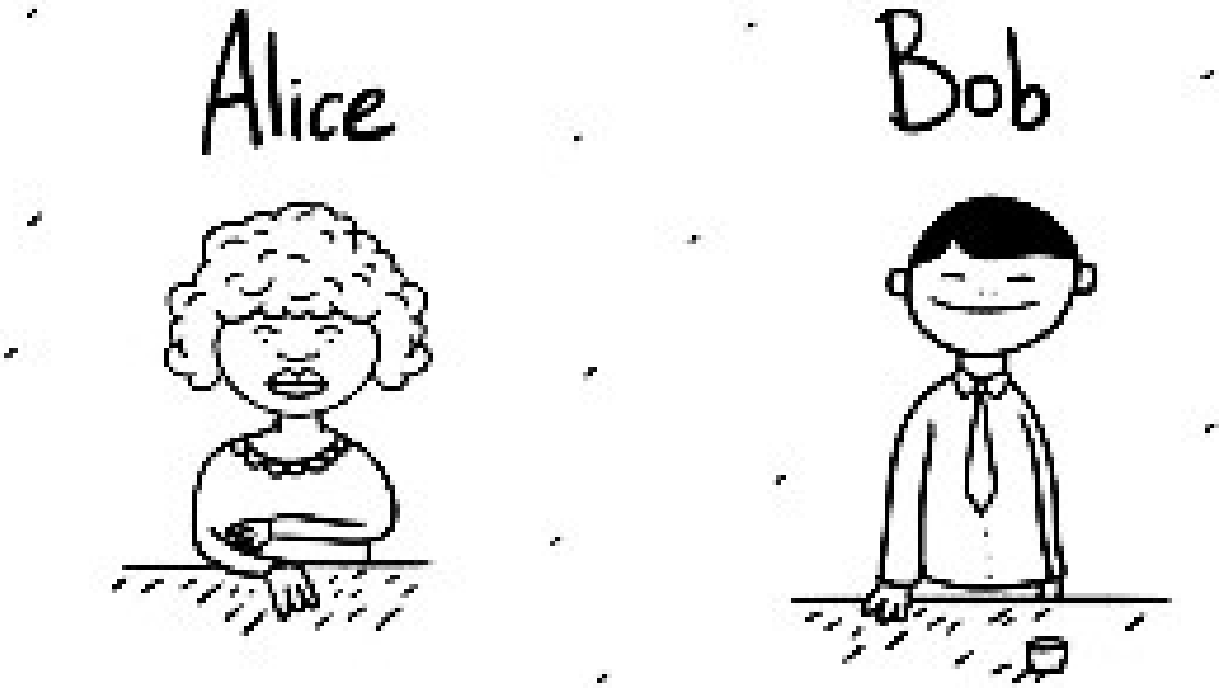
[1]MasterCLASS
E.T.S.V. Scintilla

[2]Scintilla Operator Team
E.T.S.V. Scintilla

June 27, 2023



Let's imagine...



About us

Kasper Müller
kasperm@scintilla.utwente.nl

Johan Verzijden
johanv@scintilla.utwente.nl

Contents

Why Git?

- Solves the problems from the intro.
- It's efficient and fast
- It's the most used VCS[a] in the world

[a] Version Control System

Version control systems used by responding developers:

Name	2015	2017	2018	2022
Git	69.3%	69.2%	87.2%	93.9%
Subversion	36.9%	9.1%	16.1%	5.2%
TFVC	12.2%	7.3%	10.9%	[ii]
Mercurial	7.9%	1.9%	3.6%	1.1%
CVS	4.2%	[ii]	[ii]	[ii]
Perforce	3.3%	[ii]	[ii]	[ii]
VSS	[ii]	0.6%	[ii]	[ii]
ClearCase	[ii]	0.4%	[ii]	[ii]
Zip file backups	[ii]	2.0%	7.9%	[ii]
Raw network sharing	[ii]	1.7%	7.9%	[ii]
Other	5.8%	3.0%	[ii]	[ii]
None	9.3%	4.8%	4.8%	4.3%

The origin story

- Linus Torvalds[a]
- 2005
- VCS for the Linux kernel
- *The stupid content tracker*

[a] Source of the photo: WikiMedia

Concepts of Git

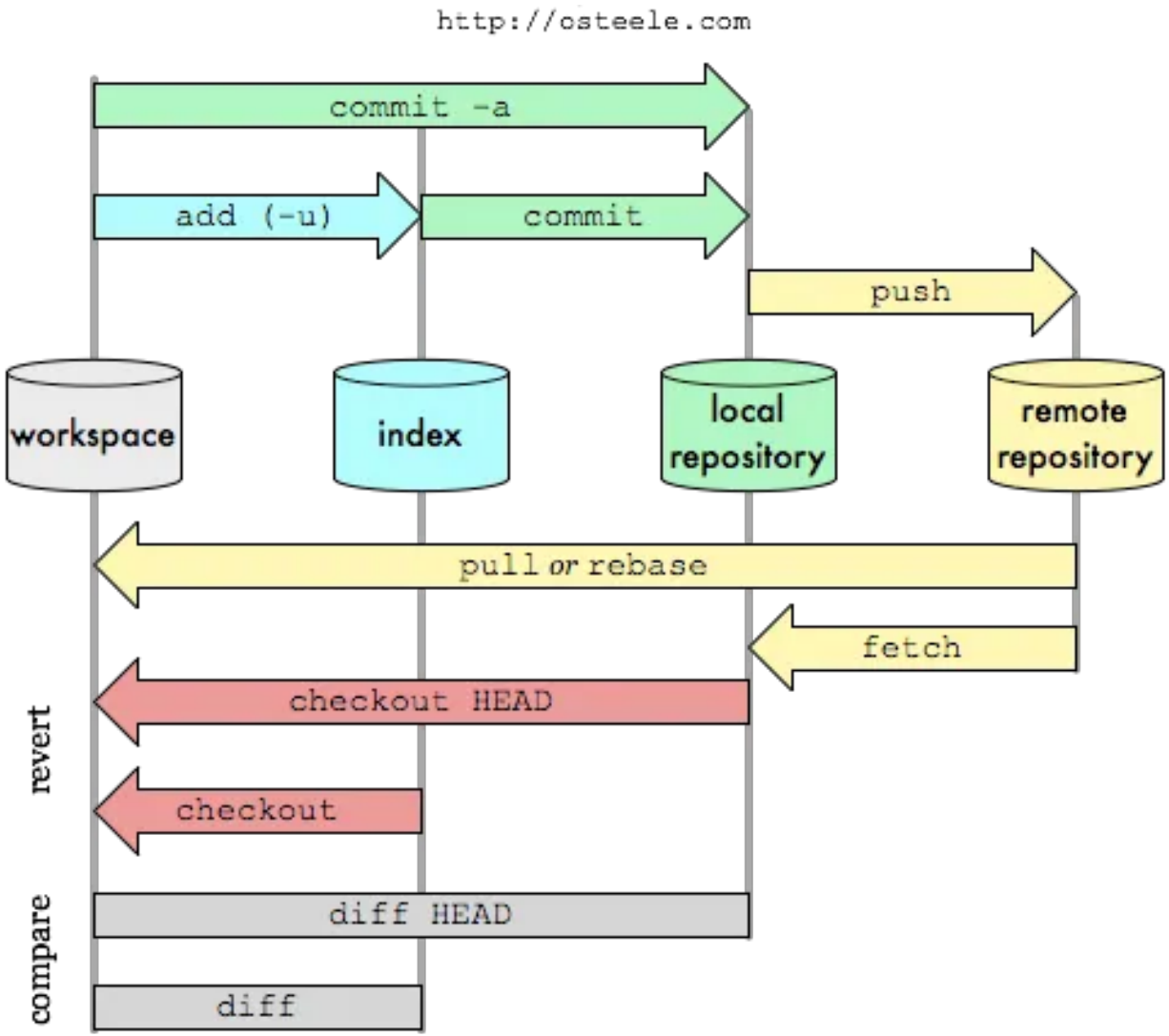
- Repository (repo)
  - A location for storage[1]
  - In this case, storage of files
  - Just a folder on your computer, e.g. `/home/johanv/my-repo`
- Commit
  - Description of changes
  - Metadata: author, timestamp, message, parent
  - Hash: Its unique identifier
- Branch
  - A list of commits belonging together
  - A tag points to the head of the branch (the last commit)
- Remote
  - The same repository in a different location
    - Really remote (`https://github.com/instructure/canvas-lms`)
    - Locally remote (`/home/kasperm/some-repo/`)

