Assignment 1: init with git

Contents

- Setting up
- Your first git repository
- Recovering old versions of files
- Dealing with conflicts in Git
- Creating A Pull Request
- Checklist before submitting

Total points: 5

updates

- added note about where to find info about setting up SSH
- added more detail about pulling from divergent branches in the conflict task

Setting up

If you haven't already, install git.

You should have a private repository for this course. It will be listed as `in3110-yourusername` at https://github.uio.no/IN3110.

Note: this repository is created by a script that runs periodically, but can only create your repo after the first time you have logged into https://github.uio.no, so make sure to do that first, even if you aren't ready to work on the assignment.

If you don't have a repo within one day of your first login to github.uio.no, let us know at in3110@simula.no.

Clone this repository. Now, create an `assignment1` folder in your git repository. The deliveries for Assignment 1 should be added there.

Skip to main content

You will need to set up credentials in order to clone your private repository. If you haven't set these up yet, you will see a message like:

```
git@github.uio.no: Permission denied (publickey).
fatal: Could not read from remote repository.
```

This is the first exercise! You can check the GitHub SSH documentation about SSH with git and GitHub in general, and the UiO documentation about github.uio.no in particular.

Your first git repository

Task 1 Add and commit a text file **(1 point)**.

Add a text file to the `assignment1` directory. The text file should be named `myfirstcommit.txt` and contain the string “This is my first commit.”. Push the commit to GitHub.

Note: Remember to always leave a descriptive commit-message.

Tip: When you commit code from the command line, git will open a text-editor for you to write the commit message in. You can change the text-editor to e.g. `nano` by writing

```
git config --global core.editor "nano"
```

or for VS Code:

```
git config --global core.editor "code --wait"
```

Tip: Use the -m flag to write the commit-message directly on the command line, without opening an editor.

```
git commit -m "This is my first commit."
```

Recovering old versions of files

With this task, you will learn how to recover old versions of a file.

Skip to main content

1. Add, commit, and push a new file called `friendly_greeting.txt` to your repository. The file should contain a friendly greeting.
2. Change the file content so the greeting is less friendly, and add, commit, and push the modified file.
3. Since the greeting is no longer friendly you should ask git to change the name of the text file `friendly_greeting.txt` to `less_friendly_greeting.txt`. Use the command

   ```
   git mv friendly_greeting.txt less_friendly_greeting.txt
   ```

4. To fetch the old version, first use

   `git log`

   to get a list of the commits, and identify the commit containing the old version of the greeting. Note down its commit hash (the string of letters/numbers after "commit "). Then use

   `git checkout COMMITHASH -- friendly_greeting.txt`

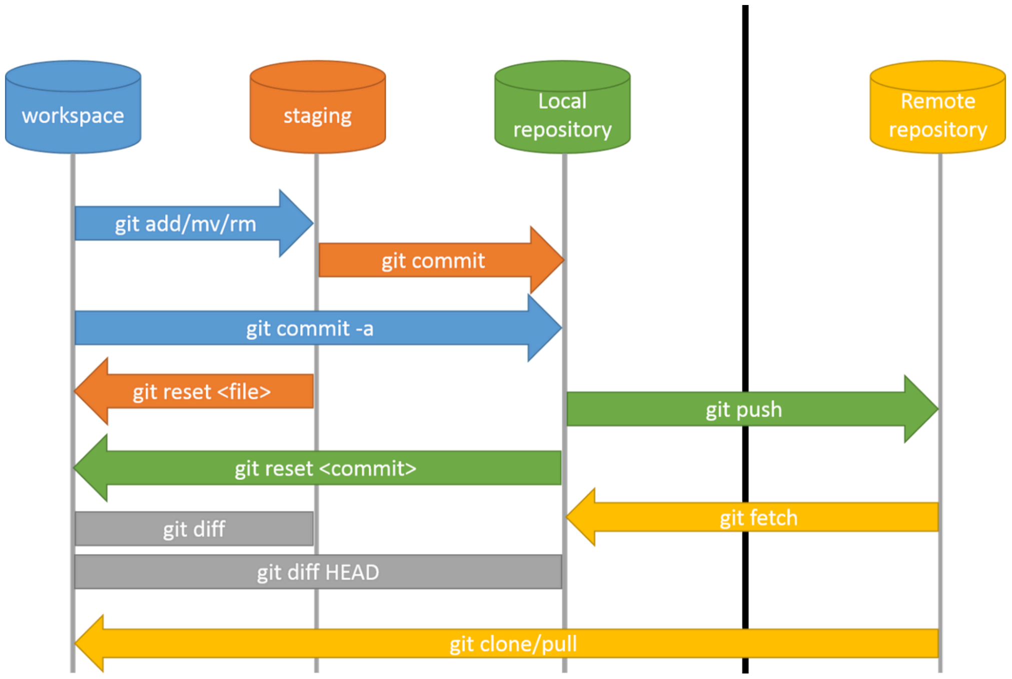
   where `COMMITHASH` is the commit hash you found using `git log`. This will recover the friendly version `friendly_greeting.txt` from the commit you chose, and `git add` it for you.

