



AG CGVR

Einführung in

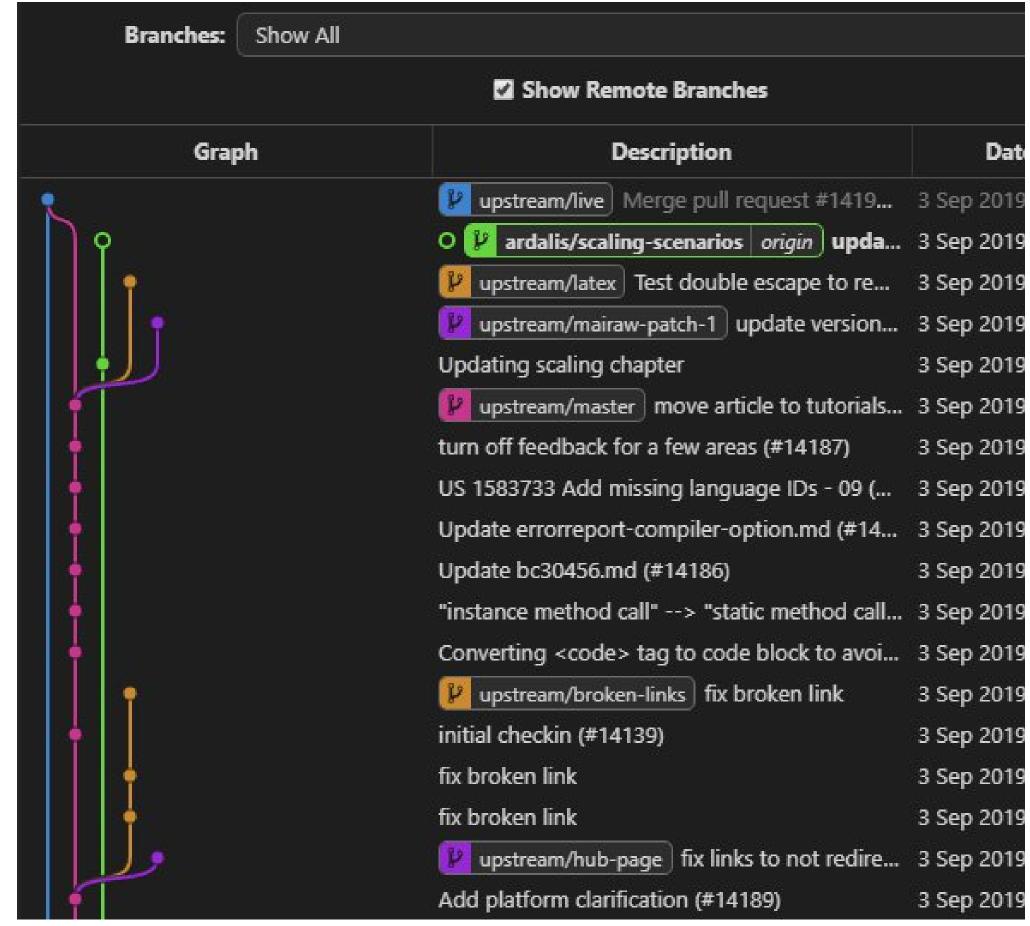




Git Einführung



- Was ist git?
 - Versionskontrolltool (VCS) zum kooperativen Arbeiten an Software / Code
 - Codeänderungen können lokal oder auf einem Server / der Cloud gespeichert werden.
 - Änderungen können dann synchronisiert und integriert werden.