[1] *repository* on Wiktionary

Principles of Git

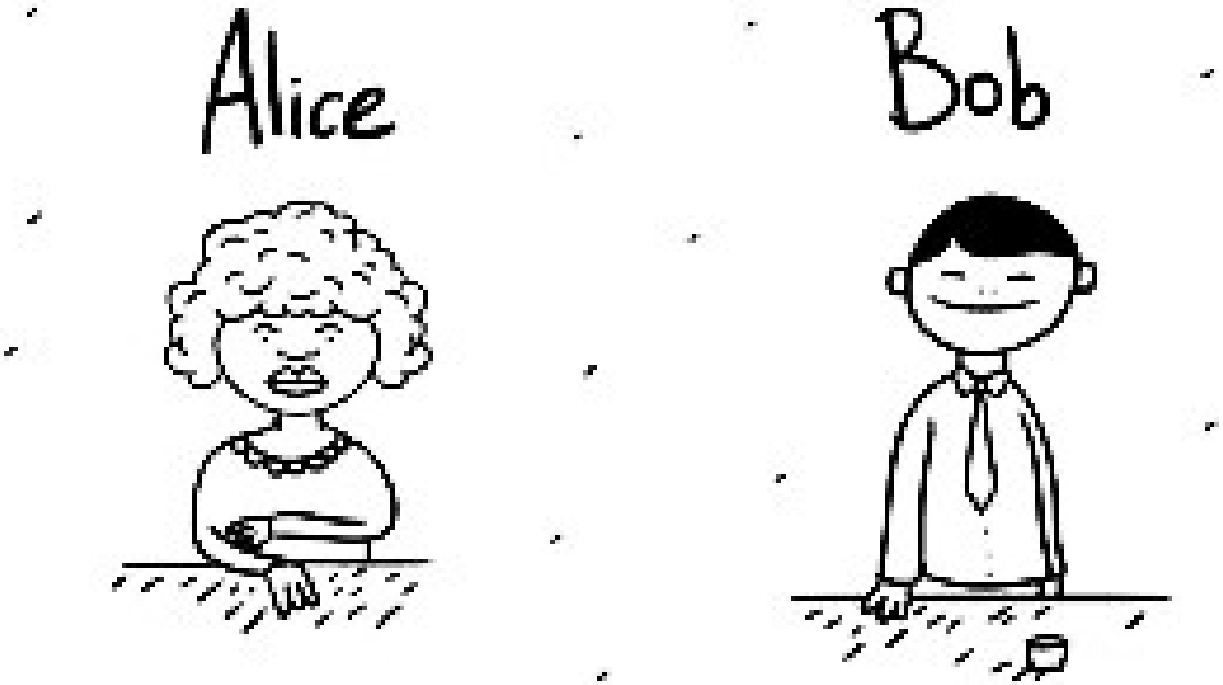
- Recording small sets of changes, not versions
- Meant for text files, not for binary files
- Meant for source files, not compiled files
- Make a new branch for each feature
- Branch names should be descriptive
  - ✓ *create-home-page*
  - ✗ *branch15*
- Commits should contain small, logical changes
- Commit messages should be descriptive
  - ✓ *Fix styling of logout button*
  - ✗ *fix*
  - ✗ *Meep*
- Commit messages should be in imperative tense
  - ✓ *Add page layout*
  - ✗ *Added page layout*
  - ✗ *Adding page layout*

Git Workflow



Let's take another look (at the situation)

Let's see how Git actually solves the problem in the introduction.



Tracking source files - Gitignore

In git you only want to track source files. Binaries or compiled files don't work well with version changes and can be different on other pcs.
Solution: don't add the changed source files, permanent solution: `.gitignore`

Installation - GUI

Some options:

- Pure Git GUI[2] clients:
  - git-gui & gitk (comes with the Git CLI)
  - Github Desktop (only for Mac & Windows)
  - gitg (only for Linux & Windows)
  - See https://git-scm.com/downloads/guis for more
- Most IDEs have a Git GUI built-in

Nice and all, but we will use the CLI[3].

[2] Graphical User Interface

[3] Command Line Interface

Installation - CLI

- Windows
  - Using *winget*:

    ```
    winget install --id Git.Git -e --source winget
    ```
  - Using an installer or portable version:
    https://git-scm.com/downloads/win
- Linux
  - Don't, you already have it (-:
  - Through your package manager:
    - `apt install git`
    - `dnf install git`
    - `pacman -S git`
- MacOS
  - Using Homebrew:
    - `brew install git`
  - Install Xcode, that ships with Git

Installation successful?

```
johanv@my-machine: ~$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
```

Break

Install Git!

Get out your laptop and install Git on it if you haven't already, we will start using it directly after the break

Practical - Goals

- Show the typical flow of working with git.
- Show you most important commands and their uses in a short time

Configuration

How:

- Using an editor:
  `git config --global --edit`
- Using commands:
  `git config --global user.name 'Johan Verzijden'`

What:

- Name (*user.name*)
- Email address (*user.email*)
- Default branch name (*init.defaultBranch*)
- Fast-forward (*pull.ff*)

Read `git help config` or `man git config` for all configuration options

Practical - Documentation

Feel free to work on your own or in pairs. If there are many questions on a certain topic we will do a class-wide explanation.

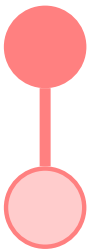
Starting a new project

```
~$ mkdir my-portfolio
~$ cd my-portfolio
~/my-portfolio$ git init
```

Make some changes, do your first commits

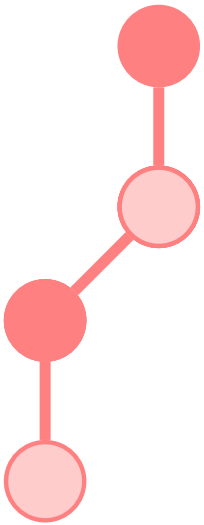
```
# Create index.html and style.css files
# Add standard HTML layout
~/my-portfolio$ git add index.html style.
    css
~/my-portfolio$ git commit -m 'Initial
    commit'
# Add heading and welcome text
~/my-portfolio$ git commit -am 'Add
    heading and welcome text'
~/my-portfolio$ git log
```

Create a new branch and switch to it



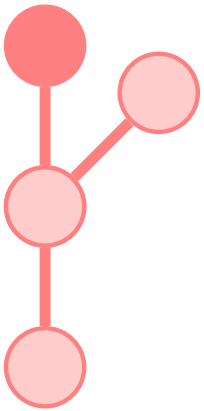
```
~/my-portfolio$ git switch -c add-git-
    project-info
#### Alternative
~/my-portfolio$ git branch add-git-
    project-info
~/my-portfolio$ git checkout add-git-
    project-info
```

