Bremen

CG
VR

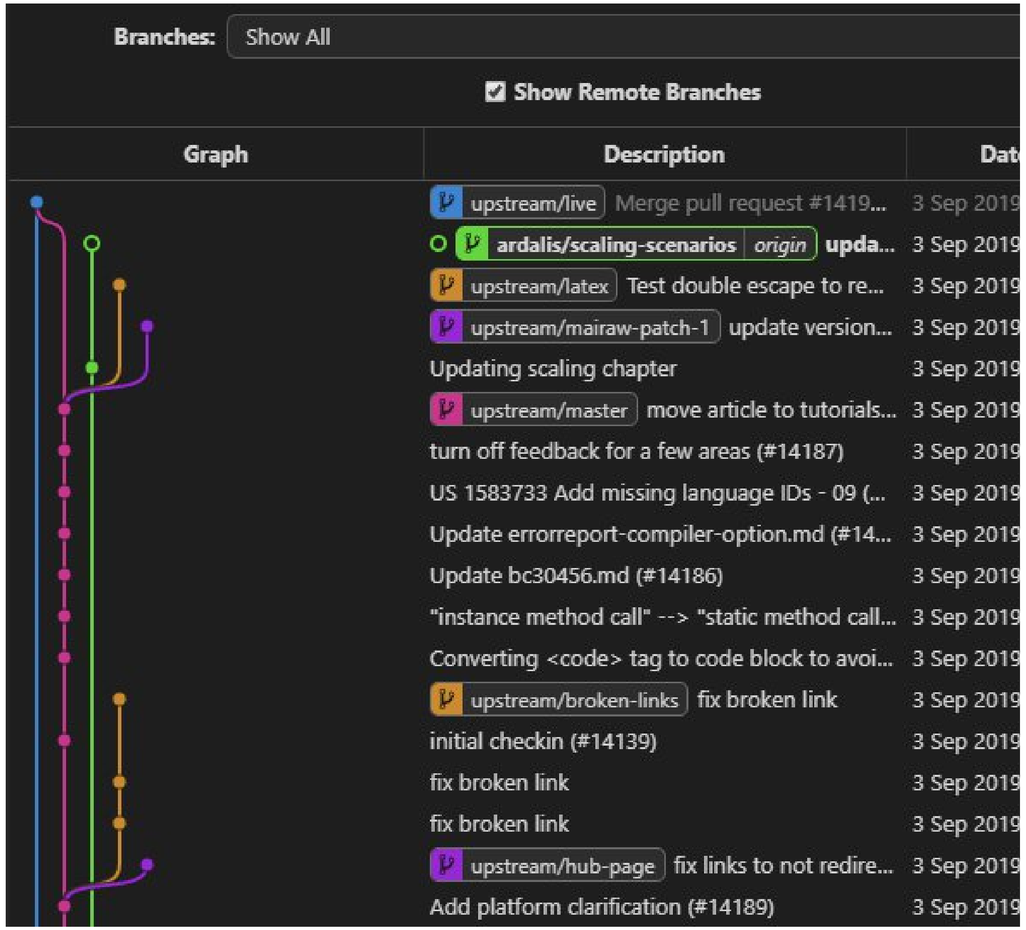
CGVR Lab

Introduction to

git

Introduction to git

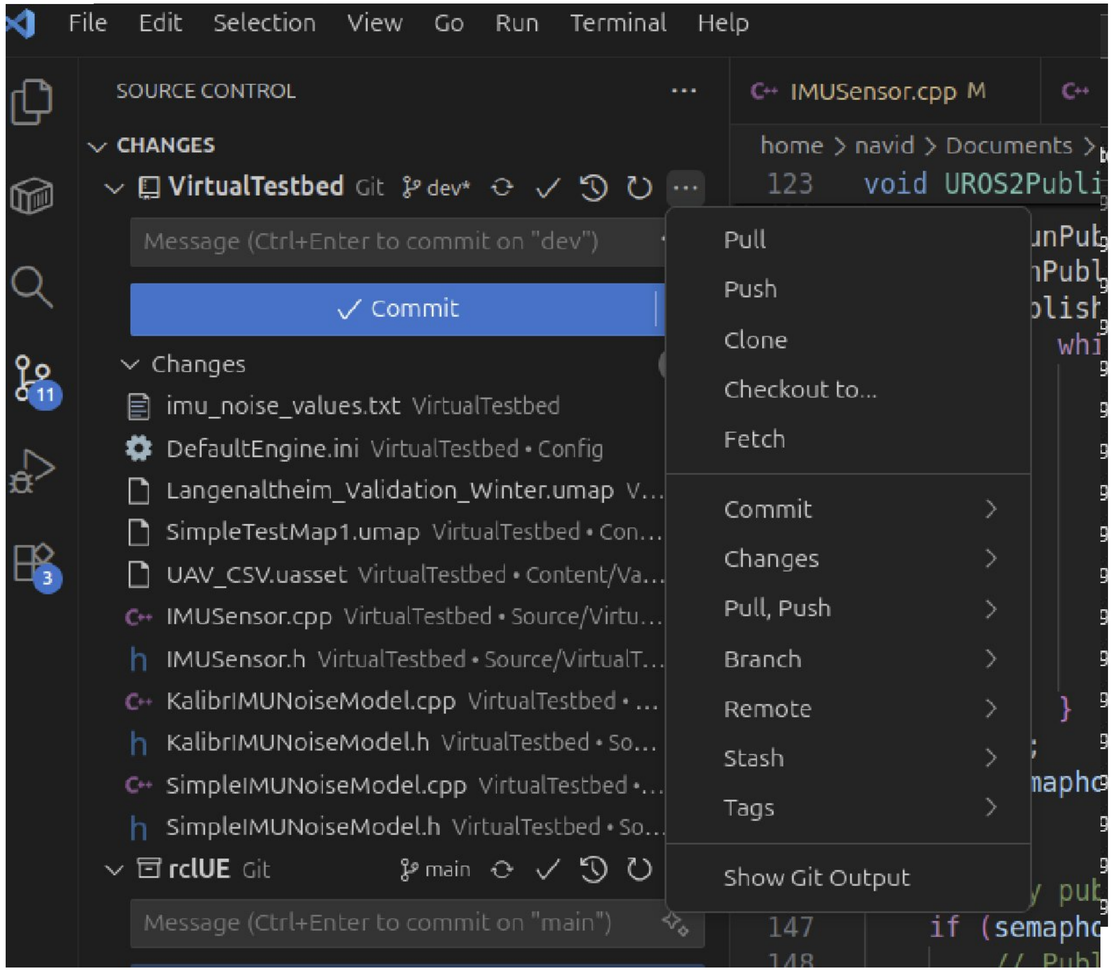
- What is git?
  - versions control software (VCS) for cooperative work on software / code
  - Code changes can be stores locally or remotely on a server / the cloud.
  - Changes can then be synchronized and integrated.