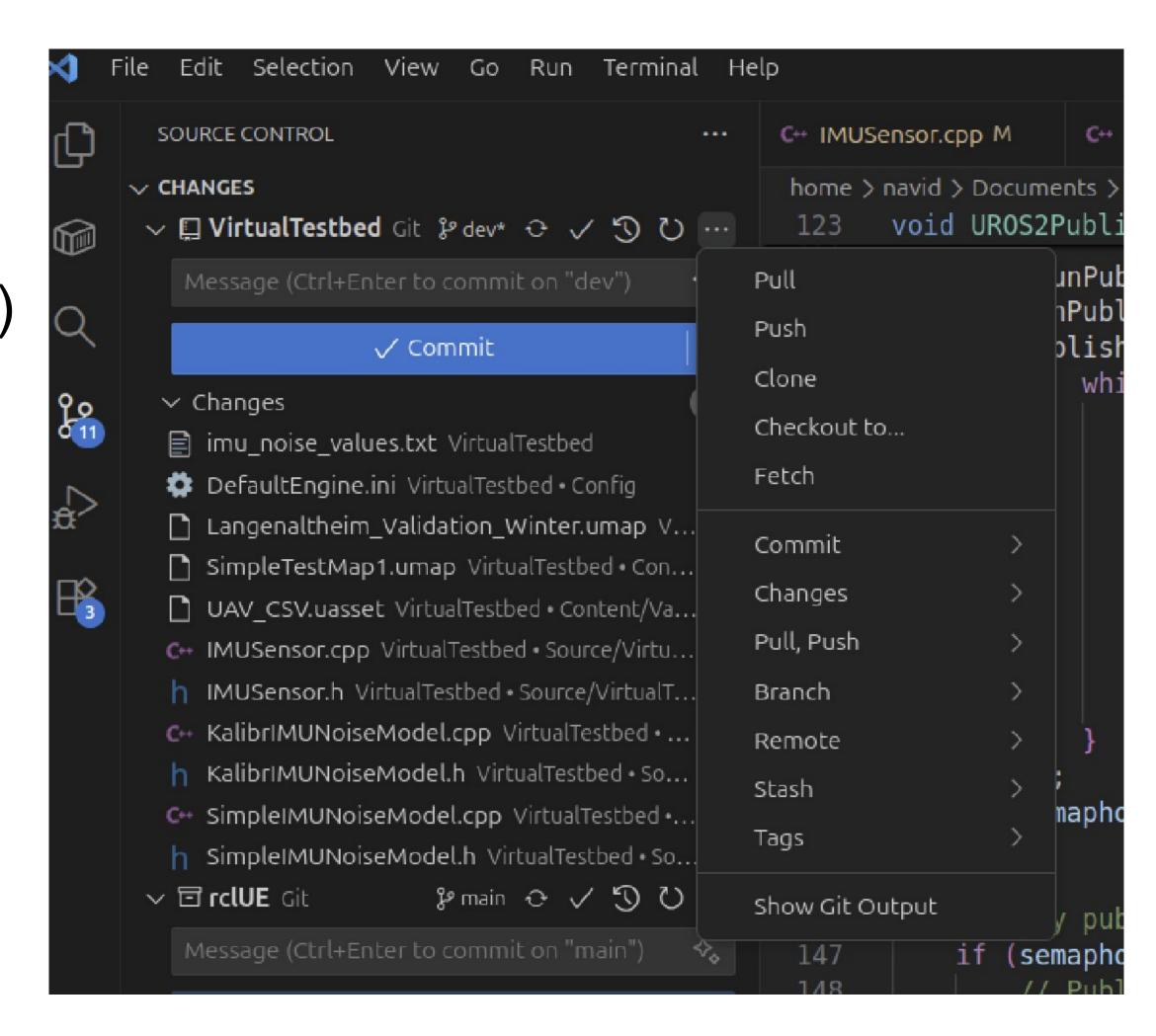
Quelle: <u>ardalis.com</u>



Git Einführung



- Wie nutzt man git?
 - Kann über das Terminal oder GUI-Anwendungen (z.B. Github-Desktop) bedient werden.
 - Heutzutage haben eigentlich alle Entwicklungsumgebungen git-Einbindung, also auch VS Code.
- Installation
 - Auf Linux & Mac i.d.R. vorinstalliert.
 - Auf Windows: https://git-scm.com/downloads/win

