2023-03-01

Advanced Git (Part 1)

A practical guide for every-day git commands.

http://www.marceleichner.de / https://www.qr-code-generator.com/

Configuration

Git offers a wide range of configuration options. This chapter will be about the configuration options relevant for this talk.

$EDITOR / core.editor

Everytime git requires some input from the user the text editor is opened. Best example: `git commit -a` opens the editor to enter the commit message.

```
export EDITOR=nano >> ~/.bashrc
```

Alternatively only for git commands:

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

Cleaning Up

Typos in last commit message ✒️

Correct typos in the last commit message using `--amend`

```
git commit --amend
```

Will open the editor with the previous commit message to adjust. Closing the editor will change the commit message.

```
git commit --amend -m "adjusted"
```

Dump your work 🗑️

reset all changes to files to the latest commit to `HEAD`.
DANGEROUS as it cannot be reverted.

```
git reset --hard
```

Would not revert untracked changes like new files. Remove new files too:

```
git reset --hard -f
```

Put your work away 📋

Use stash to switch context quickly without loosing changes:

```
git stash
```

You can have multiple stashes, re-apply, diff and clean them up.

Read more about stash

Partial Commits

Image you changed multiple sections of files while experimenting to find the right solution. Now you have different context of changes and would like to annotate each change by commiting them seperately.

```
git add --patch
```

Shows you each diff and asks wheter to `(1/1) Stage this hunk [y,n,q,a,d,e,?]?`

The most common cases you would use `y` for yes and `n` for no. Sometimes the hunk is to big you would use `s` (if available) to split it further.

Rewriting History

Caution! ⚠️

Rewriting history will change the structure of the git tree, change git hashes. Be careful when your changes are already pushed to a remote used by other developers.

Squash Branch

Reduce the changes of a branch to a single commit using --squash

```
git co -b feat/my-new-branch-name
git merge --squash my-source-branch
git commit -a "new commit message"
```

After this you would still have the `my-source-branch` which can be deleted.

Combine Commits (Squash)

Use rebase to combine multiple commits:

```
git rebase --interactive HEAD~3
```

This opens up the `$EDITOR`:

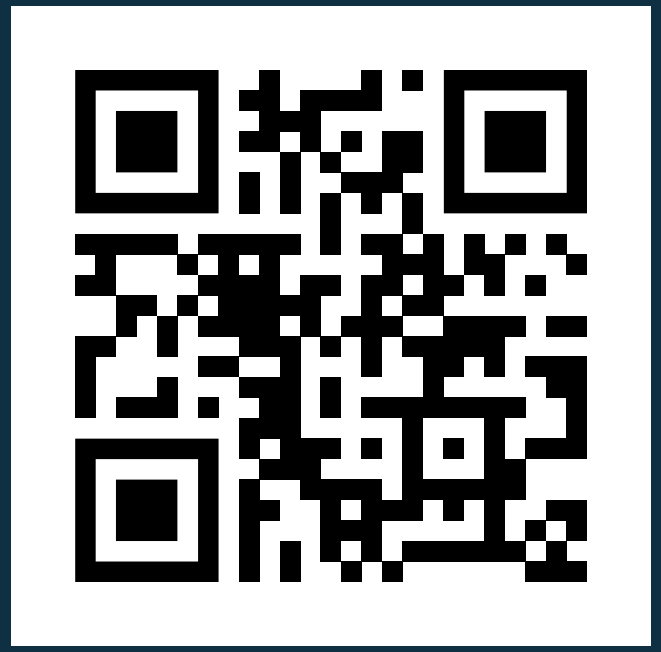
```
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
```

Outlook

- How to solve common problems / errors
- How to clean git history from secrets
- What are git hooks?
- useful git aliases
- git tuis

Questions?

- Which error message stresses you out?
- What did you always wanted to know?
- Send them to me via Slack



Thanks for listening!

✌️

Please give feedback & ask questions!