Commit and switch back



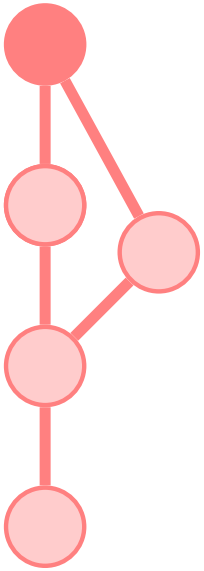
```
# Add info about your latest project
~/my-portfolio$ git commit -am 'Add info
    about latest project'
# Add info about another project
~/my-portfolio$ git commit -am 'Add info
    about another project'
# Oh wait, I don't want this in this
    branch. Let's go back
~/my-portfolio$ git log
~/my-portfolio$ git reset <previous
    commit hash>
~/my-portfolio$ git switch main
```

Commit on main branch



```
# Add contact details
~/my-portfolio$ git commit -am 'Add
    contact details'
```

Merge the two branches

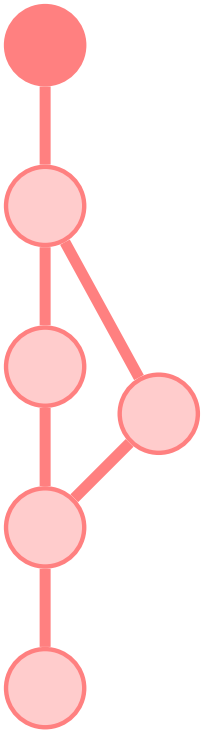


```
~/my-portfolio$ git merge add-git-project
    -info
# Merge conflict!
```

Resolve the merge conflict

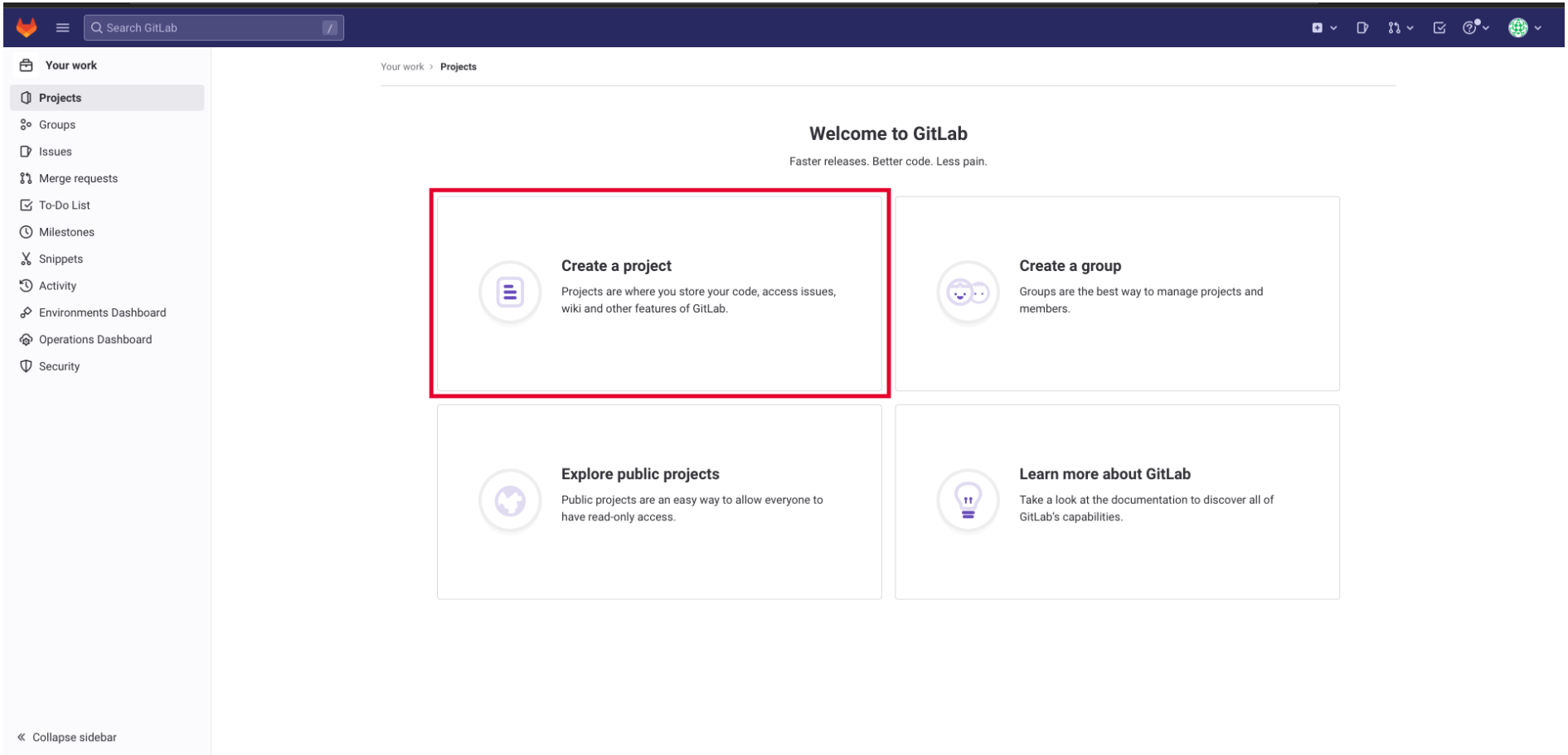
```
# Go into the file and resolve the conflict
~/my-portfolio$ git add index.html
~/my-portfolio$ git merge --continue
```

Change the text colour and commit

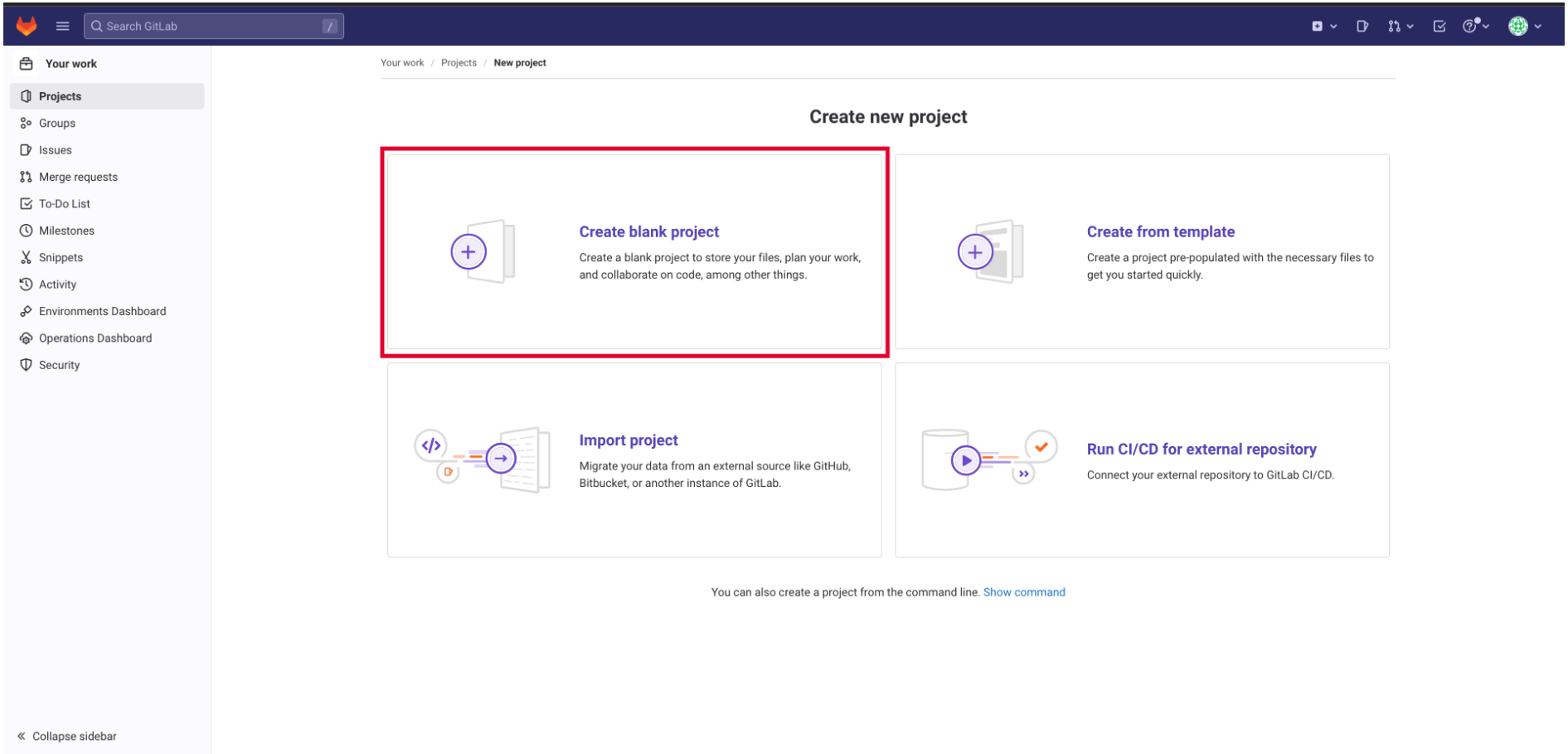


```
# Make the text and background colour the
    same
# (This is on purpose, will become clear
   later)
~/my-portfolio$ git commit -am 'Give the
   text a great colour'
```