5. Finally, use `git commit` to make a commit which recovers the friendly version, and push this commit to GitHub. Now you will have a friendly and a not friendly greeting version in your repository.

Dealing with conflicts in Git

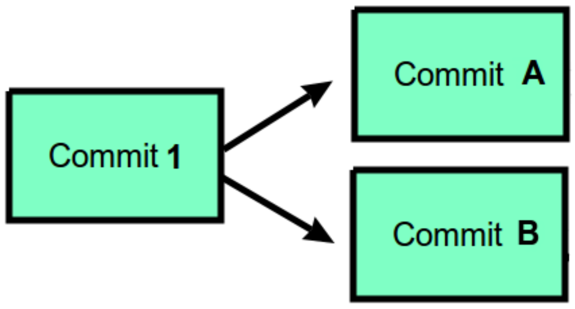
This exercise will use a lot of words to explain a common "error" which frequently happens when git is used for collaboration.

Skip to main content



When working on a project, it is customary to make a sequence of commits of your code and push these to the remote repository. In git, each of these commits will go sequentially "on top of" another commit.

Now, let us assume the repo already contains Commit 1. You and a friend then work on the repo separately. Your friend makes commit A and pushes it to the remote repository. You then make commit B and also attempt to push to the remote repository. Git will then throw an error.

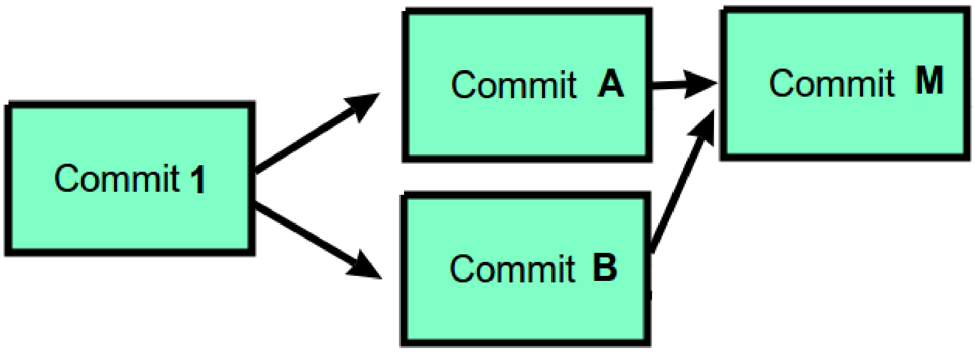


This error will notify you that commit A was made "in between" commit 1 and the commit you are trying to make (commit B), and so won't allow you to push until you have done a `git pull` to get commit A.

While it may have been possible to just pretend commit B was really supposed to go on top of A - maybe A and B are modifying separate files - git won't do that by itself, as this might have unintended side-effects.

When you do `git pull` git will see that you are trying to `pull` commit A which goes on top of commit 1. However, your local copy of the repository already has a commit B on top of commit 1, so completing the `git pull` means you have to `merge` first. This means that you have to make a new `git commit` M which does what commit B did, but on top of

Skip to main content



In some cases, it is clear how to merge two commits and git will do it automatically for you.

In other cases, it is unclear how to merge the two commits. Git will then raise a *merge conflict* that you have to resolve manually.

Dealing with merge conflicts

Let’s simulate such a scenario and fix the resulting merge conflict.

Task 3 Resolve a merge conflict following steps 1-4 below **(1 point)**:

1. Add, commit, and push a new file `gitconflict.txt` with the contents

   ```
   Here's a line

   Hello world!

   Here's another line
   ```

   to your git repository.

2. Copy the directory on your local machine containing your git repository to a different directory, e.g.

   ```
   cp -r ../ ../../my-repo-copy
   ```

   or run your original `git clone` again, in a separate directory.

3. In the two different local copies of your repository, make different changes (For example, change the line “Hello world!” to “Hello world from A!” in one, and “Hello world from B!” in the other.) to `gitconflict.txt` and commit them separately.
4. Then, attempt to push the changes of both repositories to GitHub. The first push will be fine, but the second one will fail and tell you the remote repo has commits that your

Skip to main content

local repo does not. It will suggest you pull, so do that. When you try to pull, you may see a message like:

```
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint:   git config pull.rebase false  # merge
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
```

This is because git has 3 choices of what to do when pulling from a divergent branch:

1. `rebase`, which *replays* your local history on top of the remote branch
2. `no-rebase`, which *merges* the divergent histories with a *merge commit*
3. `ff-only`, which does neither, and just returns an error

and you can specify each of these behaviors as the *default* via `git config --global`, or for each call to `git pull` via the `--flags`.

