

SageMath Workshop

Hands-On: Git and GitHub

Git and working together with others

July 4, 2025 – National Mathematical Centre, Abuja

Benjamin Hackl

Department of Mathematics and Scientific Computing @ University of Graz

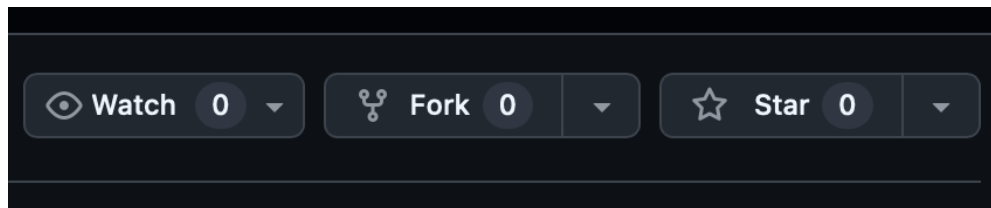


Plan for today

- ▶ Working with GitHub:
 - ▶ Navigating GitHub
 - ▶ *Forking and Cloning*
 - ▶ Working with Issues
 - ▶ Proposing Changes directly on the site
- ▶ Editor with proper Git integration: VS Code
 - ▶ Git workflow with VS Code


Repositories on GitHub

- ▶ <https://github.com/NMCAbuja-Workshop-25/swdev-git-demo>



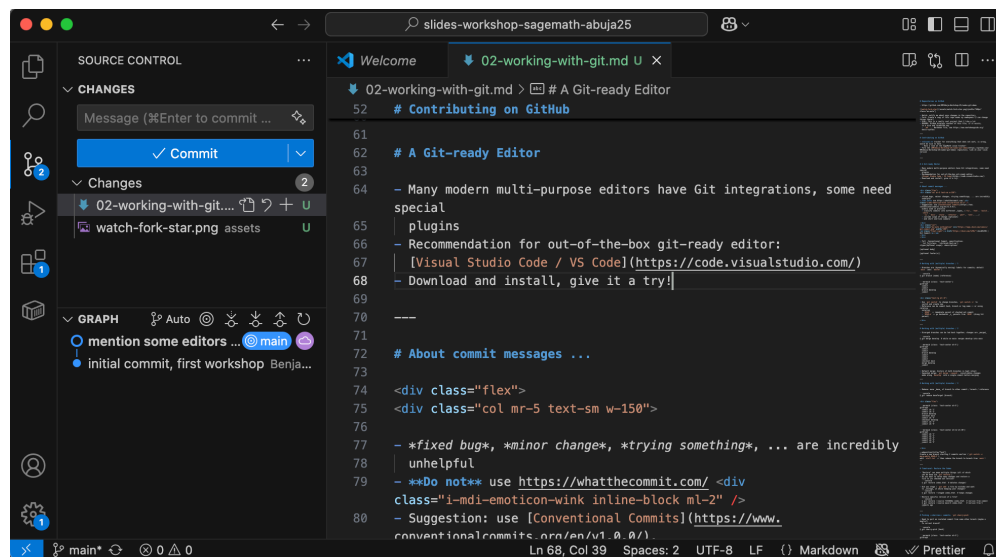
- ▶ Watch: notify me about any* changes in the repository
- ▶ Fork: Create a copy of this repo under my namespace (I can change things there!)
- ▶ Star: This is a really cool project that I like a lot
- ▶ README: GitHub displays content of this file, if it exists, in a nice and formatted way
 - ▶ `.md` ... Markdown file, see <https://www.markdownguide.org/basic-syntax/>

Contributing on GitHub

- ▶ **Issues:** tracker for everything that does not work, is wrong, would be nice to have ...
 - ▶ Have a look at the SageMath issue tracker!
- ▶ Fork the NMCAbuja-Workshop-25/swdev-git-demo repository, look at your local version
- ▶ You can make changes directly from the browser, useful for small changes and fixes!
 - ▶ "Full" editor: hit the  key on your keyboard!
- ▶ **Pull requests:** proposed changes for the owner(s) to review
 - ▶ Check a pull request in the SageMath repo!
 - ▶ Make a change (add a file, or fix a typo) in our demo repository, then create a pull request!

A Git-ready Editor

- ▶ Many modern multi-purpose editors have Git integrations, some need special plugins
- ▶ Recommendation for out-of-the-box git-ready editor: Visual Studio Code / VS Code
- ▶ Download and install, give it a try!



About commit messages ...

