Git Examples

Jens Lechtenbörger

Summer Term 2022

Contents

1 Context

In 2021, several students in CACS suggested that I create a short video for Git, demonstrating basic concepts before details show up in the corresponding presentation and exercises.

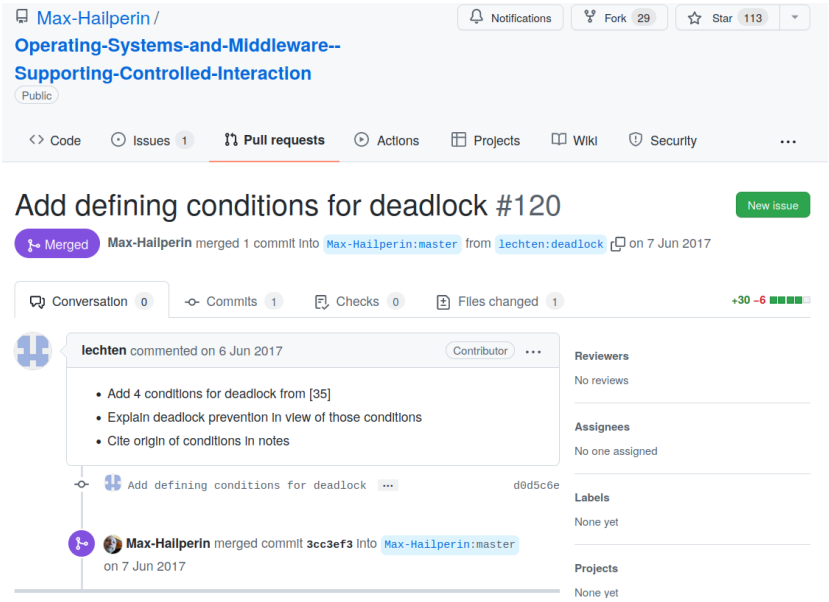
I'm not a fan of videos myself. This document is a script for that video, which you can find on the course page (source code). In my view, the script is superior to the video: It contains working hyperlinks, enables copy&paste, searching, annotations, skim reading, you do not need to decode my pronunciation, and text automatically adjusts to your individual speed. (An advantage of the video is that you can see me smiling. Sometimes. `:-)`)

Briefly, Git is a decentralized version control system, which supports collaboration on a collection of documents, keeping track of changes and versions over time. Let's look at two examples: First, LaTeX source code for a textbook on GitHub, second JavaScript code used in my presentations on GitLab.

2 Collaboration on OER Textbook on GitHub

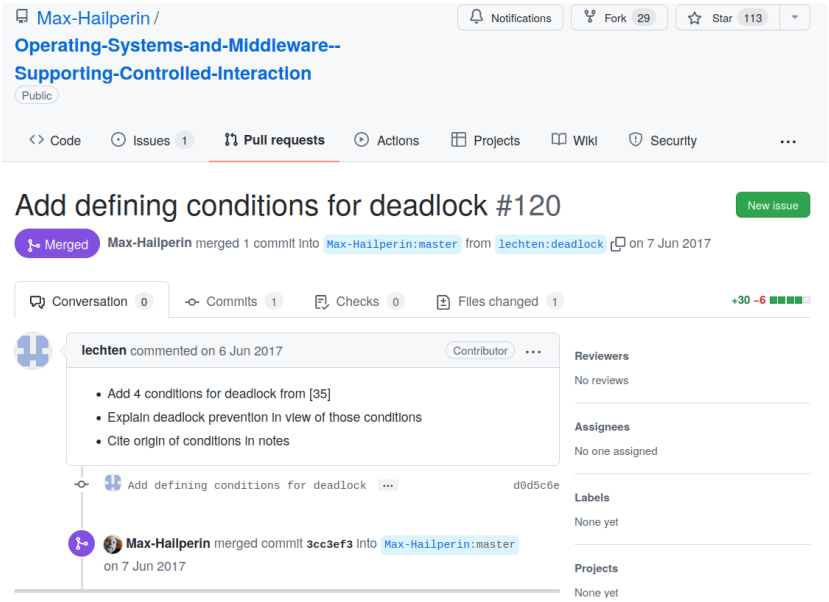
The LaTeX source files for a textbook on operating systems (used at the University of Münster) are maintained in a Git repository on GitHub.

Thus, if you as student or I as instructor find a typo or believe to be able to explain a certain concept in a better way, we can create an improved version and ask the author to incorporate our changes into the book. On GitHub, so-called *pull requests* (PRs) are used for such collaborations, while *merge request* (MR) is the term used with GitLab for the same purpose. See here for a sample PR.