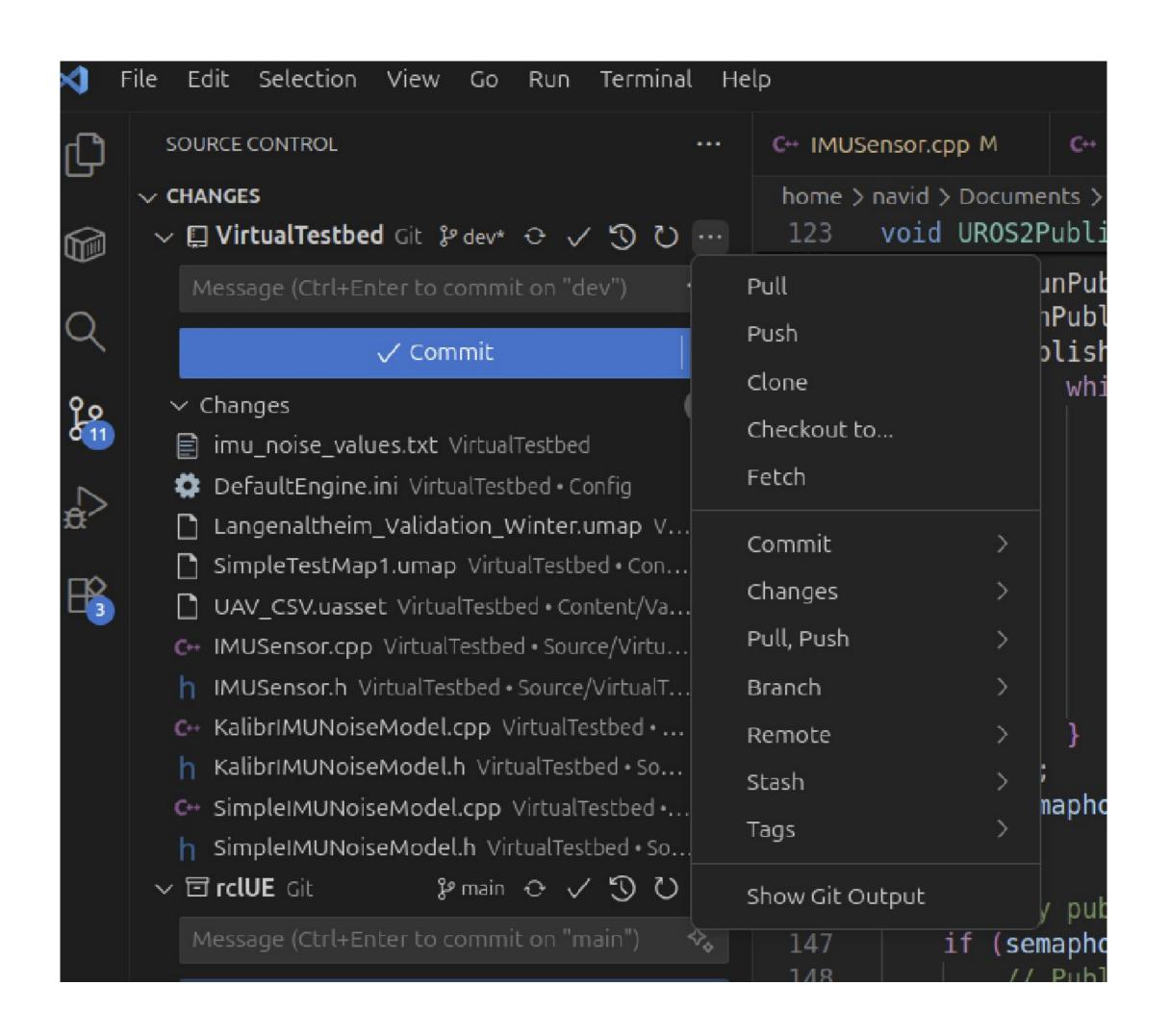




Git Einführung



- Grundlegende Begriffe
 - Repository
 - Clone
 - **Commit & Commit History**
 - Branch & Branch Tree
 - Staged Changes
 - Pull / Push
 - Stash
 - Merge







- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.





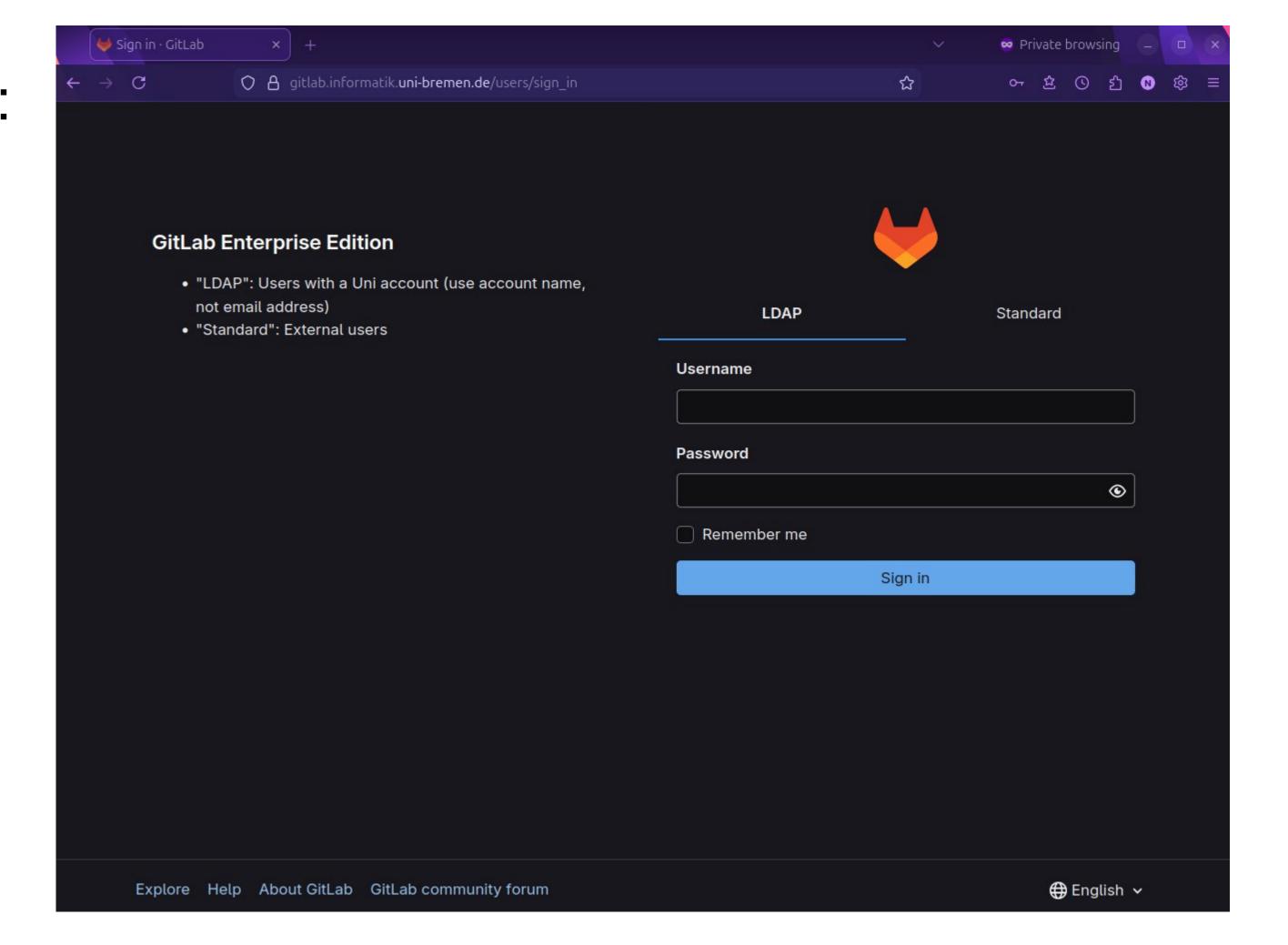
- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.



