Git Advanced

Learning Objectives

- [ ] deepen understanding of git in general
  - [ ] `git fetch`
  - [ ] `git pull`
- [ ] understand why and how git conflicts happen
- [ ] know how to combine branches:
  - [ ] `git merge`
  - [ ] `git rebase` / `git pull --rebase`
- [ ] know how to resolve git conflicts

Understanding Advanced Git Commands

`git fetch`

To check whether your local branches are up to date with the remote repository (by default: `origin`), use `git fetch`.

If there are any differences,

- `git fetch` updates the remote-tracking branches
- this means that your local branch knows about all recent remote changes, but **does not contain these changes yet**
- you still need to merge these changes manually (using `git pull` or `git merge`).


git fetch terminal output

> 💡 To see exactly which branches are up to date with origin/main, use `git fetch -v`.

`git merge`

To join two branches together, you can use `git merge`.

Make sure to

- run `git fetch` first if you want to merge a remote branch
- switch to the branch where you want to incorporate the changes
- run `git merge <branchname>`


git merge terminal output positive

> 💡 Note that `git merge` is a fast-forward merge by default, which means that it will not create a merge commit if the history is clean.

> 💡 If a fast-forward merge isn't possible, `git merge` will create a merge commit, that incorporates the change. To avoid this and keep a clean history, you can use git rebase.

`git pull`

You can use `git pull` to fetch changes from a remote repository **and** merge them into the current branch.

This is possible due to the fact that `git pull` runs `git fetch` and then `git merge`.


git pull terminal output positive

`git rebase`

To join two branches together, you can also use `git rebase`. In contrast to `git merge`, it doesn't create a merge commit, which results in a cleaner git history.

This strategy first applies all the changes of the branch we are rebasing to (the branch used after the `rebase` command). Then it applies the changes of the branch we are rebasing from (the current branch) on top of it commit by commit.

This is like "replaying" your latest changes on top of the changes from the other branch.

To rebase,

- switch to the branch where you want to incorporate the changes
- use `git rebase main` to incorporate the changes from the `main` branch into the branch you switched to
- if a conflict occurs,
  - see below how to solve it in VSCode and
  - how to finish rebasing

How to Solve Conflicts

When your create a PR and want to merge the feature branch into the main branch, it happens that somebody made changes to the same lines of code you were working on. Git cannot decide which code is the right one and marks it as a conflict.

Conflicts have to be solved manually by developers.

How to solve conflicts in VSCode

If a merge conflict occurs, you can use VSCode to solve it:

- go to the `Source Control` Tab of VSCode, and open the file marked with ❗ :


VSCode choosing changes view

- choose one option to resolve the merge conflict:
  - `Accept Current Change`: keep content of the feature branch
  - `Accept Incoming Change`: keep content of the main branch (because we are merging main into the feature branch)
  - `Accept Both Changes`: keep both changes
- save the file and stage the changes (`git add <filename>`)
- follow the further steps along git merge or git rebase

How Common Conflict Emerge

Scenario 1: Feature Branch differs from Main Branch


Conflict Message in GitHub pull request Page

There are two ways to handle the conflict:

- solve the conflict locally (see explanation below)
- solve the conflict directly on GitHub (see explanation below)

Scenario 2: Fatal Error on `git pull`

Using `git pull` can result in the error message `fatal: Not possible to fast-forward, aborting`. This happens when:

- You have done at least one commit locally.
- In the meantime, somebody else has committed to the branch you want to pull.
- The problem is that git does not know which commit(s) came first.


git pull aborting

To solve the conflict, follow these steps:

- `git pull --no-ff`
- resolve merge conflicts (if any) (see explanation below)
- `git commit -m "resolve merge conflict"`
- `git push`


git pull --no-ff output

Solving Conflicts

Solving the Conflict Locally

Option 1: `git merge`

If the feature branch has conflicts with the main branch, you can use `git merge` to solve the conflict:

- go to feature branch
- use `git merge main` to merge the main branch into the feature branch
- the terminal will output something like this:

  
  git merge main conflict output in terminal

- resolve the conflict in VSCode (as mentioned above)
- push the feature branch to GitHub

- conflict solved: you can merge the PR on GitHub 🎉

Option 2: `git rebase`

Instead of `git merge`, you can use `git rebase` to solve a conflict:

- go to feature branch
- use `git rebase main`
- resolve merge conflicts step-by-step (as mentioned above)
- use `git rebase --continue` to continue (will open an editor for the merge commit message)

  > 💡 Most likely you're in vim. To save and close the editor type `:wq` and press `Enter`.

- `git push --force-with-lease` (protects the remote branch if different from local branch)

Solving the Conflict Remote on GitHub

Instead of solving a conflict locally, you can use the GitHub conflict manager.

Resources

- Git Handbook
- Visualizing Git
- Git Emergency Help: ohshitgit.com