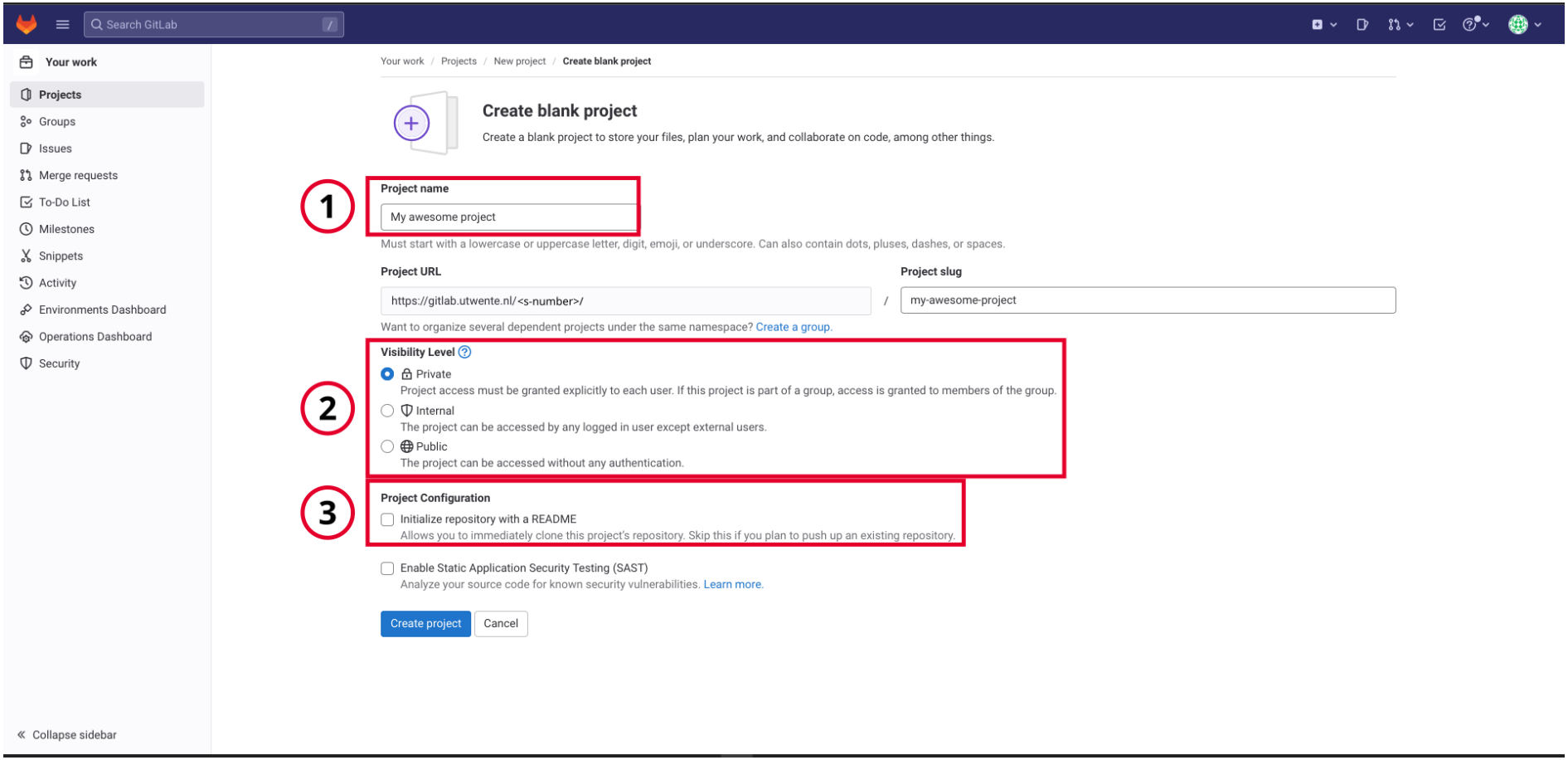
Create repository on Gitlab



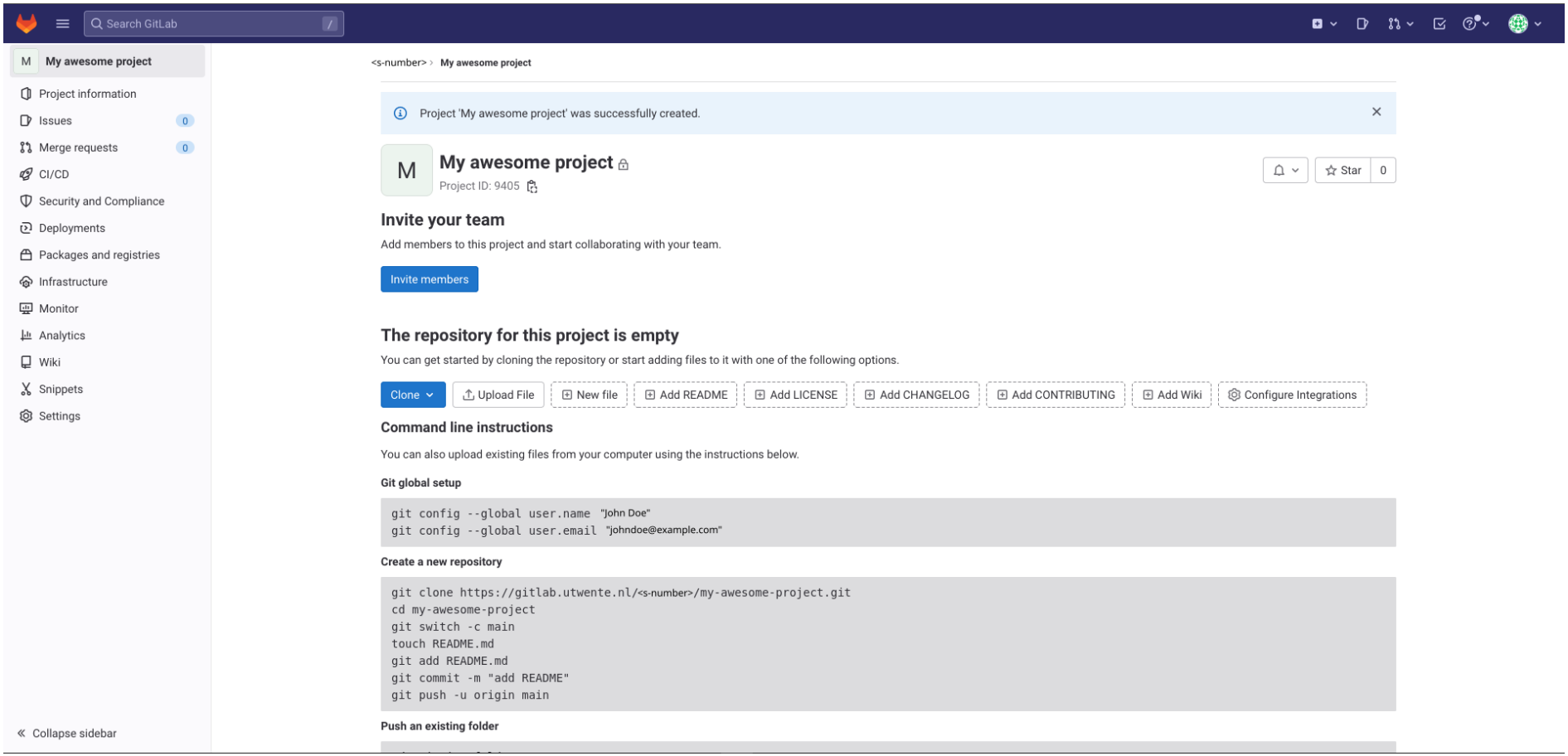
Create repository on Gitlab



Create repository on Gitlab



Create repository on Gitlab



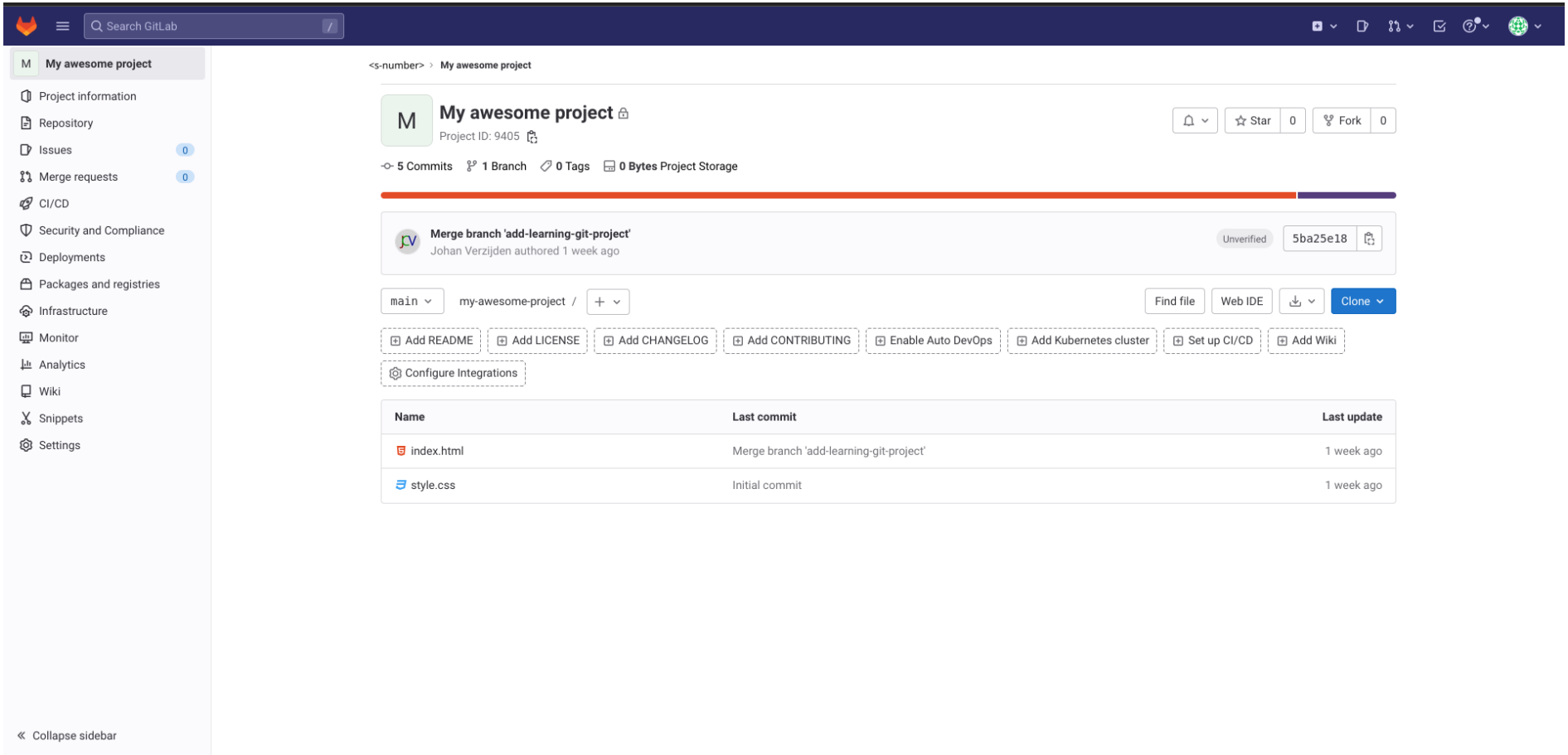