For today, let's make a merge commit with:

```
git pull --no-rebase
```

This will cause a merge conflict which you will need to resolve manually. Your error will look something like this:

```
Auto-merging gitconflict.txt
CONFLICT (content): Merge conflict in gitconflict.txt
Automatic merge failed; fix conflicts and then commit the result.
```

If you type `git status`, you will see that git started merging stuff automatically, but didn't know what to do with `gitconflict.txt`, so git left it in a half-merged state. Opening the copy of `gitconflict.txt` in the second repository shows you something like this:

```
Here's a line
```

Skip to main content

```
Hello world from B!
=======
Hello world from A!
>>>>>>> daa83f4b022b0b5b61a40fef6bb8eedacfe9fd5a

Here's another line
```

Notice that the line on which conflicting changes was made has been replaced by autogenerated text from git. The text above `=======` is what commit B wants this line to contain, while the text below is what commit A wants this line to contain. The text after `>>>>>>>` is the commit hash of commit A

To finish the merge, modify the whole chunk to be whichever you want the 'final version' of the file to contain. (For example, maybe "Hello world from everyone!".) Then save the file.

You can then `git add` and `git commit` your changes and push them. Having done so, try typing `git log --graph` to see a visualization of what happened. Good job, you just resolved a conflict!

Tip: Text-editors such as Visual Studio Code have support for git, including viewing file changes since the last commit and resolving merge conflict.

Creating A Pull Request

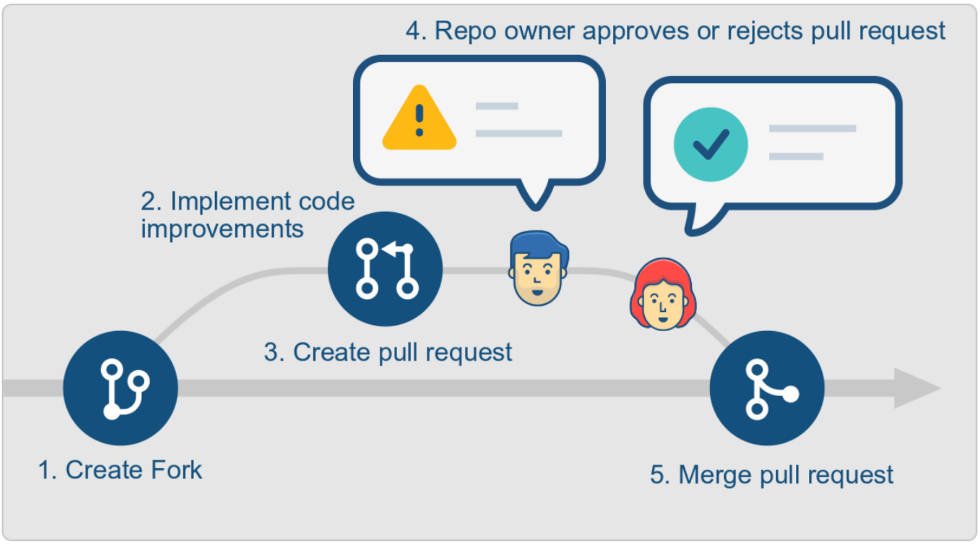
In software development, you will often work with a team on a specific project. To propose changes and collaborate on a project an important tool is the so-called "Pull Request". Pull requests are how most open source work is conducted.

Consider the following example: Sarah owns a project repository and Bill would like to suggest an improvement or new feature to that project. The procedure for Bill's code contribution consists of five steps:

1. Bill creates a *copy* of Sarah's repository. This copy is called a 'fork'.
2. Bill implements his improvements or new feature and pushes these changes to his 'fork'.
3. Bill sends a request to Sarah to include his changes. He does this by creating a 'pull request'.
4. Sarah reviews Bill's changes and can either reject or accept the pull request. If rejected, Bill can commit further improvement to his fork until Sarah is happy.
5. Once the pull request is accepted, Sarah merges the pull request. This means that Bill's