Source: ardalis.com

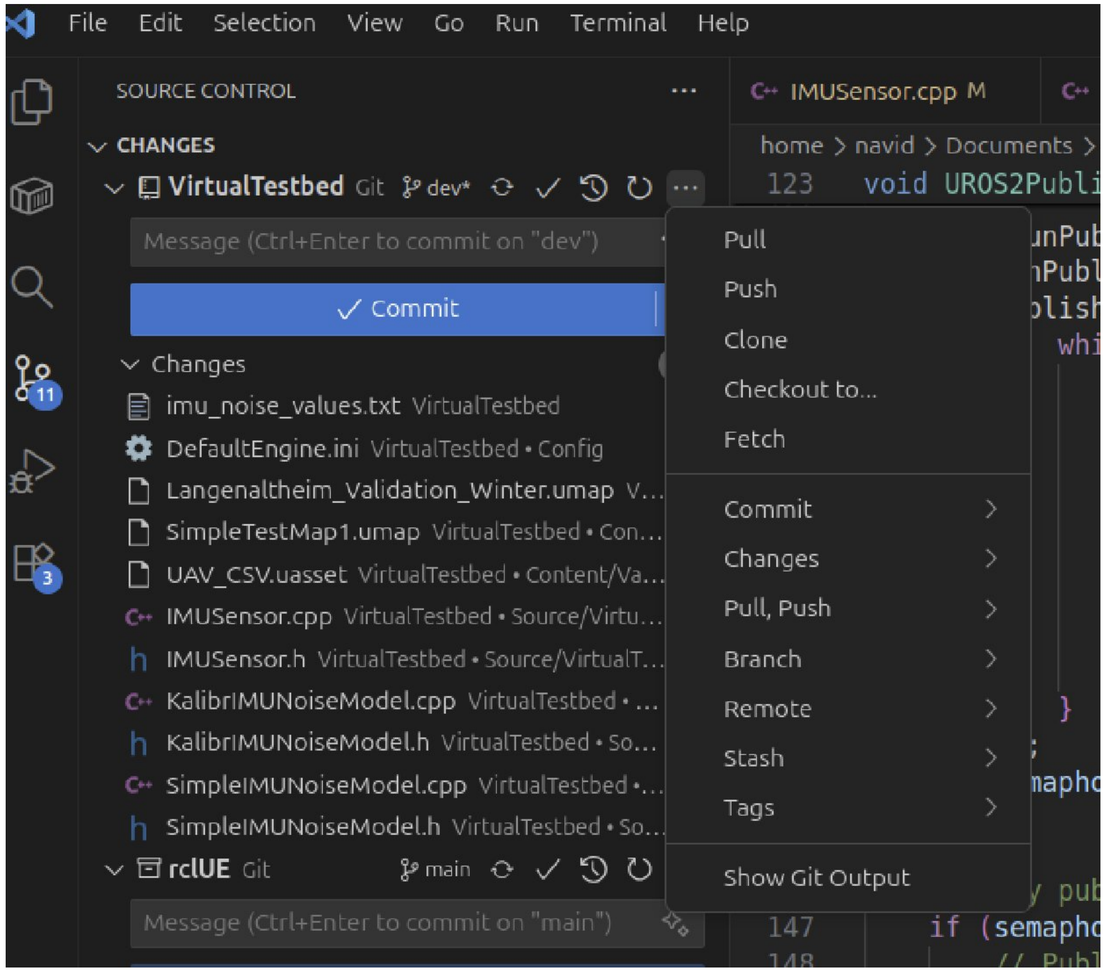
Introduction to git

- How to use git?
  - Via the terminal or GUI-applications (e.g. Github-Desktop).
  - Nowadays all IDEs / code editors have git-integration, therefore also VS Code.
- Installation
  - Usually preinstalled on Linux & Mac.
  - For Windows: https://git-scm.com/downloads/win



Introduction to git

- relevant terminology
  - repository
  - clone
  - commit & commit history
  - branch & branch tree
  - staged changes
  - pull / push
  - stash
  - merge



Interactive Demonstration with VS Code

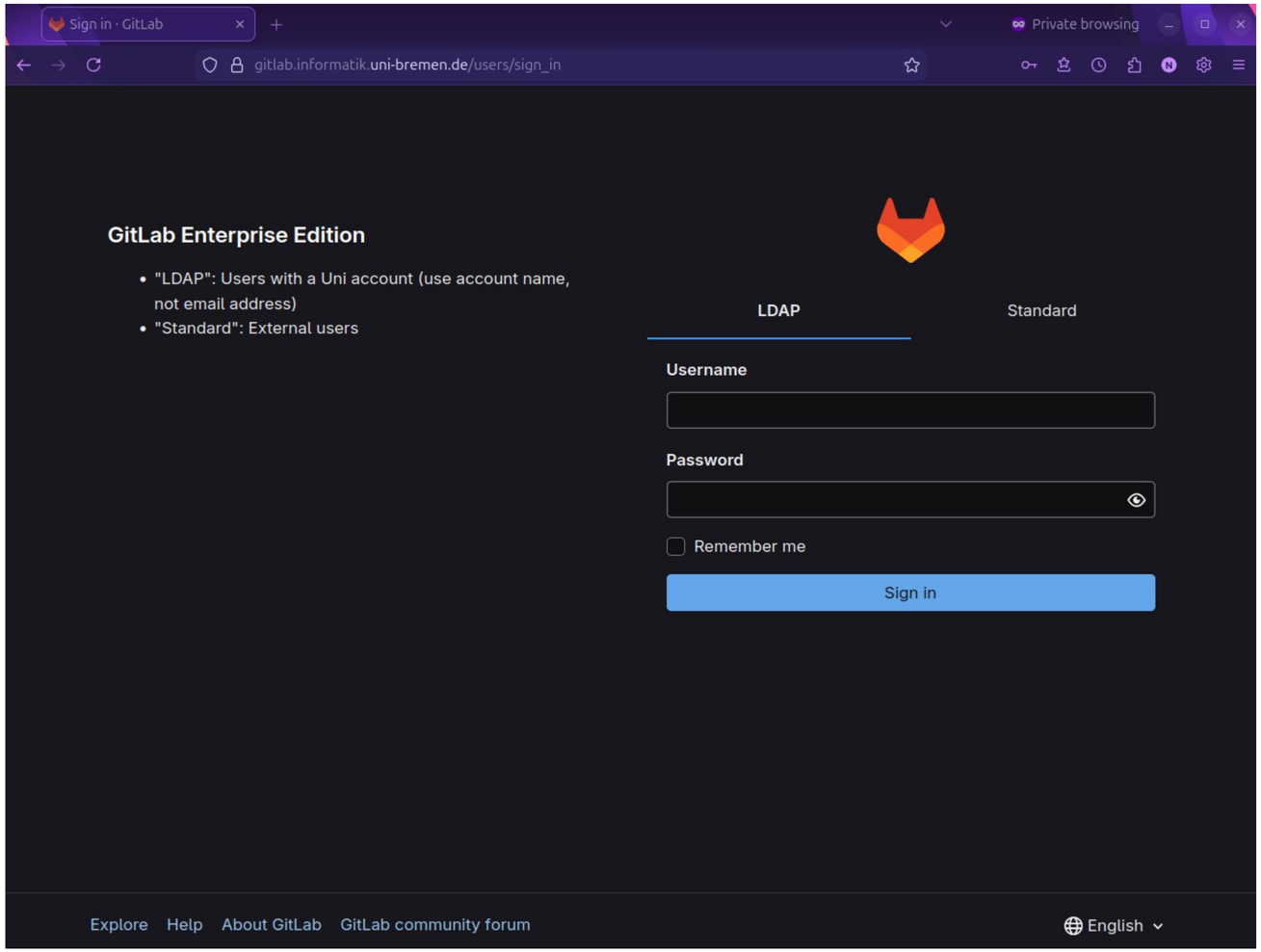
1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

Interactive Demonstration with VS Code

1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

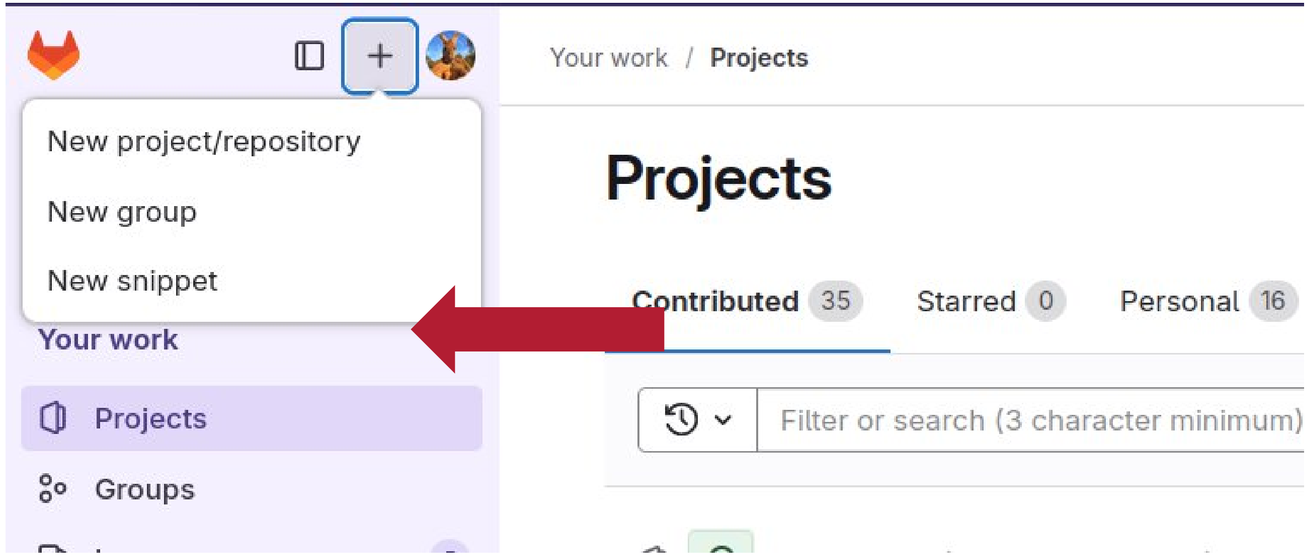
Creating a Repository

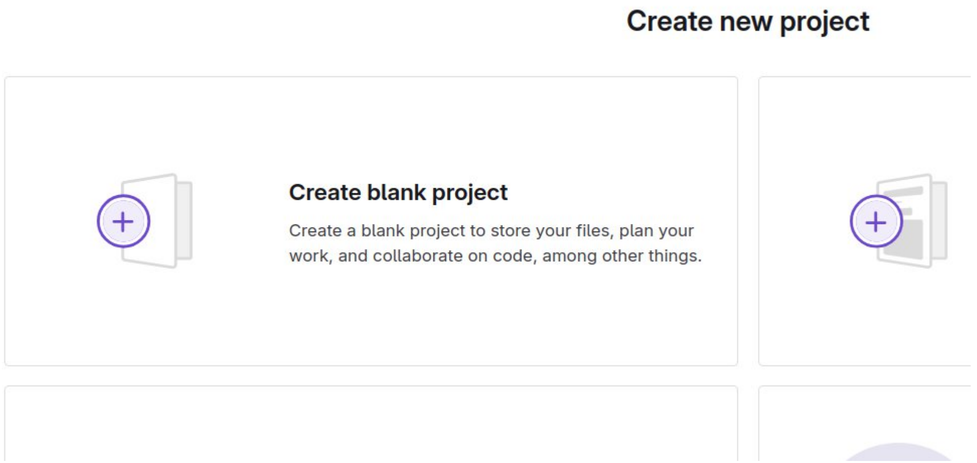
1. Log into GitLab with your university credentials. Link: https://gitlab.informatik.uni-bremen.de