- ▶ *fixed bug, minor change, trying something, ...* are incredibly unhelpful
- ▶ Do not use <https://whatthecommit.com/> ☹
- ▶ Suggestion: use Conventional Commits, widely used in practice
 - ▶ classify commits into different *types* (`fix` , `feat` , `build` , `chore` , `ci` , `docs` , `style` , `refactor` , `perf` , `test` , ...)
 - ▶ include scope of change (optional)
 - ▶ and short one-line summary

- ▶ Full *Conventional Commit* specification:

```

1 <type>[optional scope]: <description>
2
3 [optional body]
4
5 [optional footer(s)]

```

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

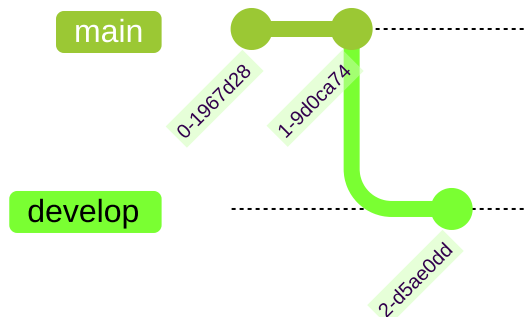
AS A PROJECT DRAGS ON, MY GIT COMMIT
MESSAGES GET LESS AND LESS INFORMATIVE.

xkcd#1296 | Git Commit

Working with (multiple) branches / 1

- ▶ Branches are (dynamically moving) labels for commits; default `main` (was: `master`)

```
1 $ git branch [name] [reference]
```

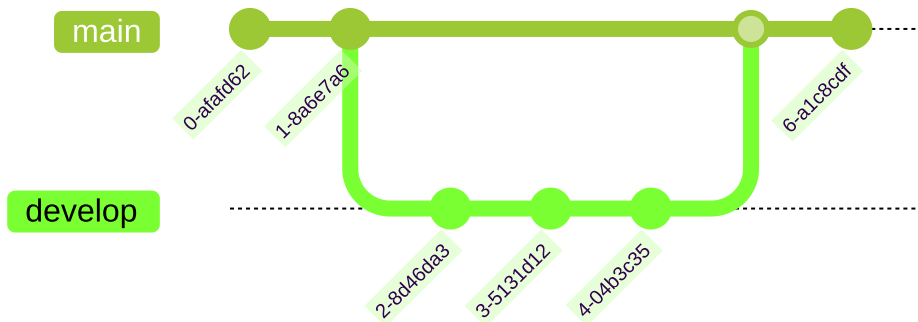


- ▶ Use `git switch` to change branches, `git switch -c` to switch and create new
- ▶ Reference can be commit hash, branch or tag name – or using relative notation
 - ▶ `HEAD^` – immediate parent of checked out commit
 - ▶ `HEAD~n` – go backwards n parents from `HEAD` (along 1st parent)

Working with (multiple) branches / 2

- ▶ Diverged branches can be led back together; changes are *merged*

```
1 $ git merge develop # while on main: merges develop into main
```



- ▶ Default merge: history of both branches is kept intact
- ▶ Squashed merge: `git merge --squash`, consolidates changes made along **develop** into a single commit before merging

Working with (multiple) branches / 3

- ▶ Rebase: move *base* of branch to other commit / branch / reference

```
1 $ git rebase baseTarget [branch]
```



Task

Create a new branch starting 2 commits earlier (`git switch -c more-edits HEAD~2`), edit `stuff.txt` – then rebase the branch to branch from `main` !

Timetravel: Restore the Index

- ▶ "Restore" can mean multiple things (all of which can be done via `git restore`)
- ▶ Do you want to throw away changes and restore a file to its checked out version?

```
$ git restore index.html # deletes changes!
```

- ▶ Did you stage / `git add` a file by mistake and want to *unstage* it while keeping your changes?

```
$ git restore --staged index.html # keeps changes
```

- ▶ Restore specific version of a file?

```
$ git restore --source 723468be index.html # version from commit  
$ git restore --source main~5 index.html # version from 5 commits ago
```