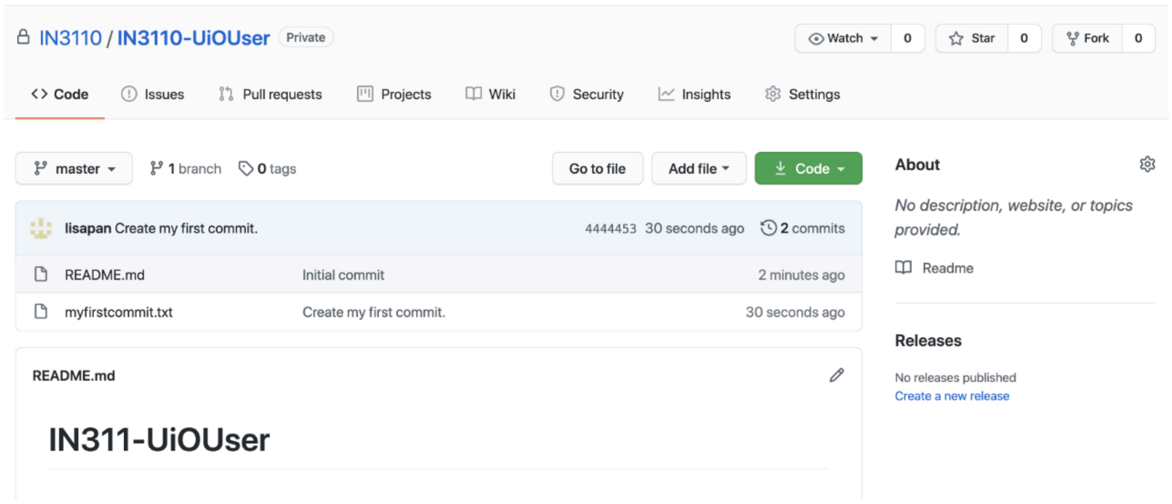
Skip to main content

The figure below visualizes these steps. Note that this procedure even works if Bill does not have write access to Sarah's repository.


4. Repo owner approves or rejects pull request
2. Implement code improvements
3. Create pull request
1. Create Fork
5. Merge pull request

Task 4 Make a pull request following steps 1-5 below **(2 points)**:

1. Go to your IN3110/IN4110 repository. Click on the 'Fork' button on the top right to create your personal copy of the repository.

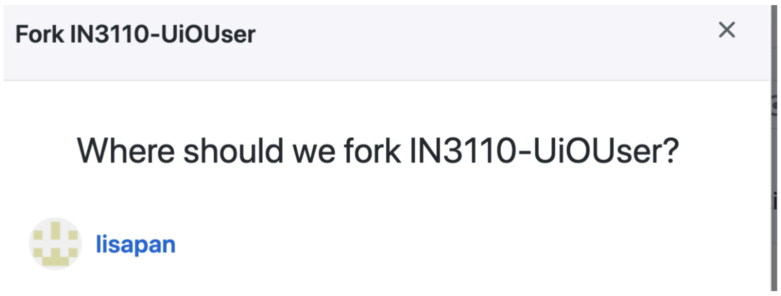

IN3110 / IN3110-UiOUser Private
Watch 0 Star 0 Fork 0
Code Issues Pull requests Projects Wiki Security Insights Settings
master 1 branch 0 tags
Go to file Add file Code
About
No description, website, or topics provided.
Readme
Releases
No releases published
Create a new release

lisapan Create my first commit.	4444453 30 seconds ago	2 commits
README.md	Initial commit	2 minutes ago
myfirstcommit.txt	Create my first commit.	30 seconds ago