Creating a Repository

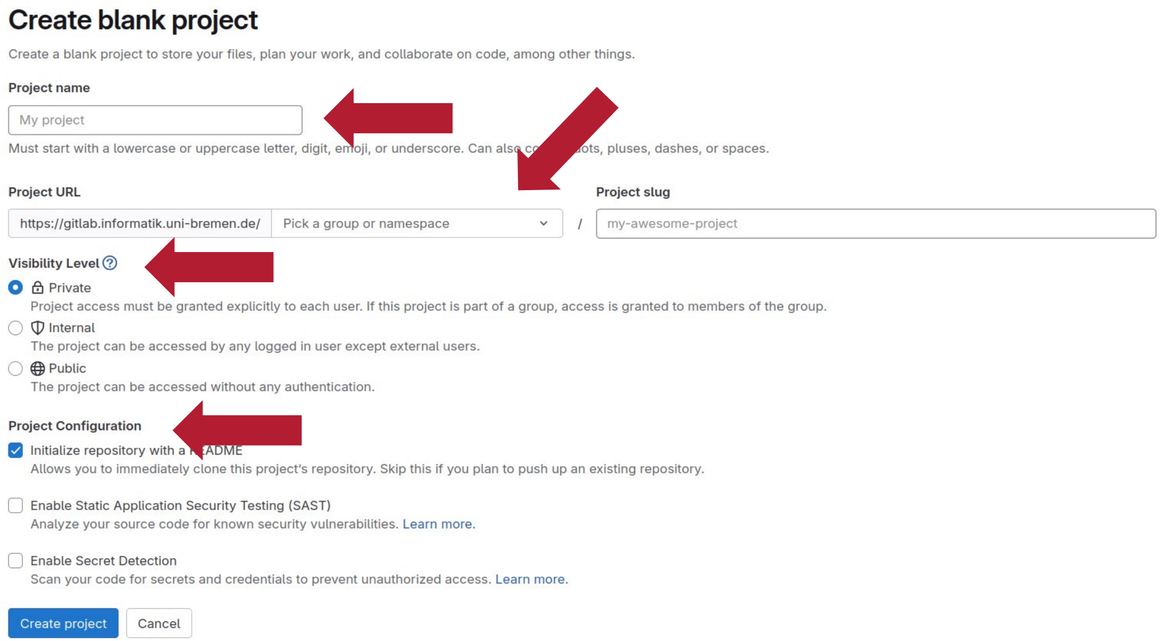
1. Log into GitLab with your university credentials. Link: https://gitlab.informatik.uni-bremen.de
2. On the top left, select “New Project / Repository”, then “Empty Project”.





Creating a Repository

1. Log into GitLab with your university credentials. Link: https://gitlab.informatik.uni-bremen.de
2. On the top left, select “New Project / Repository”, then “Empty Project”.
3. Select a projekt name, namespace, visibility level, and projekt configuration.



Interactive Demonstration with VS Code

1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

Clone a Repository - `git clone <repository url>`

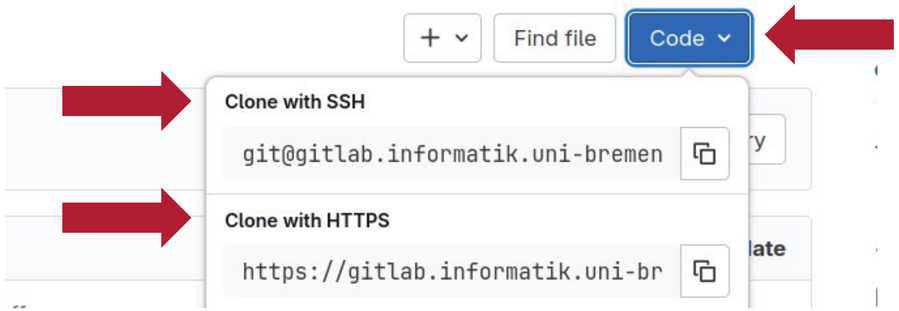
1. Choose a method of authentification.
   - HTTPS or SSH
   - HTTPS either via password or auth-token

Clone a Repository - `git clone <repository url>`

1. Choose a method of authentification.
   - HTTPS or SSH
2. Open your target folder either in your IDE or the terminal.

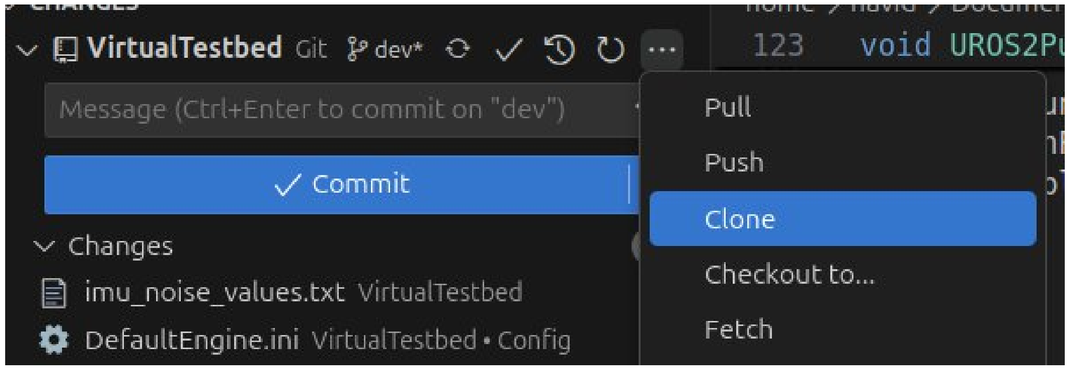
Clone a Repository - `git clone <repository url>`

1. Choose a method of authentification.
   - HTTPS or SSH
2. Open your target folder either in your IDE or the terminal.
3. Copy the Url of the repo.
   - In top right, under “Code”.
   - Pay attention to the Url-type!



Clone a Repository - `git clone <repository url>`

1. Choose a method of authentification.
   - HTTPS or SSH
2. Open your target folder either in your IDE or the terminal.
3. Copy the Url of the repo.
4. Clone the repo via your IDE or the terminal!



Interactive Demonstration with VS Code

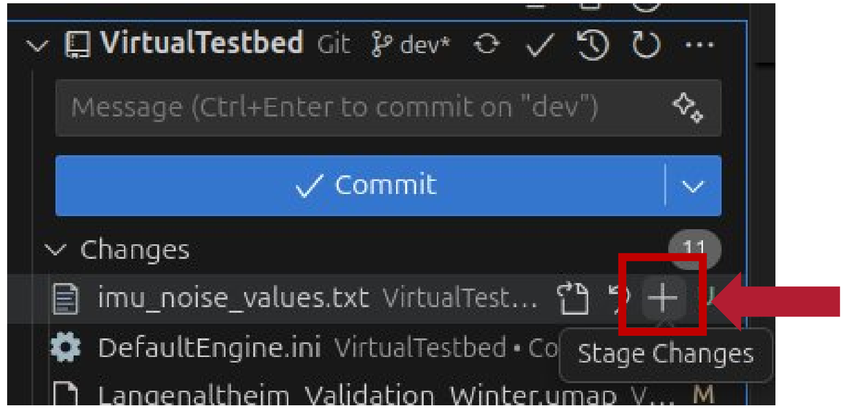
1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main` .