Set up remote tracking locally

```
~/my-portfolio$ git remote add origin https
    ://gitlab.utwente.nl/<s-number>/<project-
    name>.git
```

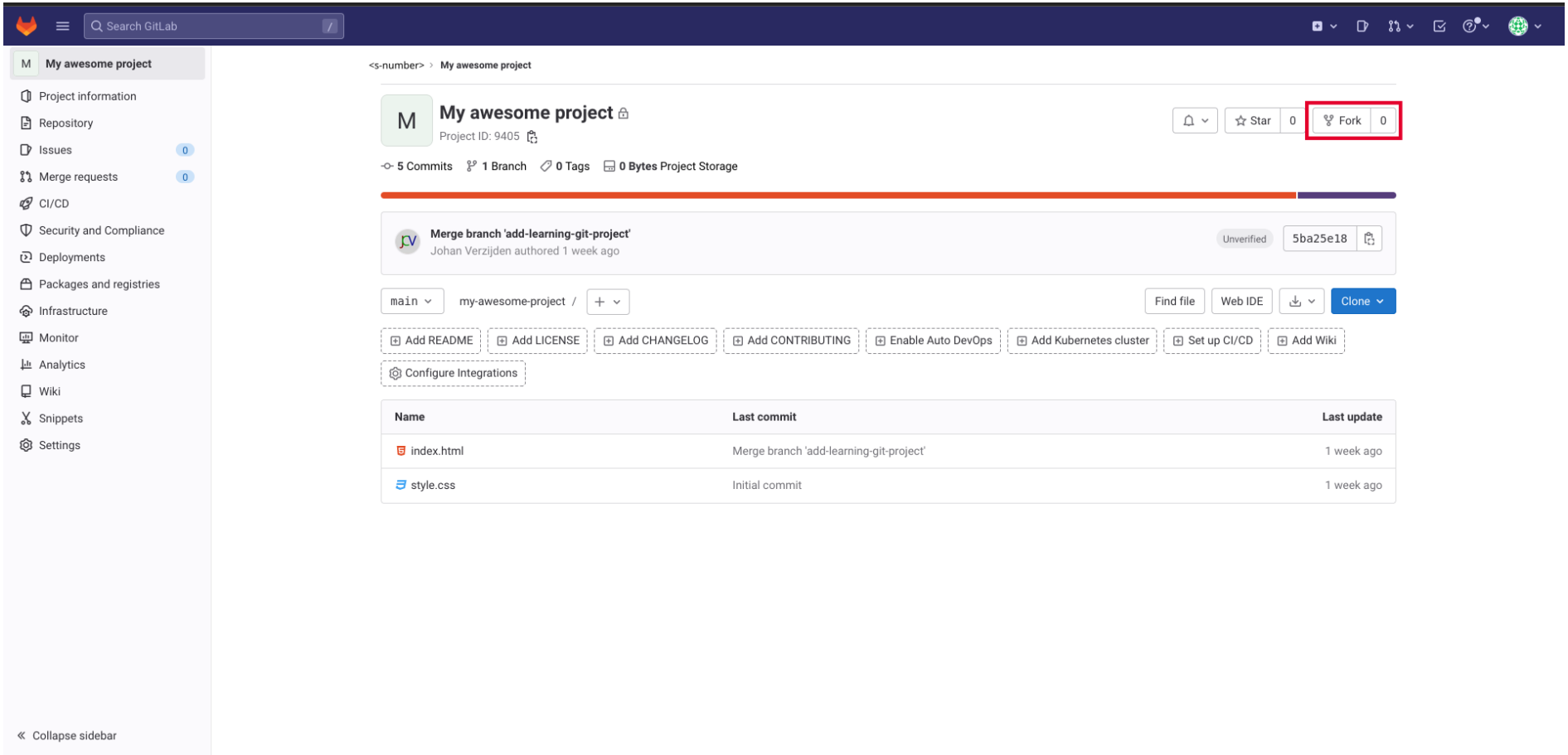
Push to remote

```
~/my-portfolio$ git push -u origin main
```