README.md

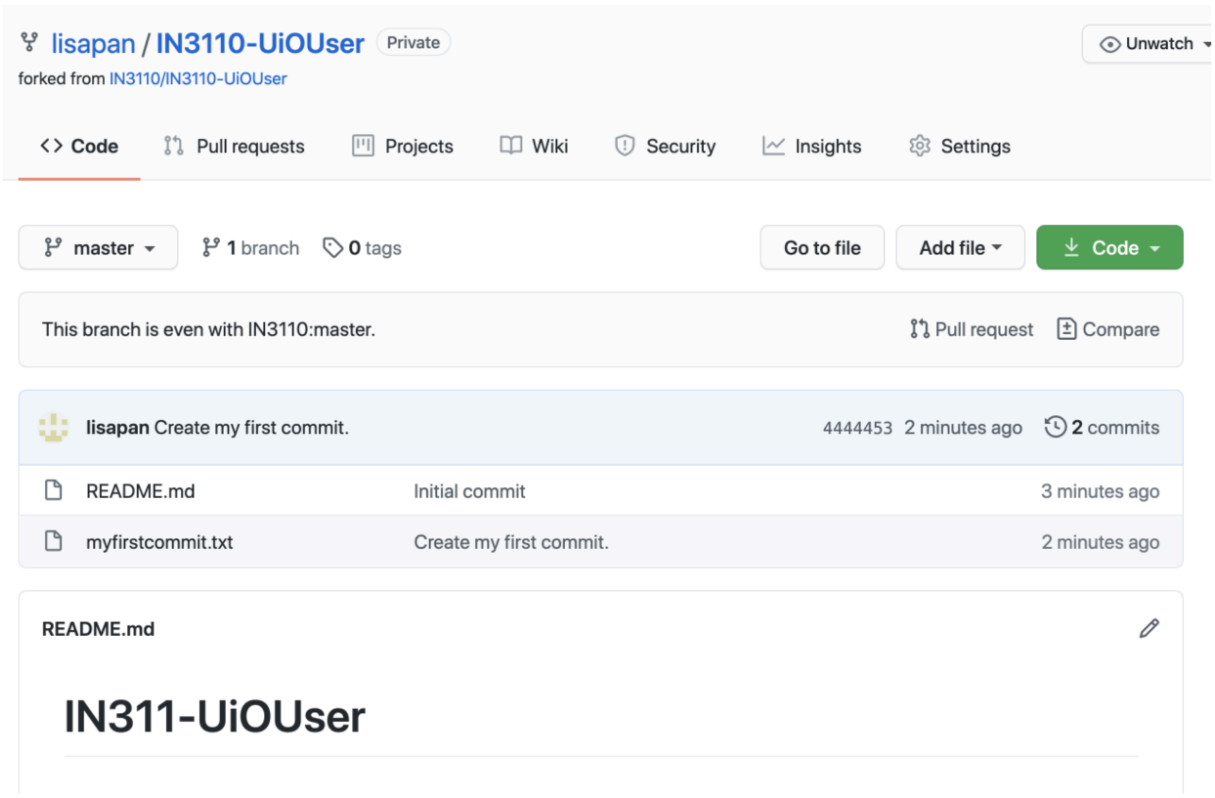
IN311-UiOUser

This could open up a dialogue window asking where to fork your repository to. Make sure you choose your personal UiO User name.



2. Once forked, you will be forwarded to the repository page of your personal copy.

Skip to main content



This branch is even with IN3110:master.

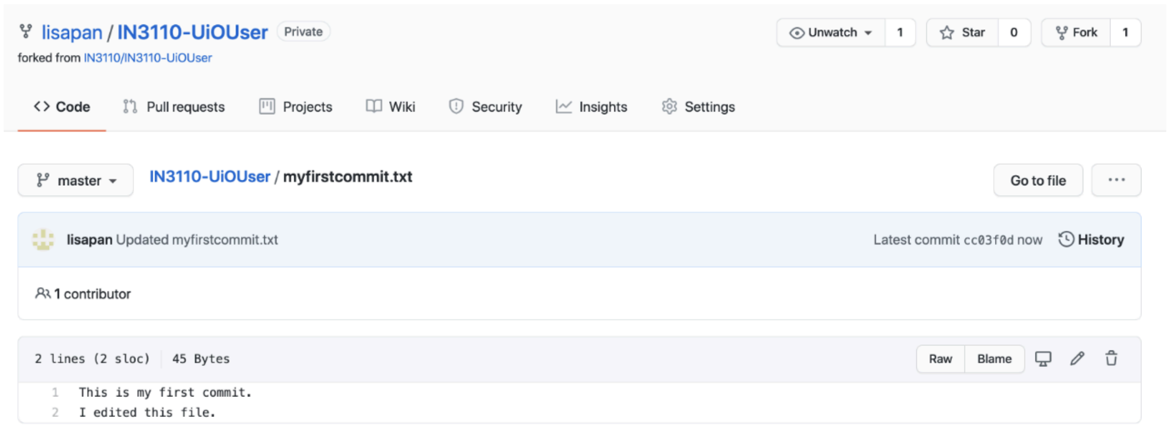
lisapan Create my first commit. 4444453 2 minutes ago

README.md	Initial commit	3 minutes ago
myfirstcommit.txt	Create my first commit.	2 minutes ago

IN311-UiOUser

You can clone this repository as usual to your computer.

3. Modify the text file `myfirstcommit.txt` in your forked repository by adding a line to it saying "I edited this file." Commit and push these changes to your forked repository. **Note:** Don't forget to write a commit message mentioning what you changed. Your modified file in your forked repository will look like this.



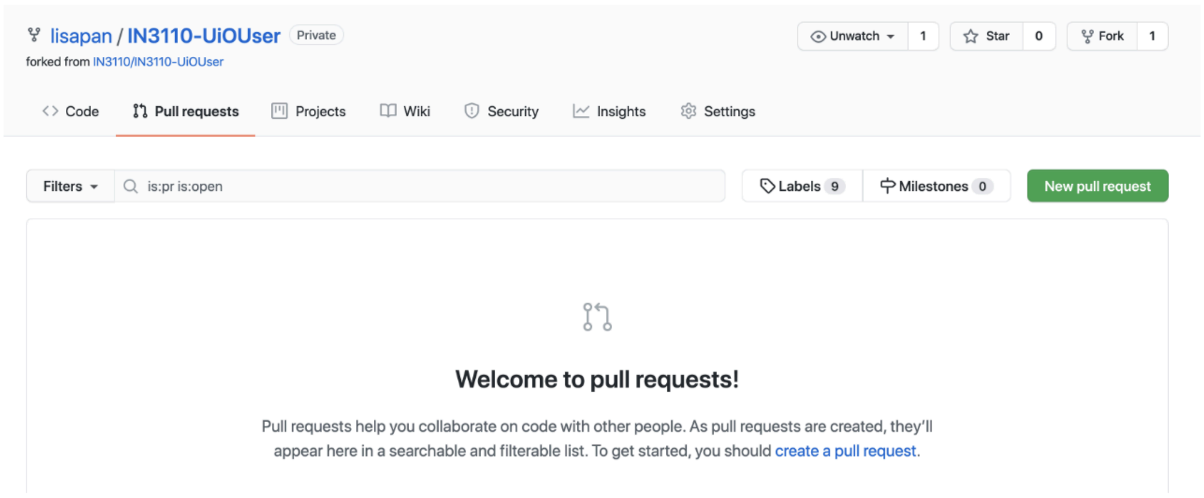
lisapan Updated myfirstcommit.txt Latest commit cc03f0d now