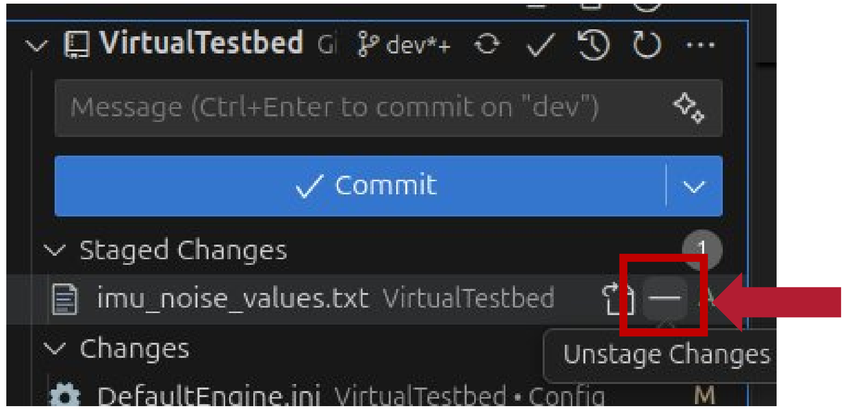
Interactive Demonstration with VS Code

1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

Staging Changes - `git add <file path>`

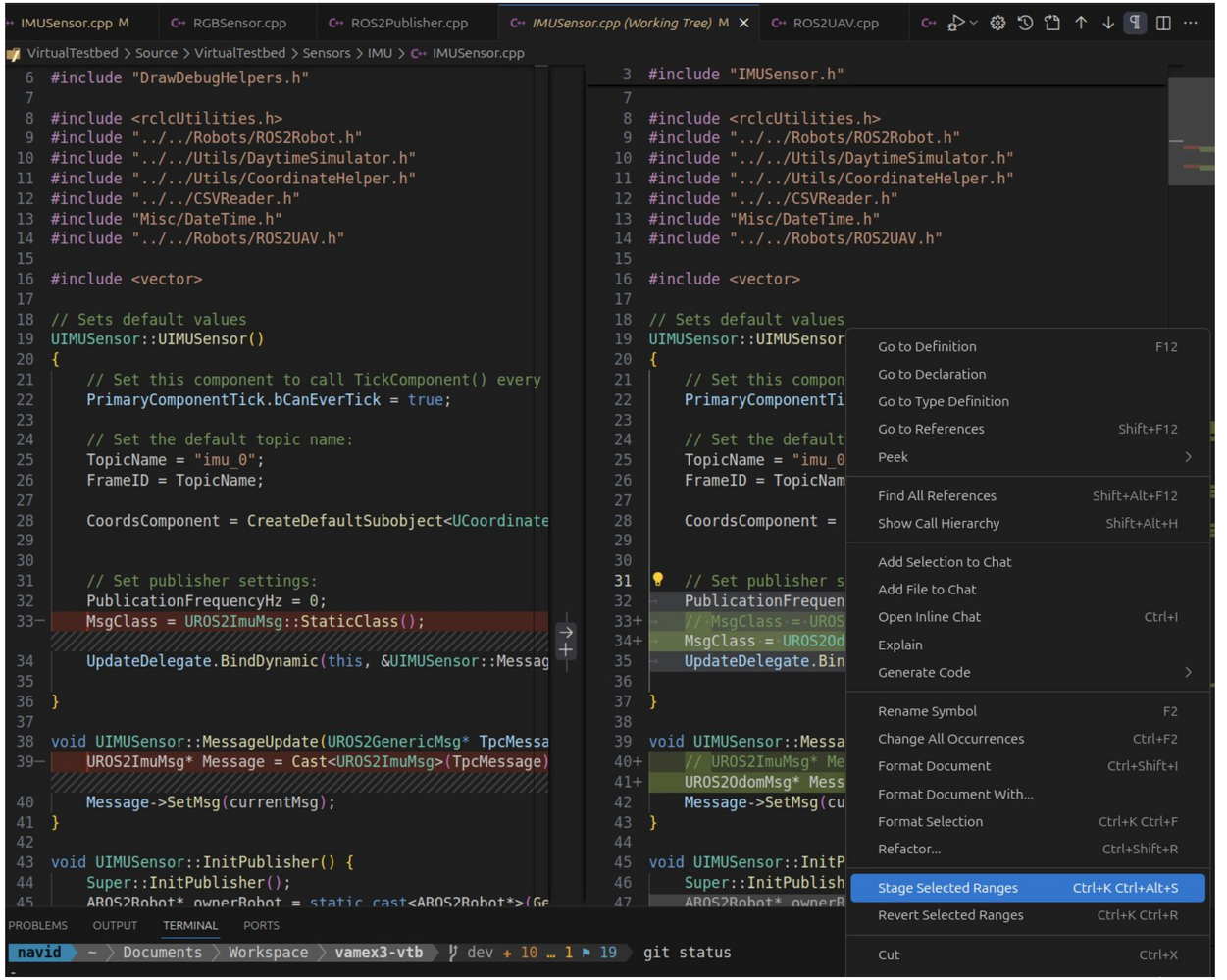
1. Select files for staging (+) or deselect them (-).





Staging Changes - `git add <file path>`

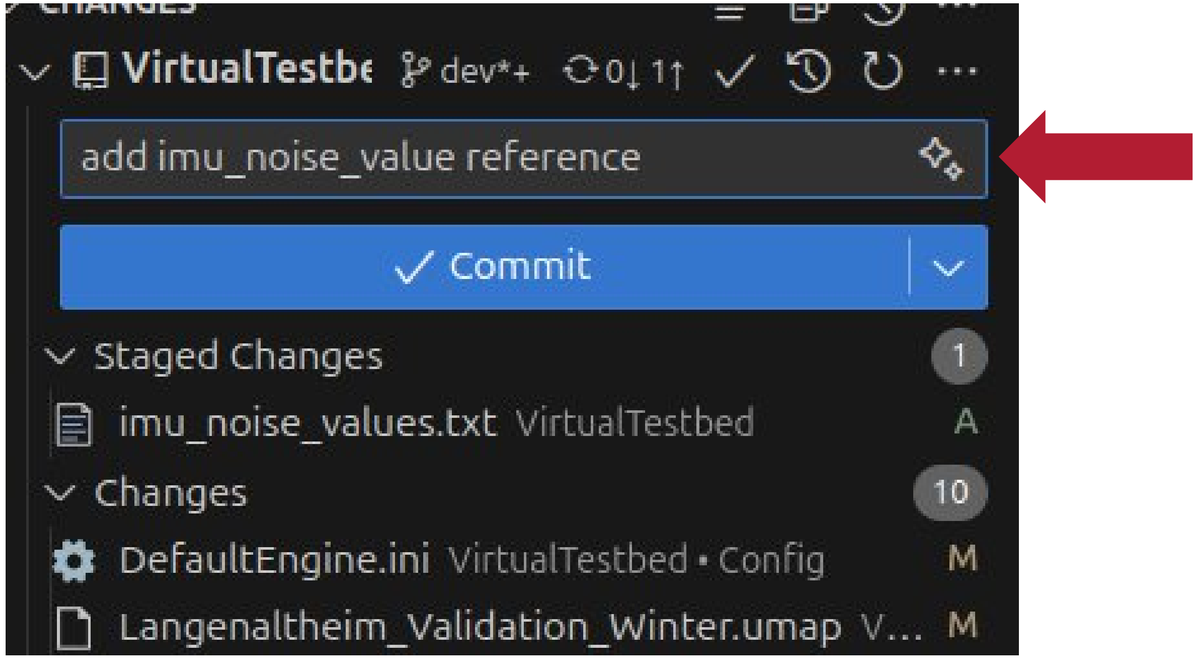
1. Select files for staging (+) or deselect them (-).
2. It’s also possible to only stage parts of a changed file.
   - Can be a bit flimsy at times.



Changes between two revisions of the same file in the VS-Code diff editor. It is possible to mark sections and stage those separate from other changes.

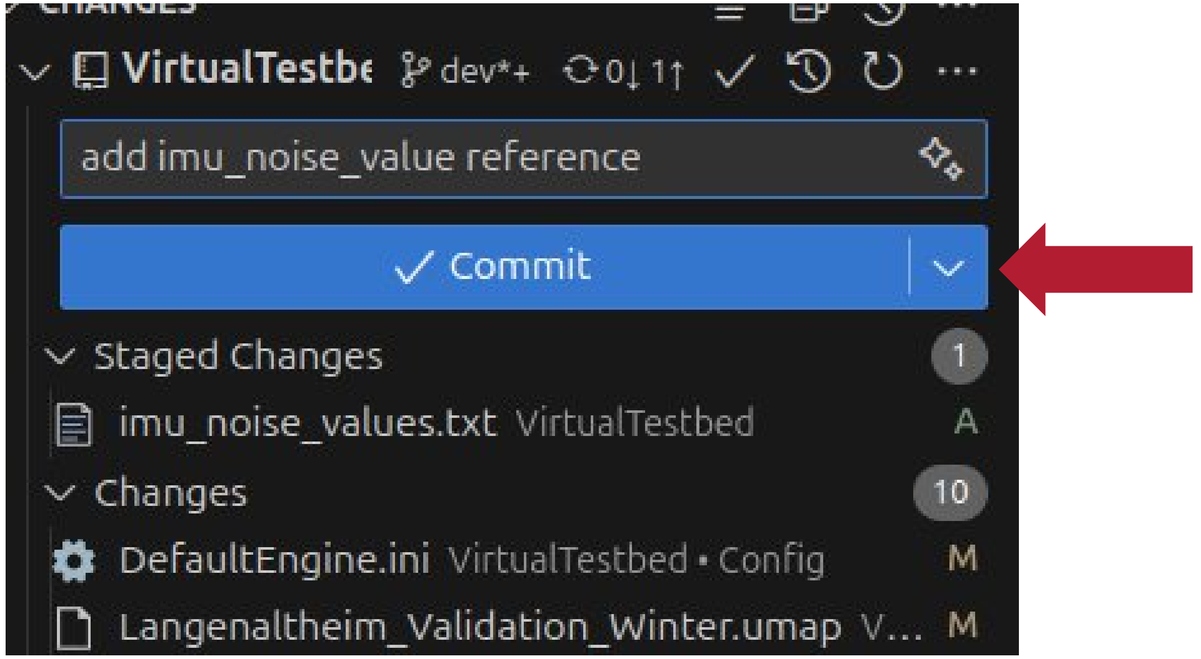
Committing Changes - `git commit -m "<message>"`