You will learn details of the so-called feature branch workflow for such collaborations. Briefly, *branches* are different versions of the repository, and a *feature branch* is a branch that aims to handle a specific feature or issue. Frequently, the `main` or `master` branch represents a stable version (although teams are free to name their branches as they want; also, any number of branches may be used, say for development, testing, releases). In any case, feature branches contain work-in-progress until they are *merged* into the stable version. For the sample PR you see that the book's author, Max Hailperin, accepted it and merged the branch `deadlock` created by user `lechten`, that's me, into the stable version. `:-)`

Note that Git comes with *diff* functionality to highlight differences between versions, which simplifies review processes.



Collaboration is particularly attractive if freedom granting licenses are being used. In case of educational resources such as this textbook, we then talk about Open Educational Resources, OER for short.

Please feel free to collaborate in a similar fashion on our OER. :-)

Let's turn to the second example.

3 Collaboration on Free Software on GitLab

While Git is sometimes being used for educational resources, it is more popular for software development, again particularly when free and open licenses are being used. If you are using such software and you find a bug or would like to add a new feature, you can create an improved version of that software and suggest the inclusion of your improvements into the original version. This can happen using the feature branch workflow that I talked about already.

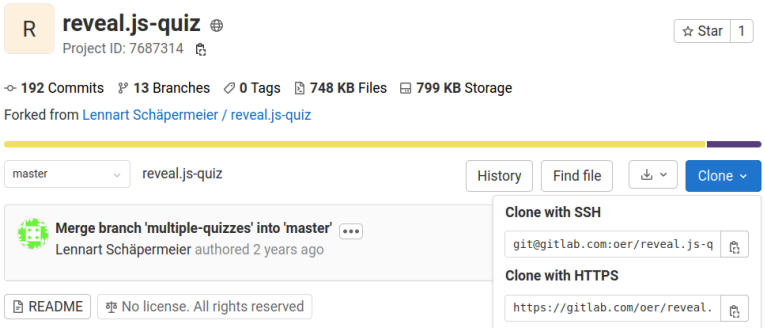
Let's see a step-by-step example for the feature branch workflow with Git and GitLab. I use the JavaScript presentation framework reveal.js for my OER presentations. In addition, I use a quiz plugin for reveal.js, which enables students to check their understanding and supports their learning with automatic feedback. In that plugin, I discovered an incompatibility with a recent change in reveal.js, which I want to fix now. Here, you see the GitLab repository of that plugin, which, by the way, was created by a former student of mine in the context of a Bachelor's specialization module. :-)

1. For demo purposes, I use a new empty directory (and set `LANG` for English messages as well as `PAGER` to prevent pagination when capturing long results—you do not need those `export` commands):

   ```
   mkdir -p /tmp/git-demo
   cd /tmp/git-demo
   export LANG=en_US.UTF-8
   export PAGER=cat
   ```

2. On GitLab I do not have special permissions on the plugin's repository. Thus, I first *fork* the repository. Forking means that I create a copy of the project with which I can do whatever I want. I already forked the project in the past, so now I just go to my fork.

3. I obtain a local copy of the source code, so that I can change files with the editor of my choice afterwards. The Git operation for copying is called *clone*, for which a GitLab GUI button provides repository addresses.



Usually, the HTTPS address is meant for anonymous read access by the general public, while the SSH address requires a user account on the platform. As I want to change my fork, I use the SSH address subsequently. (If you want to try out the below commands without a GitLab account, you need the HTTPS address. However, then you will not be allowed to push changes in step 7 below. In exercises, you will work with an account...)

```
git clone git@gitlab.com:oer/reveal.js-quiz.git
```

Output:

```
Cloning into 'reveal.js-quiz'...
remote: Enumerating objects: 683, done.
remote: Counting objects: 100% (683/683), done.
remote: Compressing objects: 100% (283/283), done.
remote: Total 683 (delta 415), reused 658 (delta 397), pack-reused 0
Receiving objects: 100% (683/683), 350.96 KiB | 1.57 MiB/s, done.
Resolving deltas: 100% (415/415), done.
```

Note that the cloned source code now exists in a new directory, `cd` (change directory) into it, and `ls` (list files):

```
cd reveal.js-quiz
ls
```

Output:

```
quiz  README.md
```

4. I create a new branch for my bugfix with the command `git checkout -b`. As the bug is related to the propagation of events, I call the branch `fix-event-propagation`.