2 lines (2 sloc) 45 Bytes

```
This is my first commit.
I edited this file.
```

4. It is time to create a 'pull request'. Click on the flag "Pull request" in your forked repository and click on the green button "New pull request".

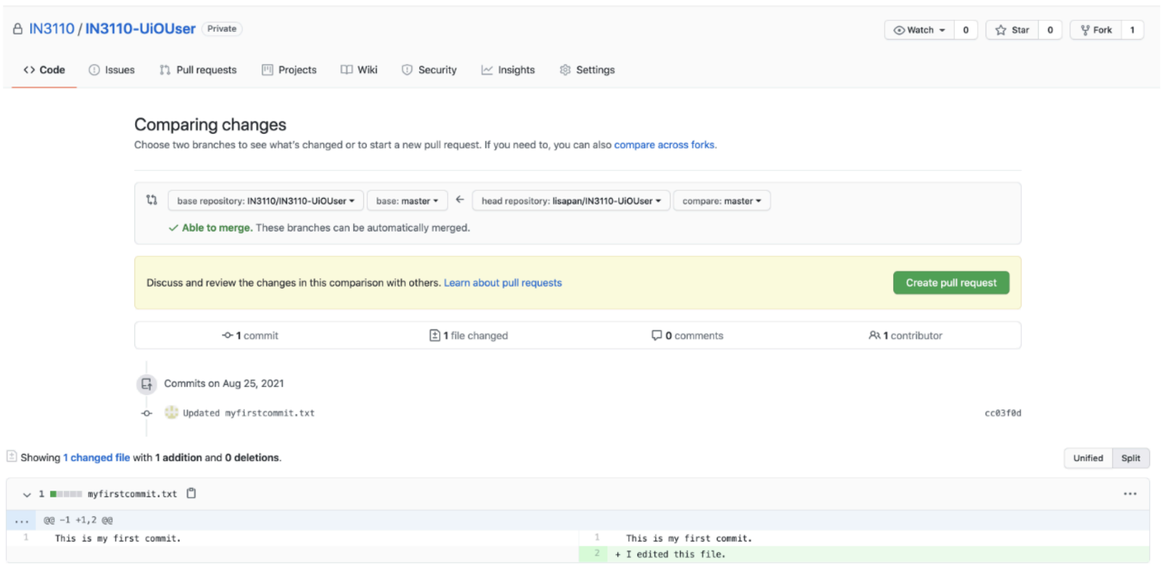
Skip to main content



Welcome to pull requests!

Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should create a pull request.

Make sure that the base repository is the "IN3110/IN3110-UioUser" repository. Merge your modifications from your fork into that repository.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: IN3110/IN3110-UiOUser ▾ base: master ▾ ← head repository: lisapan/IN3110-UiOUser ▾ compare: master ▾