1. Give your changes a descriptive commit message.
   - This step is not optional!!!



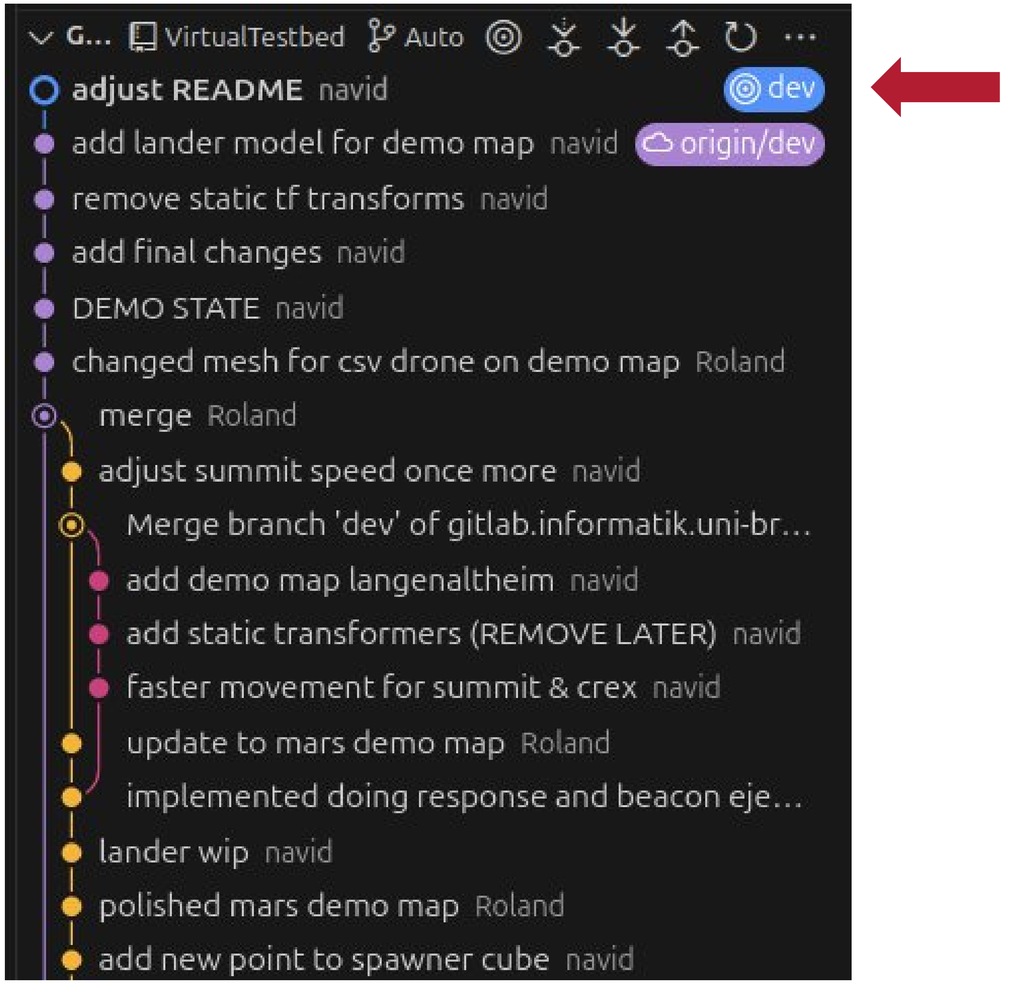
Committing Changes - `git commit -m "<message>"`

1. Give your changes a descriptive commit message.
   - This step is not optional!!!
2. With the “Commit”-Button, your changes are added to the local Commit History.



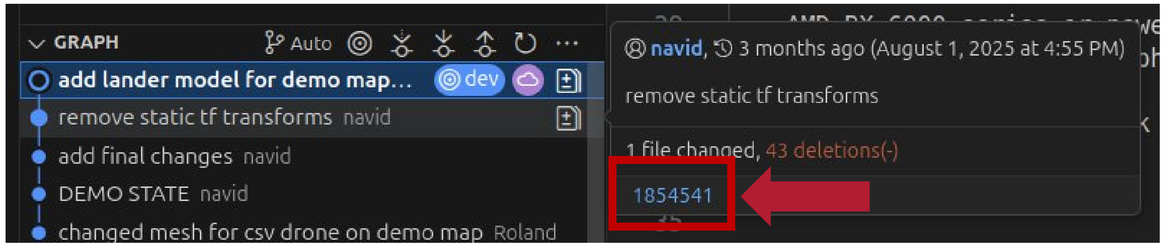
Committing Changes - `git commit -m "<message>"`

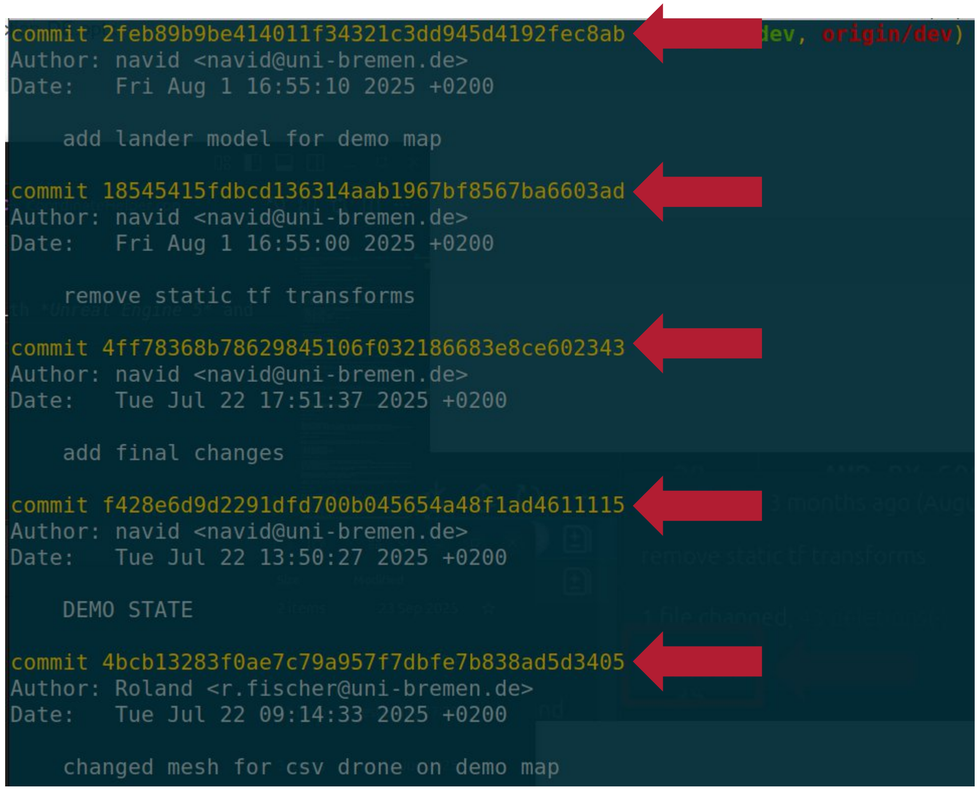
1. Give your changes a descriptive commit message.
   - This step is not optional!!!
2. With the “Commit”-Button, your changes are added to the local Commit History.
   - But not yet to the remote repository (GitLab, GitHub, etc.). Usually denoted with `origin/` .