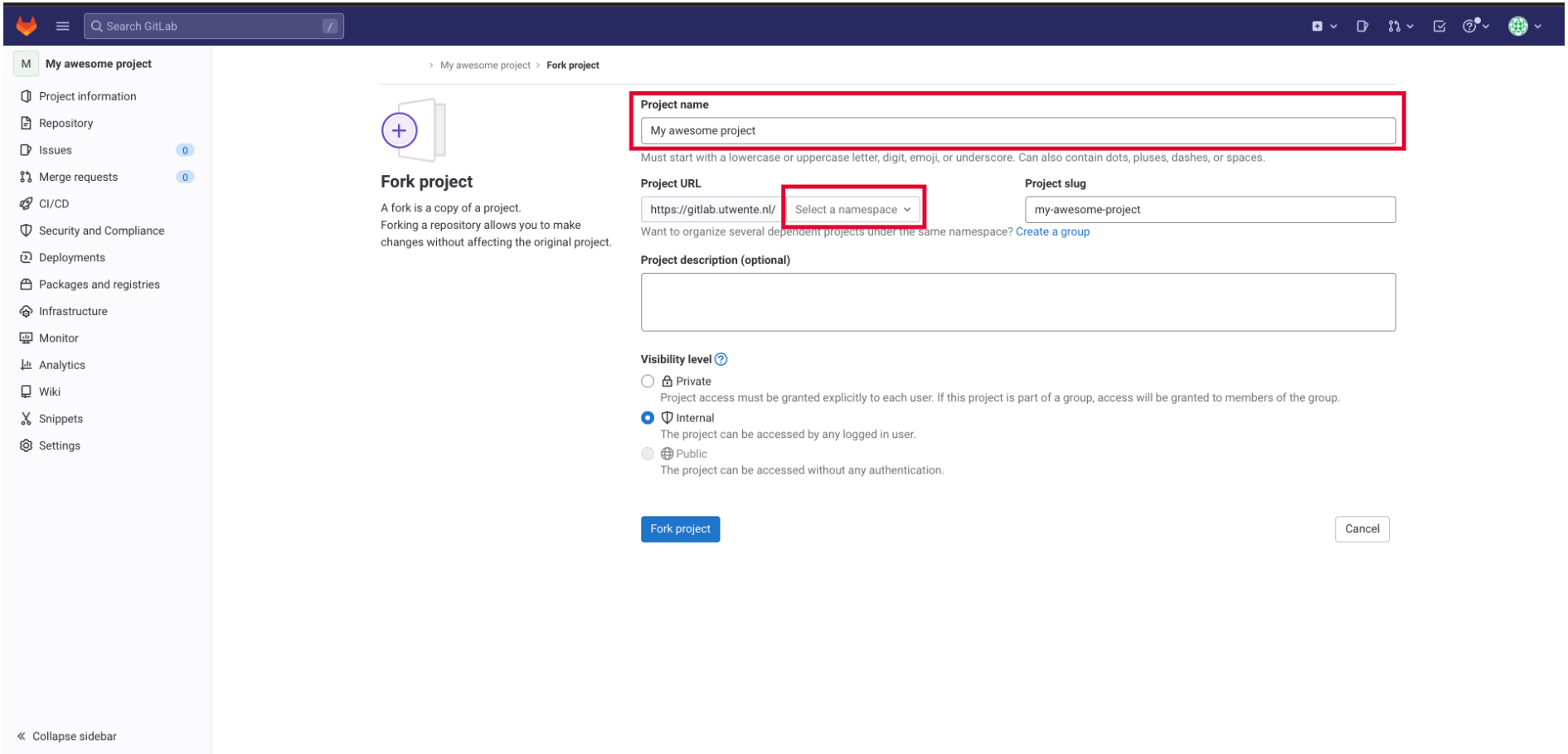
Look at repository on Github/Gitlab



Fork from neighbour



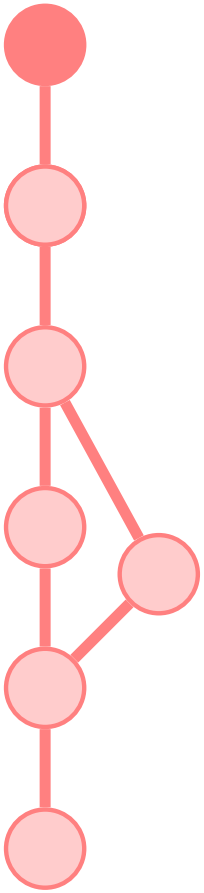
Fork from neighbour



Clone fork to local

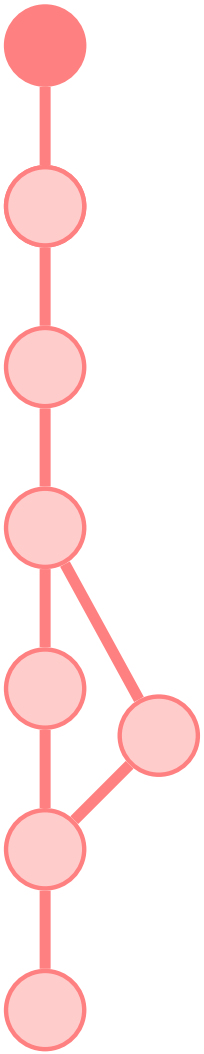
```
~$ git clone https://gitlab.utwente.nl/<s-
   number>/<forked-project-name>
~$ cd <forked-project-name>
```