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. Learn about pull requests

1 commit · 1 file changed · 0 comments · 1 contributor

Commits on Aug 25, 2021

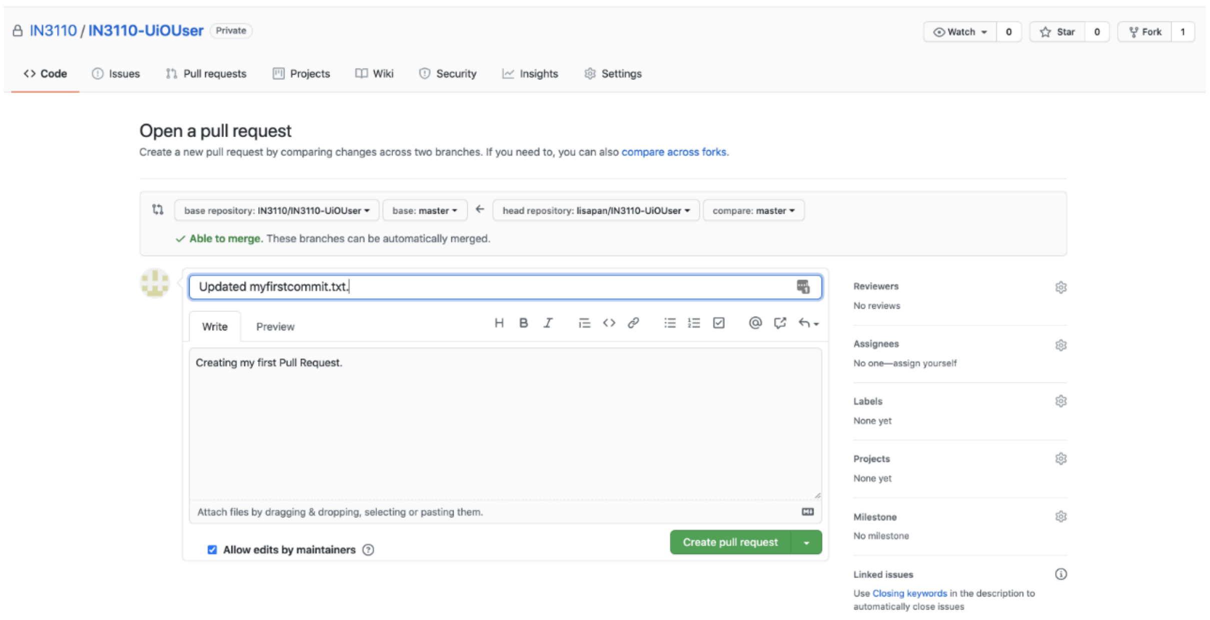
Updated myfirstcommit.txt cc03f0d

Showing 1 changed file with 1 addition and 0 deletions.

```
1 myfirstcommit.txt
@@ -1 +1,2 @@
This is my first commit.          This is my first commit.
                                  + I edited this file.
```

Leave a short descriptive message with the pull request as well as a short comment describing what you did (see image below). Push the button "Create pull request".

Skip to main content



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base repository: IN3110/IN3110-UiOUser base: master ← head repository: lisapan/IN3110-UiOUser compare: master

✓ Able to merge. These branches can be automatically merged.

Updated myfirstcommit.txt.

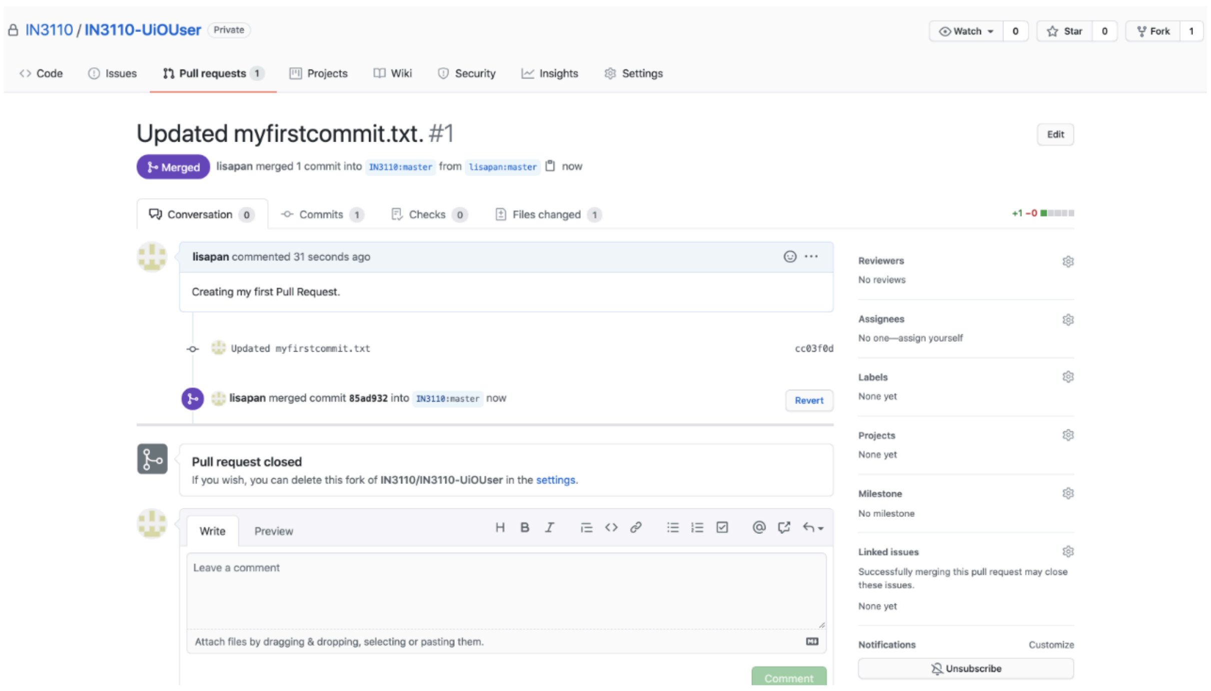
Creating my first Pull Request.