Commit Hashes & History

- Each commit has a unique hash.
- It is commonly used to reference it.
  - Usually the short hash suffices.
- Always listed in the commit history.
  - `git log`



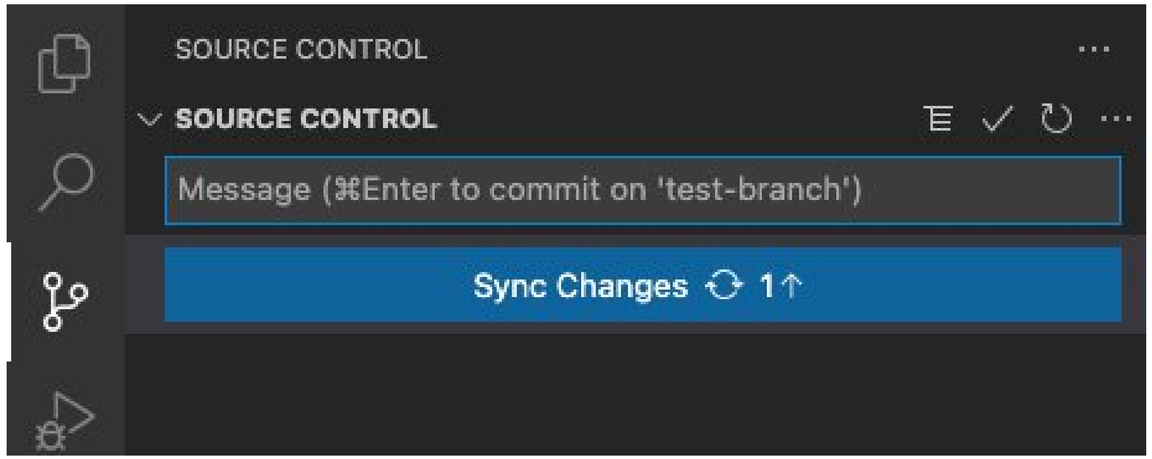


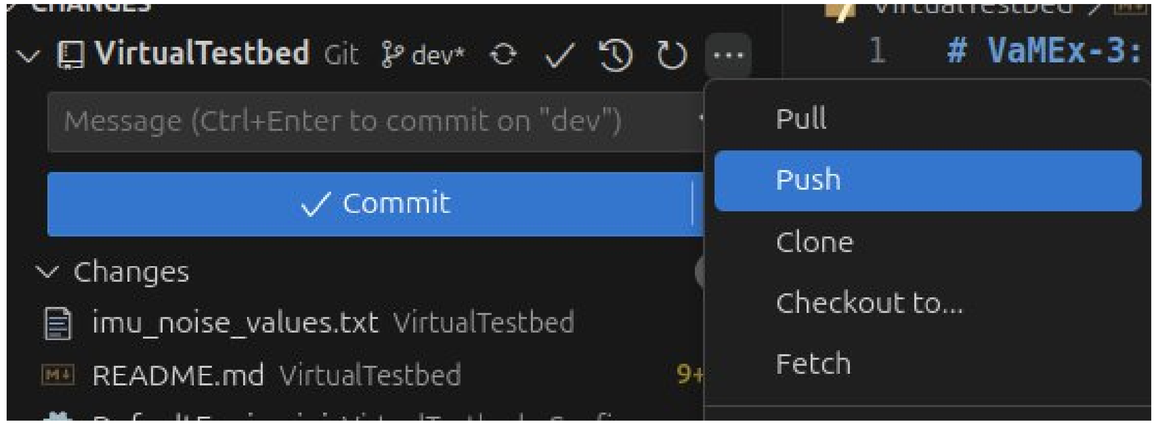
Interactive Demonstration with VS Code

1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

Pushing Commits - `git push`

1. Push the changes via one of the following two options:
   - “Sync Changes”; only shown if there are no more local changes to be committed
   - source control context menu





Pushing Commits - `git push`

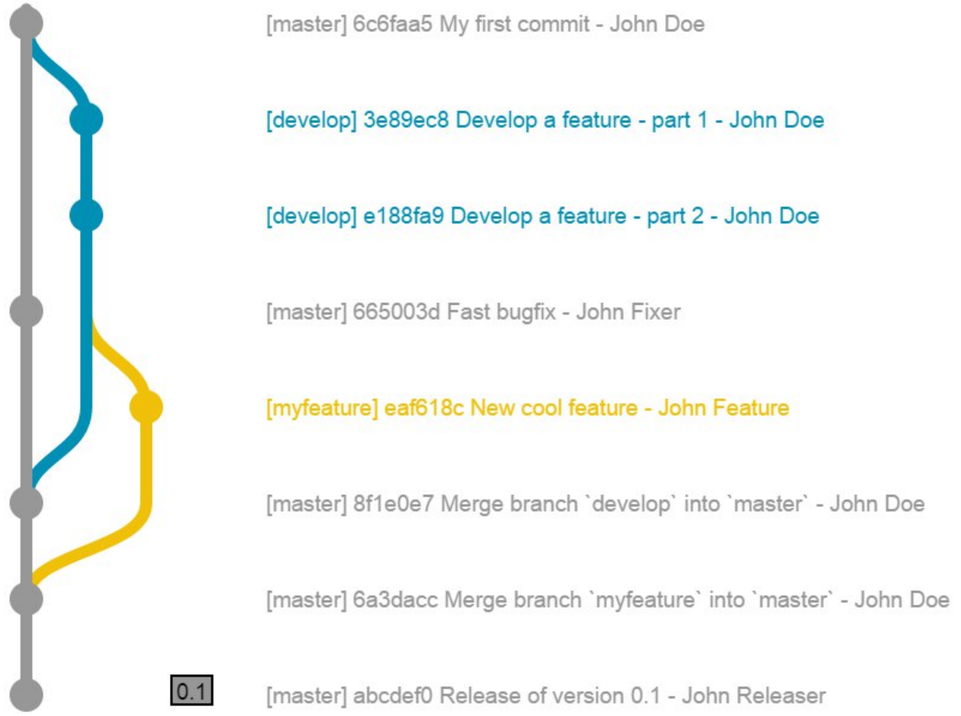
1. Push the changes
2. If someone has already pushed on that branch, there can be problems.
   - Changes need to first be integrated locally.
   - There are many approaches:
     - Branch & Merge
     - Rebase
     - …

Interactive Demonstration with VS Code

1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

Branches

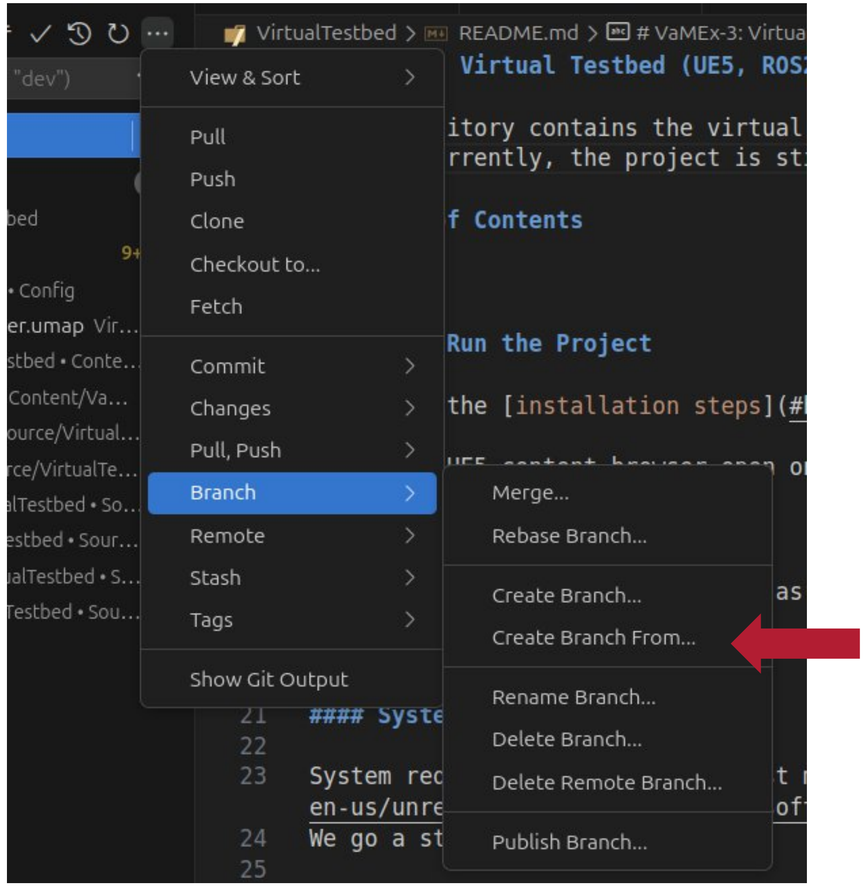
- “alternate states” of your project
  - Why? Gives the possibility to work on something in isolation, without constantly needing to reintegrate external changes.
- reintegration via a „merge”
  - Creates a merge commit.
  - In the case of a merge conflict adjustments need to be made.