Revert last commit to fix the text colour



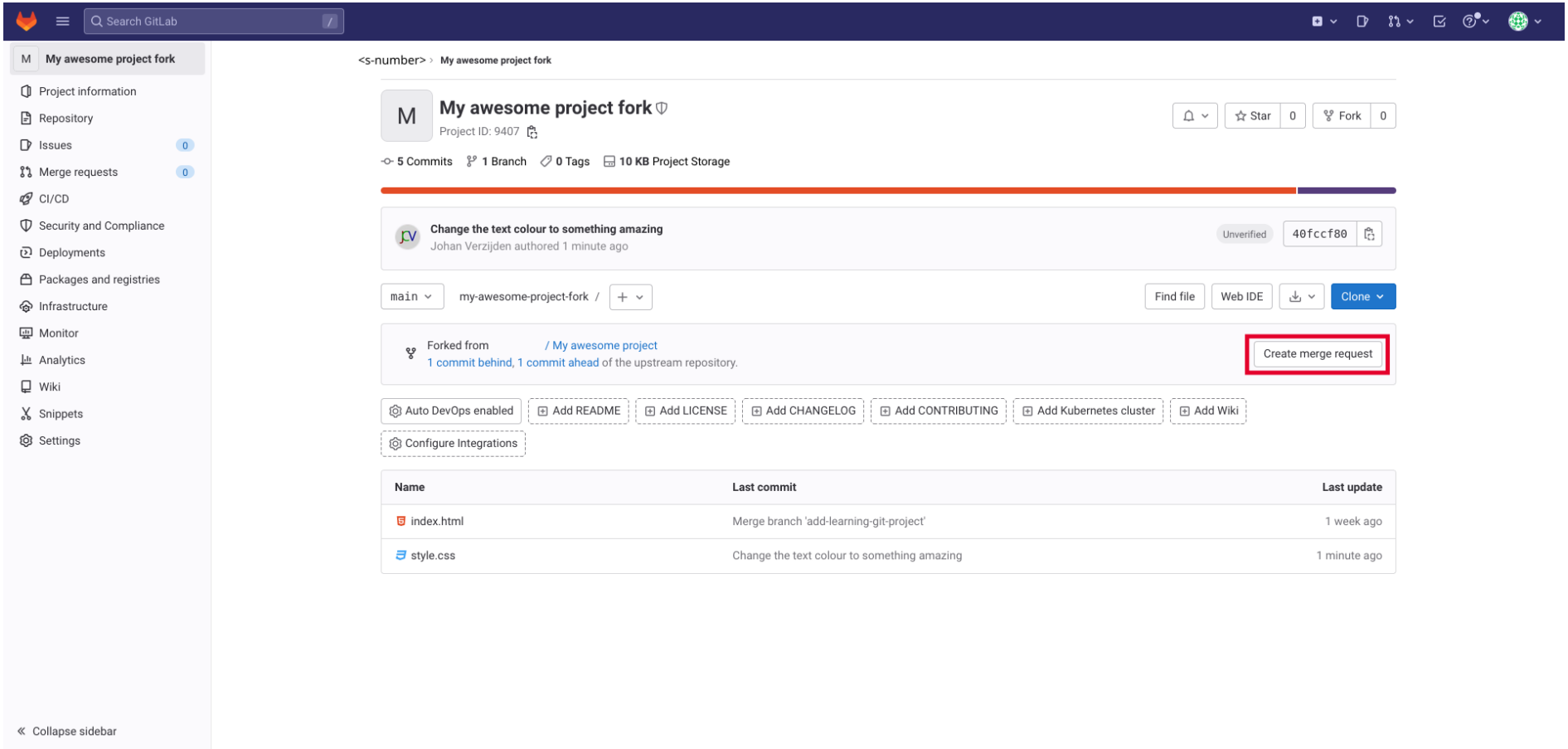
```
~/<forked-project-name>$ git revert <
    commit hash>
```

Do a different commit, push to remote again

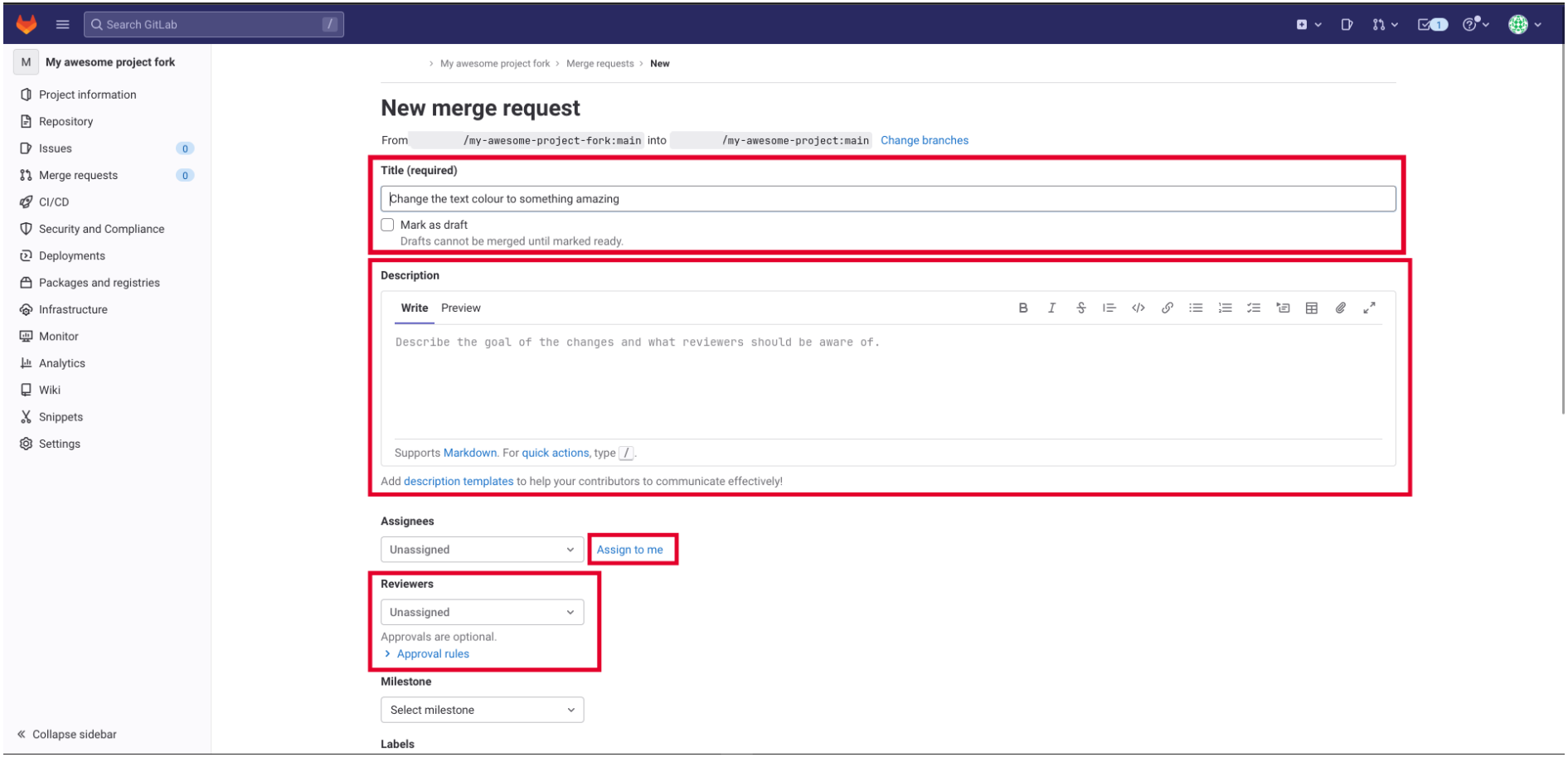


```
# Change text to a different colour
~/<forked-project-name>$ git add .
~/<forked-project-name>$ git commit -m '
    Change the text colour to something
    amazing'
~/<forked-project-name>$ git push
```

