Git tutorial notes

Michael Papenbrock

September 2019

1 One-time setup

These steps have to be done only once after installing Git on a new computer.

- `git config --global user.name "FirstName LastName"`
- `git config --global user.email adress@example.uu.se`

2 Get a copy of repository

Before the regular workflow starts, the repository has to be downloaded from the server.

- `git clone url_to_repository`

  As URL choose the following:

- Code discussions repository: `https://github.com/mpapenbrock/CodeDiscussions`

3 Workflow

At any point, when you want to know the state of your local repository, use the following:

- `git status`

It is advised to use this as often as possible.

The regular workflow for making changes to the repository looks like this:

1. `git pull`
2. Perform work by creating or modifying files in your local repository as needed
3. `git add file1 file2 ...`
4. `git commit -m "descriptive message about changes"`

5. `git push`

In order, these do:

1. Checks for and downloads the latest version of the repository from the server and updates the local repository to that version. Do this at the beginning of every work session.
2. That is up to you.
3. Decides which files go into the next version that is to be committed. Note: This is sometimes referred to as "staging changes", for example in some GUIs.
4. Creates a new version with the files that have been added in the previous step. Also attaches a short (!) message to that version that should reflect what has been changed.
5. Sends the new versions back to the server.

Note that you might want to perform steps 2 to 4 several times in a longer work session, e.g. when working on multiple exercises, but steps 1 and 5 only at the beginning and the end of that session.

4 Resolving conflicts

When multiple users work on a project at the same time, conflicts can arise. Git is able to resolve many of these conflicts on its own, but sometimes action from the user is required. For example, this can happen when two users edit the same file simultaneously and try to upload conflicting versions of it. This can be avoided to a degree by running `git pull` and `git push` frequently.

Usually, Git will tell you what to do. However, it is always a good idea to run `git status`. If a conflict is present, it will likely inform you that there are "unmerged paths" and point out the problematic file. At this point, the file will contain both the conflicting changes and it will be necessary to manually modify it. Once you are happy with it, perform the steps 3 to 5 from earlier to upload the resolved state as the latest version.

Note that there is not always a silver bullet to resolving conflicts when working with Git. However, by pulling and pushing often as well as working in a separate directory, most conflicts can be avoided.