Quelle: GitHub/frappe/charts

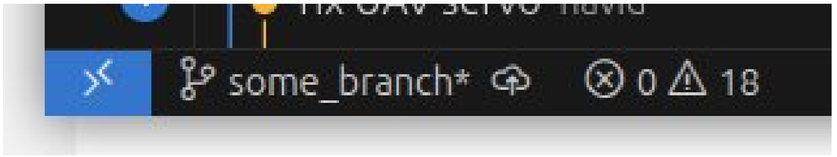
Creating a Branch - `git branch -c <branch name>`

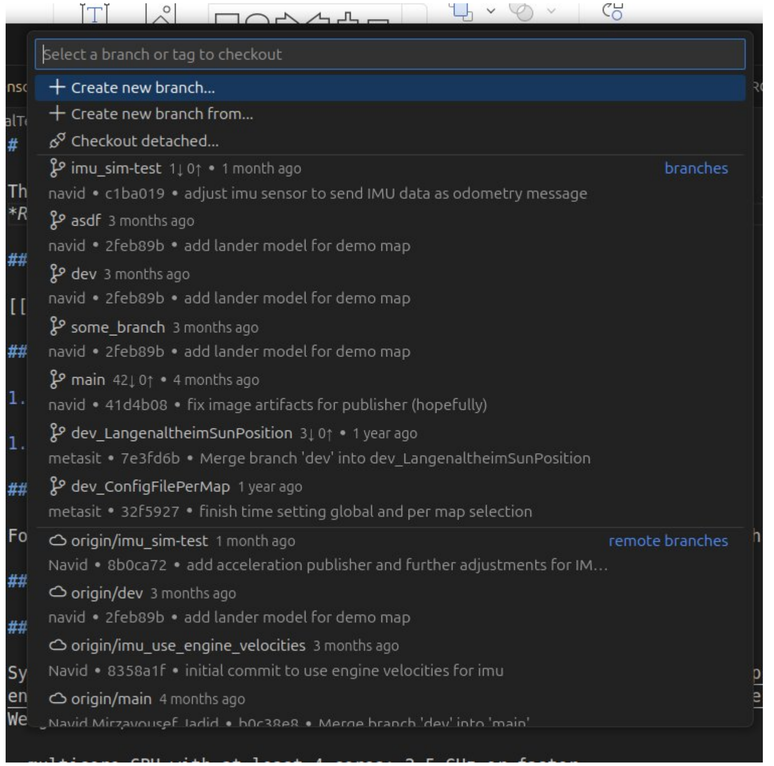
1. Create a new branch based on `main` .



Creating a Branch - `git branch -c <branch name>`

1. Create a new branch based on `main` .
2. Branches can also be created with the button on the bottom left.
   - `git switch <branch>`
   - Can also create a new branch from there.





Interactive Demonstration with VS Code

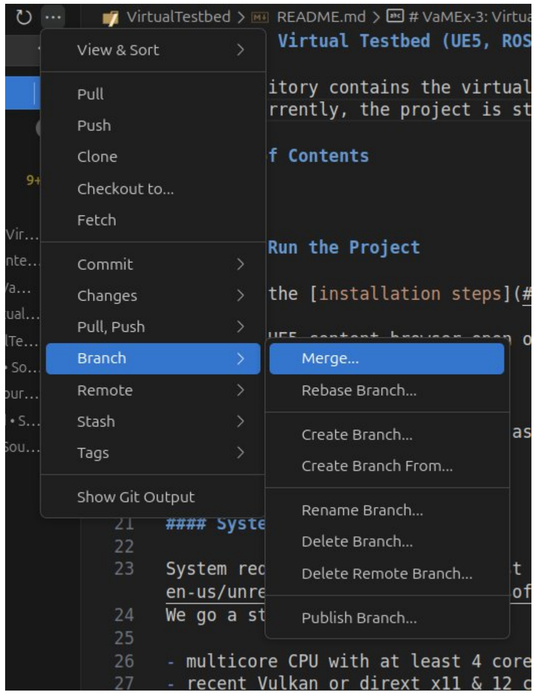
1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

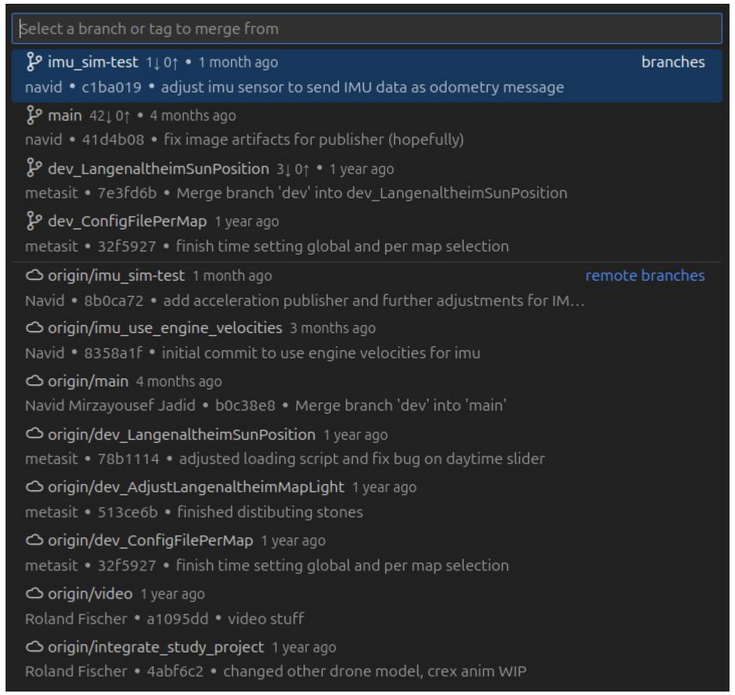
Interactive Demonstration with VS Code

1. Create a repository on GitLab.
2. Clone the repository (either via HTTPS or SSH).
3. Add a file to your local repository (e.g. a Markdown-file; `.md`).
4. Add some text to that file.
5. „Stage” your changes and „commit” them with a message.
6. „Push” your changes.
7. Create a new branch.
8. Make changes to your file, commit them, and then push them.
9. Merge your local branch into `main`.

Merging Branches - `git merge <branch name>`

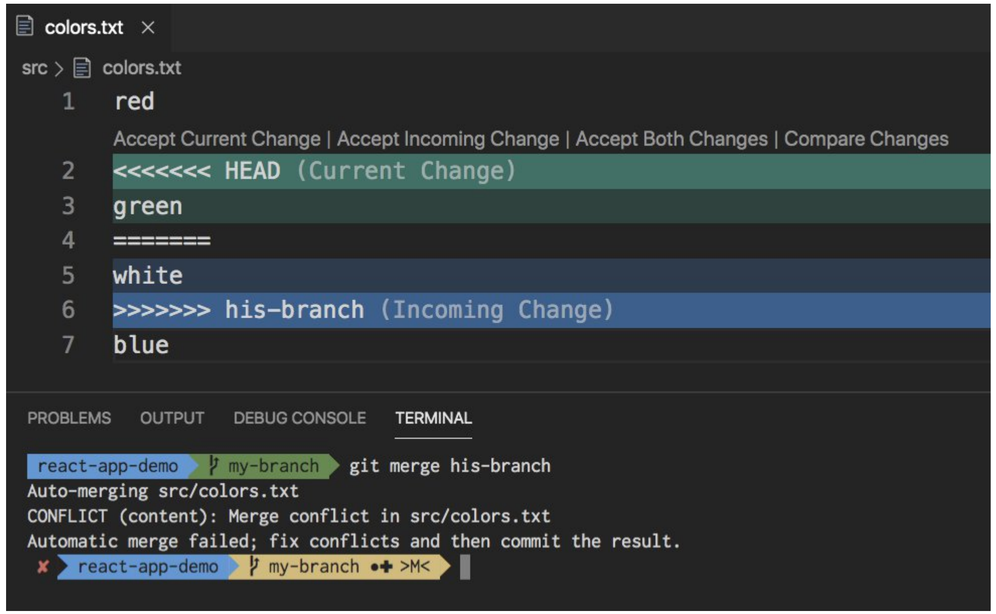
1. Switch to `main`.
2. Merge your new branch into `main`.
   - Merge always incorporates changes into the active branch.





Merge Conflicts

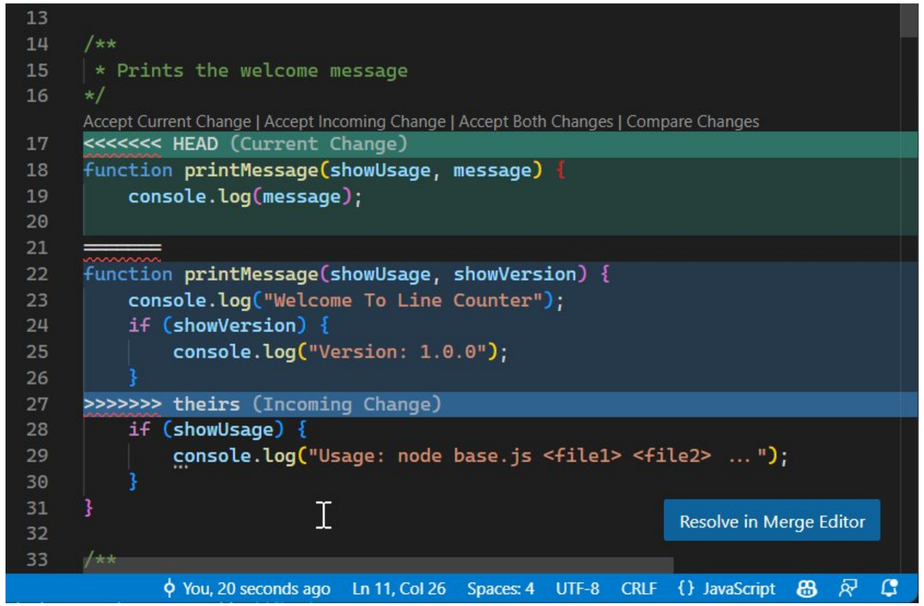
- What if multiple people worked on the same file and push / merge to the same branch?
  - We get a so-called „merge conflict”.
    - Now it is necessary to decide which changes to keep and which to discard.
    - In case of uncertainty, a merge can always be aborted.
      - `git merge --abort`