   ```
   git checkout -b fix-event-propagation
   ```

   Output:

   ```
   Switched to a new branch 'fix-event-propagation'
   ```

5. I modify the source code. For demo purposes I did that already. Let's copy a modified file into this directory:

   ```
   cp ~/src/wiwi-gitlab/misc/code/slickQuiz.js quiz/js
   ```

   Let's ask Git about the status of this project:

   ```
   git status
   ```

   Output:

   ```
   On branch fix-event-propagation
   Changes not staged for commit:
     (use "git add <file>..." to update what will be committed)
     (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   quiz/js/slickQuiz.js

   no changes added to commit (use "git add" and/or "git commit -a")
   ```

   Let's see what I changed, i.e., the difference, or *diff* proposed by me (similarly to differences we saw for the PR above):

   ```
   git diff
   ```

   Shortened output (first 12 lines; initial lines encode what was compared; lines with `@@` indicate position of change, line with `+` indicates an added line; `-` would indicate a deleted line, but does not occur here):

   ```
   diff --git a/quiz/js/slickQuiz.js b/quiz/js/slickQuiz.js
   index 1711568..325680f 100644
   --- a/quiz/js/slickQuiz.js
   +++ b/quiz/js/slickQuiz.js
   @@ -698,6 +698,7 @@
                // Bind "start" button
                $quizStarter.on('click', function(e) {
                    e.preventDefault();
   +                e.stopPropagation();

                    if (!this.disabled && !$(this).hasClass('disabled')) {
                        plugin.method.startQuiz.apply (null, [{callback: plugin.config.a
   ```

6. Tell Git what changes to remember. Recall that above we saw suggestions from `git status` how to proceed. In general, it is a very good idea to read each output of Git carefully. And to make sure that one understands it. Web searches leading to StackOverflow usually help. Let's follow the suggestion:

```
git add quiz/js/slickQuiz.js
```

```
git status
```

Output:

```
On branch fix-event-propagation
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

modified:   quiz/js/slickQuiz.js
```

Apparently, the status changed. Git now tells us that changes are recorded to be *committed*. Commit is the technical term for an operation that records changes permanently, along with an explanation for the changes. Commit messages follow certain conventions, with a short first line, up to 50 characters, imperative, capitalized, and without dot; followed by an empty line; followed by a description where lines are wrapped at 72 characters.

```
    git commit -m "Prevent propagation of events

Reveal.js 4.2.0 introduced a new event handler, onSlidesClicked, which
breaks the hyperlinks used by the plugin (they now also trigger a
navigation to the presentation's first slide).
Therefore, stop the propagation of events."
```

Output:

```
[fix-event-propagation fcab02a] Prevent propagation of events
 1 file changed, 6 insertions(+)
```

The status is clean again:

```
git status
```

Output:

```
On branch fix-event-propagation
nothing to commit, working tree clean
```

My new commit is present in the log.

```
git log -n 3
```

Output:

```
commit fcab02a1b0f0f51d121f53c9b02f00f767e9271b (HEAD -> fix-event-propagation)
Date:   Mon Mar 28 15:17:39 2022 +0200

    Prevent propagation of events

    Reveal.js 4.2.0 introduced a new event handler, onSlidesClicked, which
    breaks the hyperlinks used by the plugin (they now also trigger a
    navigation to the presentation's first slide).
    Therefore, stop the propagation of events.

commit 602e918981acdf37aab5a71494b19a6317fd8ae0 (origin/master, origin/HEAD, master)
Merge: 6fdb244 c7e2c86
Date:   Tue Oct 1 17:25:09 2019 +0000

    Merge branch 'multiple-quizzes' into 'master'

    Allow multiple quizzes per presentation

    See merge request schaepermeier/reveal.js-quiz!11

commit c7e2c86ad2f5cb20088229b1c19b1a602bad79f6 (origin/multiple-quizzes)
Date:   Fri May 10 14:20:57 2019 +0200

    Fix initialization

    If no quiz has the name "quiz", the variable ‘quiz‘ is undefined.
```

7. I *push* my new local commit to the remote fork at GitLab:

   ```
   git push -u origin fix-event-propagation
   ```

8. Let's see my fork at GitLab: `https://gitlab.com/oer/reveal.js-quiz`
   Note that GitLab identified the new branch and suggests that I create an MR.

9. Create MR.

That's it for now. When we create MRs, maintainers receive e-mail notifications. Then, it is up to them to review the changes. Hopefully, maybe after some discussion, they merge the changes into their stable version. If that should not happen in this specific case, I can still use my fork, which contains my changes.

Which free/libre and open source project do you want to improve next? `:-)`

License Information