Attach files by dragging & dropping, selecting or pasting them.

[✓] Allow edits by maintainers

Reviewers: No reviews
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Linked issues: Use Closing keywords in the description to automatically close issues

5. You will now see an open "Pull request" at the "IN3110/IN3110-UioUser" repository. To close the pull request, you can merge it. Finish the pull request by confirming the merge.



Updated myfirstcommit.txt. #1

Merged lisapan merged 1 commit into IN3110:master from lisapan:master now

lisapan commented 31 seconds ago

Creating my first Pull Request.

Updated myfirstcommit.txt cc03f0d

lisapan merged commit 85ad932 into IN3110:master now

Pull request closed

If you wish, you can delete this fork of IN3110/IN3110-UiOUser in the settings.

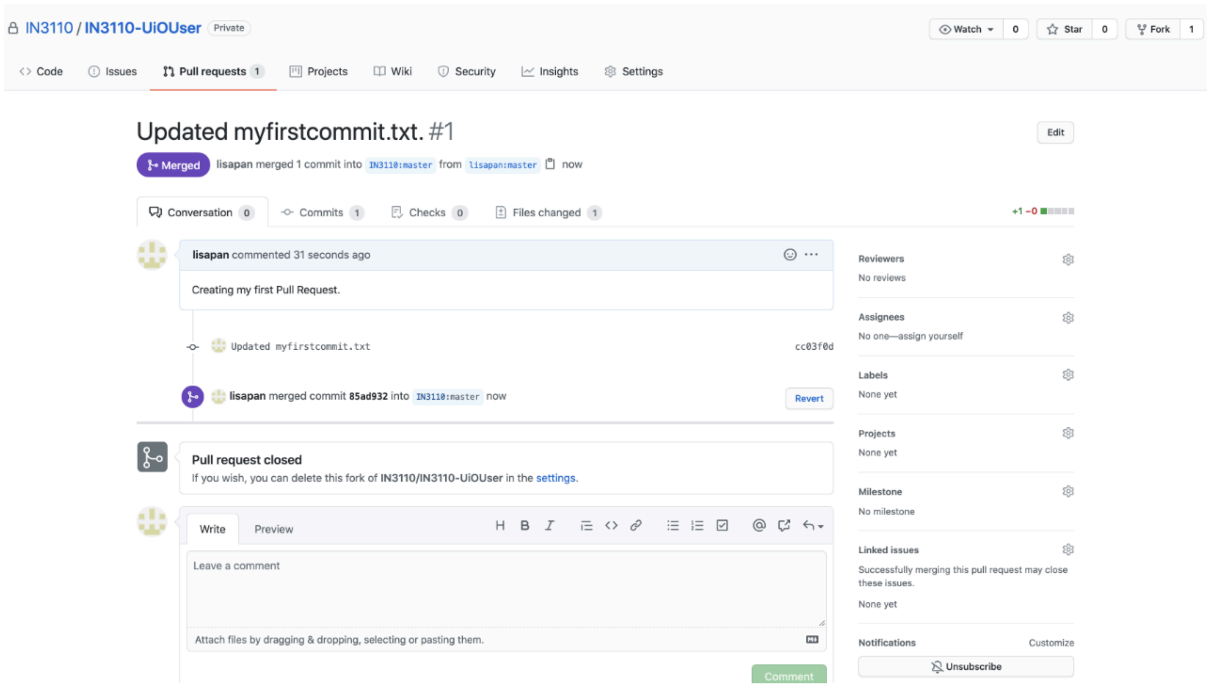
Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Reviewers: No reviews
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Linked issues: Successfully merging this pull request may close these issues. None yet
Notifications

Congrats! You finished your first pull request! You should be able to see a "closed" pull request under the flag "Pull requests" as below.

Skip to main content


IN3110 / IN3110-UiOUser Private

Watch 0 Star 0 Fork 1

Code Issues Pull requests 1 Projects Wiki Security Insights Settings

Updated myfirstcommit.txt. #1

Edit

Merged lisapan merged 1 commit into IN3110:master from lisapan:master now

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 −0

lisapan commented 31 seconds ago

Creating my first Pull Request.

Updated myfirstcommit.txt cc03f0d

lisapan merged commit 85ad932 into IN3110:master now Revert

Pull request closed

If you wish, you can delete this fork of IN3110/IN3110-UiOUser in the settings.

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
None yet

Notifications Customize

Unsubscribe

Checklist before submitting

In order for your code to be gradable, please note:

- ☐ Assignment is in your `assignment1` folder
- ☐ Don’t forget to push! *Committing is not turning it in*, make sure your work can be seen at https://github.uio.no/IN3110/in3110-yourname (**not** your fork at https://github.uio.no/yourname/in3110-yourname)

Previous
Assignments for IN3110/4110

Next
Assignment 2: Automating the boring stuff