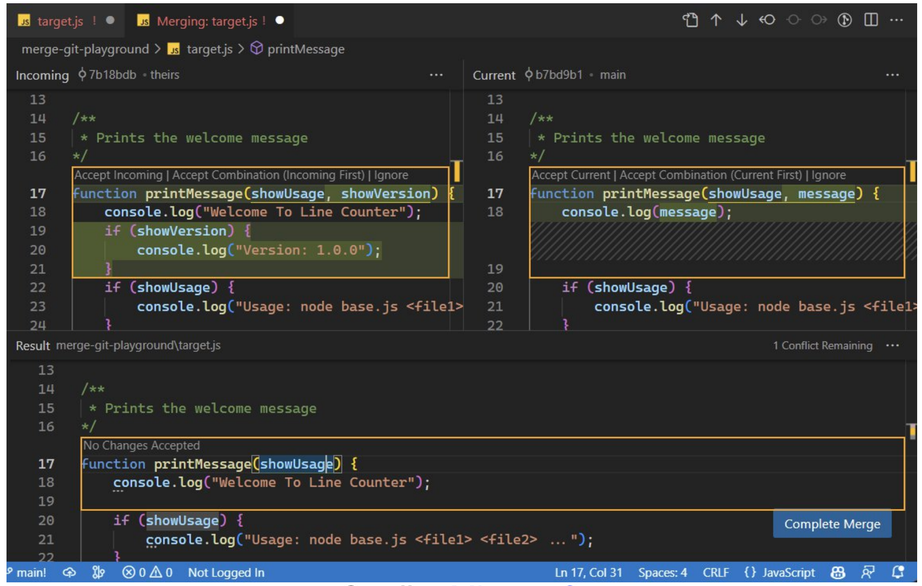


Quelle: ihatetomatoes.net

Merge Conflicts

- VS Code has built-in solution.
- Inline Merge-Editor (top)
  - Current Change: local state
  - Incoming Change: incoming changes
  - Both: combination
- Three-Way Merge Editor (bottom)
  - Left incoming, right current, bottom result.
- Options can be selected via the GUI. But changes can also be directly made in the editor.

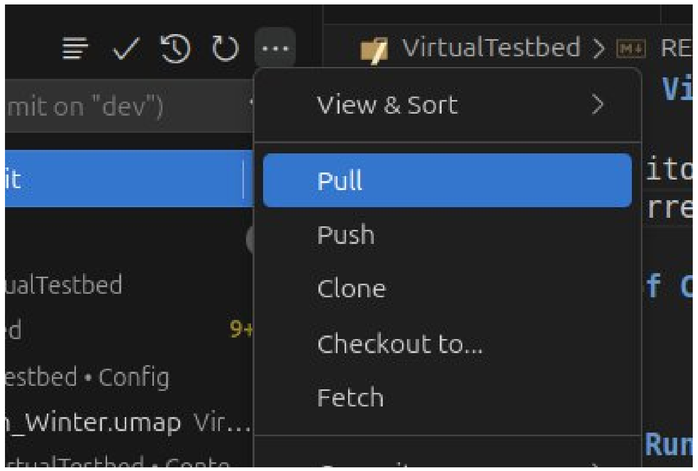




Quelle: Microsoft

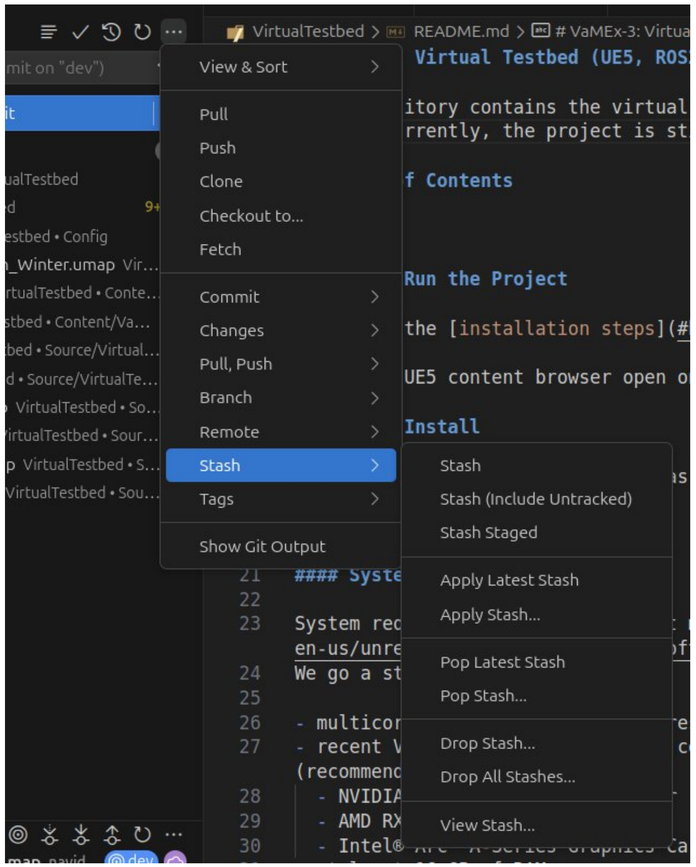
Pulling Changes / Commits - `git pull`

- `git pull` : Retrieves changes from the remote repository.
  - Can create problems, if the same file was edited both locally and remotely.
    - If changes have been committed locally, then we get a merge conflict.
    - If not, then the local changes can be “stashed” away and reintegrated after pulling.



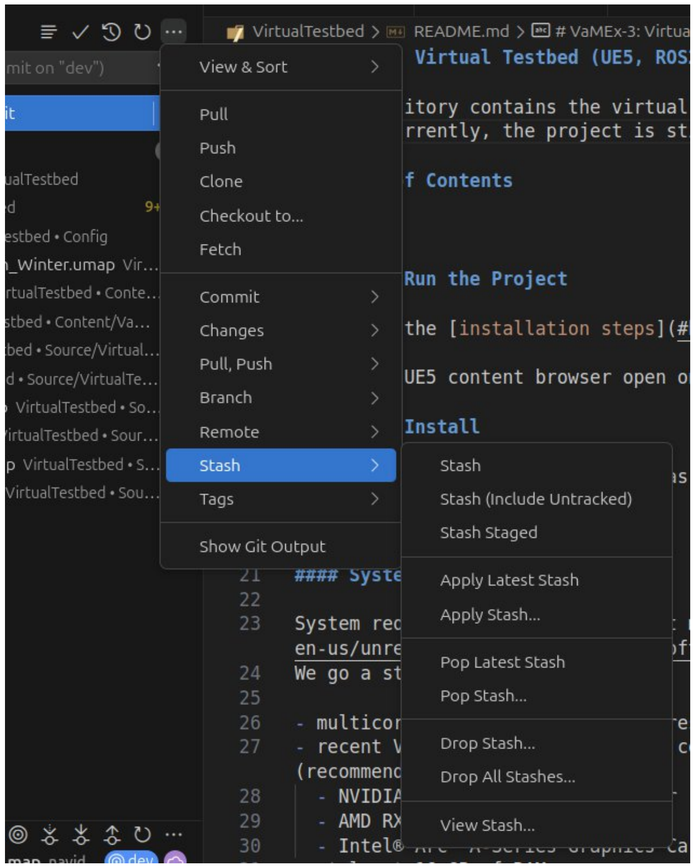
Stashes

- Sometimes it is useful to “stash away” local changes.
- `git stash` : Put the local changes into a stash.
  - Warning: Untracked files are not stashed by default.
  - `git stash -u`
- `git stash list` : Show the current list of stashes.



Stashes

- `git stash pop [stash number]`:
  Remove the last / newest entry in the list of stashes and apply the changes. There is the option to select an earlier stash by passing the `[stash number]` (counts from 0).
- `git stash apply [stash number]`:
  Same as `stash pop`, but keeps the entry.
- The stash entry is always kept for safety, if there is a conflict during `pop` or `apply`.



Other Useful Commands

- `git checkout <commit hash>` : Change to the state of the specified commit on the current branch.
- `git checkout -` : Change to the latest state on the active branch.
- `git checkout <branch name>` : Switch to the specified branch.
- `git checkout -b <branch name>` : Create a new branch and switch to it.
- `git revert <commit hash>`: Revert / Undo the changes of the specified commit. Creates a revert commit; also keeps the original commit in the history.

A Few Final Words of Advice

- Git / VCS is both powerful and essential in software development!
- This was just an introduction. You will learn much more in your own work.
  - For most things, there are already online tutorials or forum entries.
  - Chat Bots can also be of help.
  - Be cautious in both cases! Wrong decisions / commands can lose you hours of prior work.

Additional Resources

- Git Explained in 100 Seconds (Youtube)
- Learn Git in 15 Minutes (Youtube)
- A Grip On Git (https://agripongit.vincenttunru.com/)
  - short visual introduction to git
  - ~11 min read
- Learn Git Branching (https://learngitbranching.js.org/)
  - interactive tutorial for git in the webbrowser
  - ~30-60 min