Resolving merge conflicts

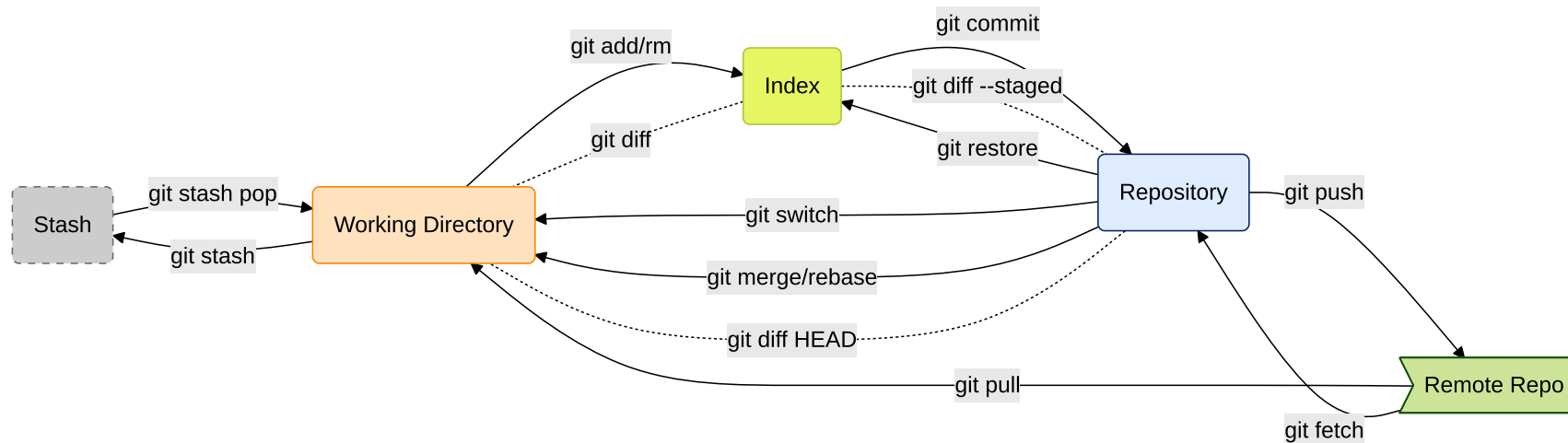
- ▶ When merging back diverged branches, conflicts can happen
- ▶ Git needs help to figure out how the *merged* version looks like (happens e.g. when both branches have edited same line differently)
- ▶ Git will insert markers in conflicted files, edit (merge!) these; then `git add / commit`

```
<<<<<< HEAD
...  # local changes
=====
...  # incoming changes
>>>>>> origin/other-branch
```

Task

Merge the `new-section` branch into `main` in our test repo, resolve the conflict manually!

Overview: Basic Workflow



Who wrote that code?!

- ▶ Detailed author information is available in the repository metadata
- ▶ `git blame` allows checking this information efficiently

```

1  $ git blame manim/__main__.py
2  daf23c9d1 (GameDungeon      2022-01-19 22:26:21 -0600
3  daf23c9d1 (GameDungeon      2022-01-19 22:26:21 -0600
4  018e4a3c1 (J              2023-12-10 23:47:11 +0100
5  32b714a89 (Jason Villanueva 2022-04-24 03:05:23 -0700
6  dcb90a865 (Naveen M K      2021-04-03 13:45:12 +0530
7  32b714a89 (Jason Villanueva 2022-04-24 03:05:23 -0700
8  4c6b6b6ac (e4coder         2021-05-05 14:07:48 +0500
9  efd6474b7 (Benjamin Hackl   2023-08-05 11:03:09 +0200
10 32b714a89 (Jason Villanueva 2022-04-24 03:05:23 -0700
11 4c6b6b6ac (e4coder         2021-05-05 14:07:48 +0500 1
12 a87bb2848 (Jason Villanueva 2021-04-01 23:53:07 -0700 1

```



More useful commands, tools, resources

More useful Git commands

- ▶ `git reflog` – keeps track of reference updates; "Git journal"
- ▶ `git worktree` – manage multiple worktrees of same repository
- ▶ `git bisect` – binary search for "bad" commit
- ▶ `git submodule` – include locked version of repo of dependency in your project
- ▶ `git revert` – undo one commit
- ▶ `git reset --hard [reference]` – reset branch label to reference (this can drop commits!)

Did you break your repo?

- ▶ <https://ohshitgit.com/> – Fix suggestions for common problems with Git repositories
- ▶ <https://rtyley.github.io/bfg-repo-cleaner/> – More robust version of `git filter-branch`
+ etc., for systematic branch/repo-rewrites (e.g., remove secret from all branches without massive interactive rebase ...)

References

- ▶ MIT Missing Semester: Version Control
- ▶ Interactive Git Tutorial, in your browser
- ▶ Pro Git book (free)
- ▶ Git from the bottom up: a deep dive, including implementation background
- ▶ So You Think You Know Git - FOSDEM 2024
- ▶ Some diagrams in these slides are based on the course material of *Software Management* @ University of Innsbruck; © Simon Haller-Seeber