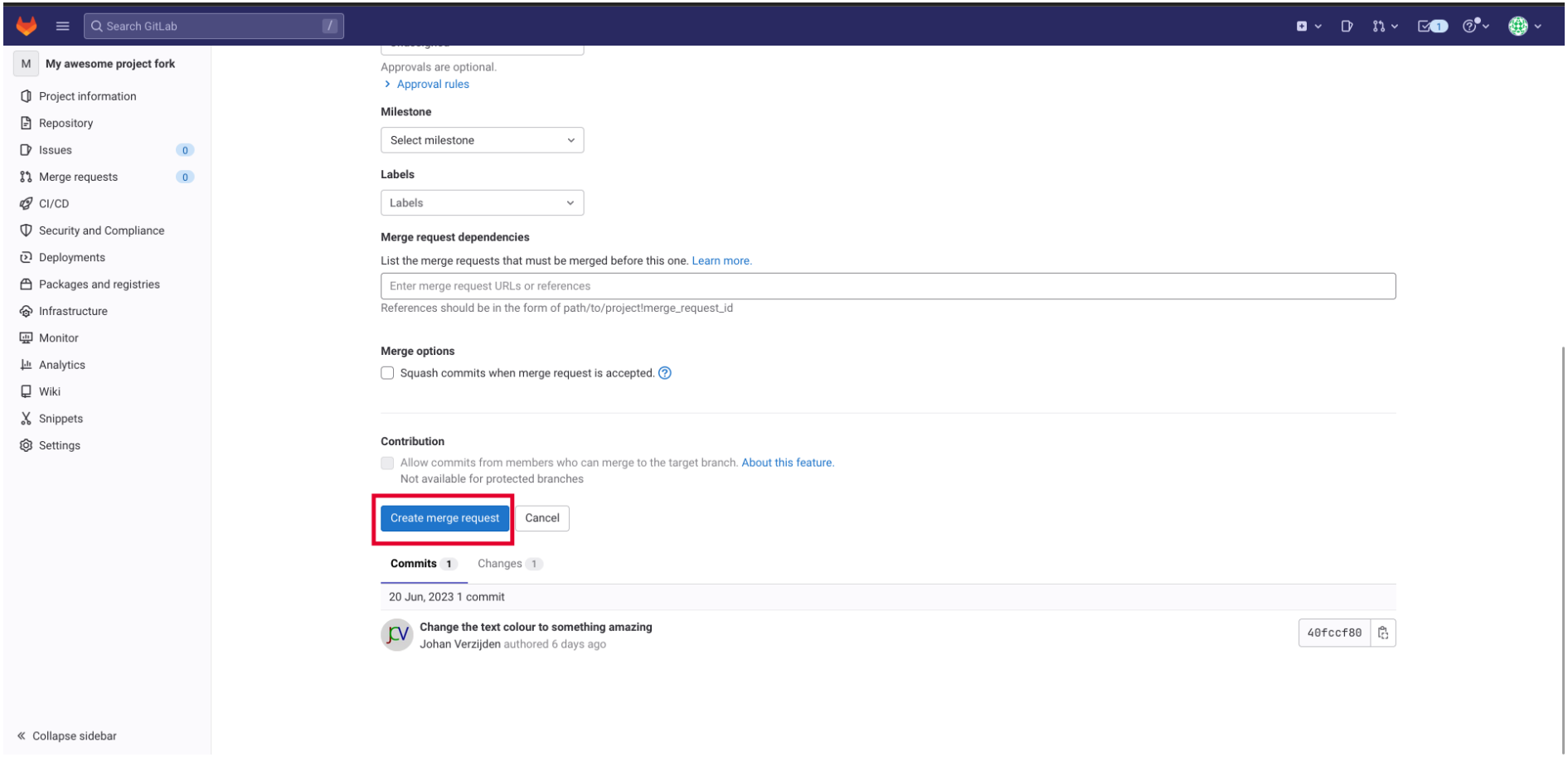
Open merge request



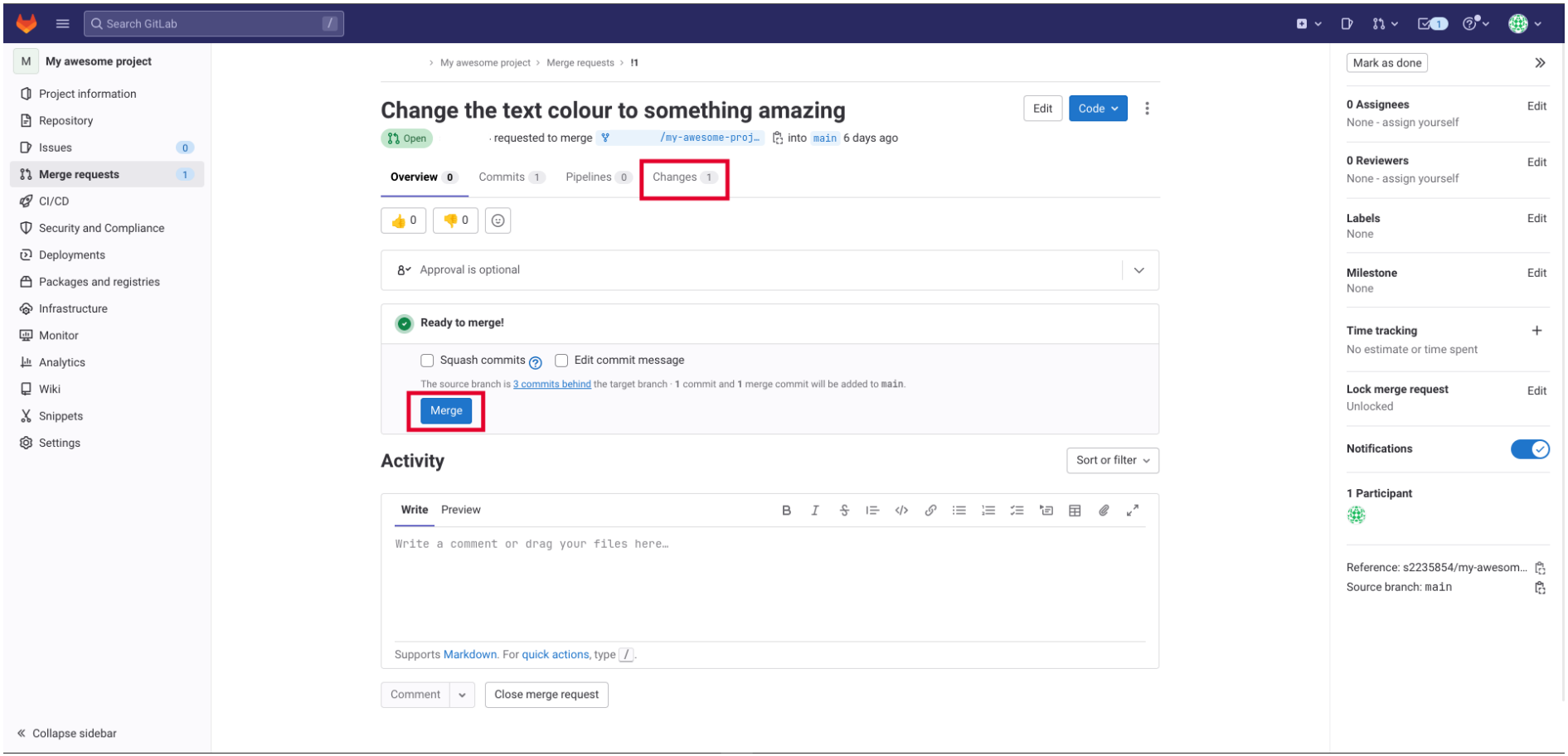
Open merge request



Open merge request



Review and accept merge request



Look at Blame

```
~/my-repo $ git blame index.html
```

Advanced topics

- Stash
- Rebase
- Cherry-pick

Common Git Commands - Cheat sheet

`git ...`

- `init`
- `add [<filenames>]`
- `commit [-m '<message>']`
- `switch [-c] <branch name>`
- `log [ --graph --oneline]`
- `push`
- `pull`
- `fetch`
- `clone <path/URL>`
- `blame <file> [--color-by-age]`
- `revert <commit>`
- `reset [--hard] <commit>`

Sources for future reference

- `git help <command>`
- `man gittutorial`
- gitimmersion.com

Contact details

Think of a question later on? Feel free to reach out to us!

MasterCLASS
masterclass@scintilla.utwente.nl

Kasper Müller
kasperm@scintilla.utwente.nl

Johan Verzijden
johanv@scintilla.utwente.nl