Repository anlegen



1. Loggt euch auf GitLab mit eurem Uni-Account ein. Link: https://gitlab.informatik.uni-bremen.de

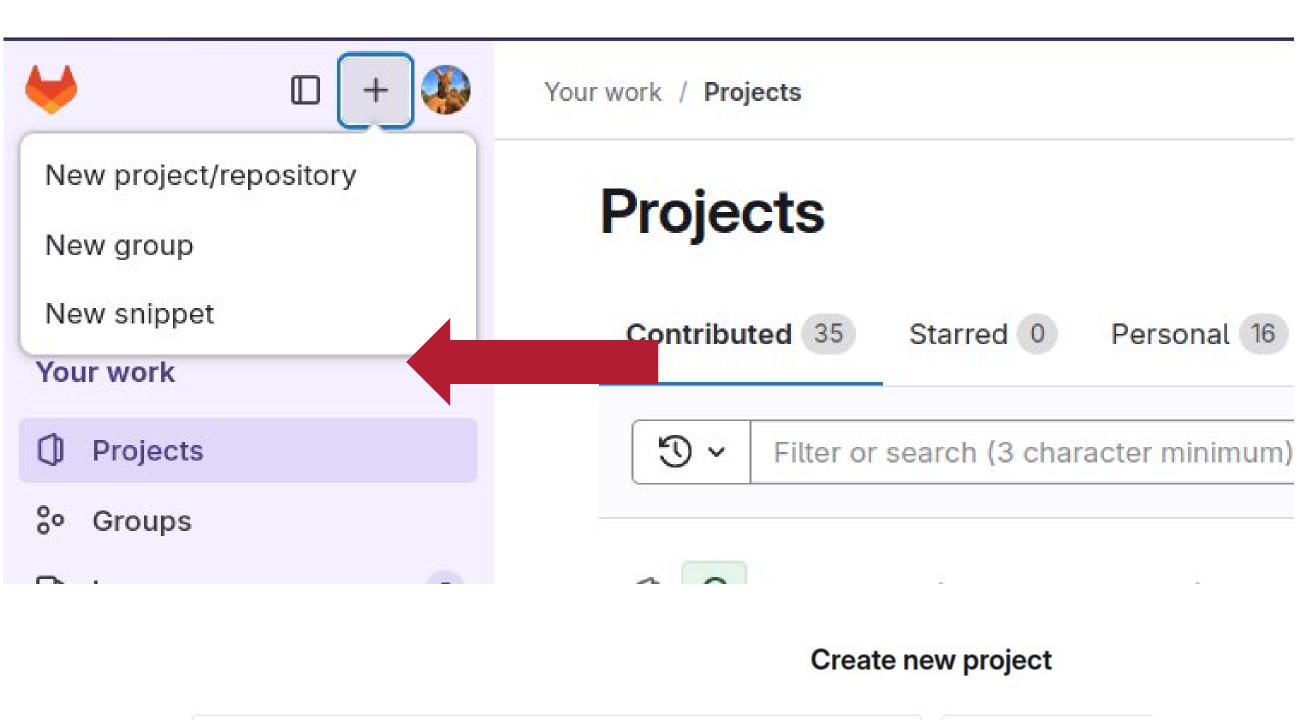


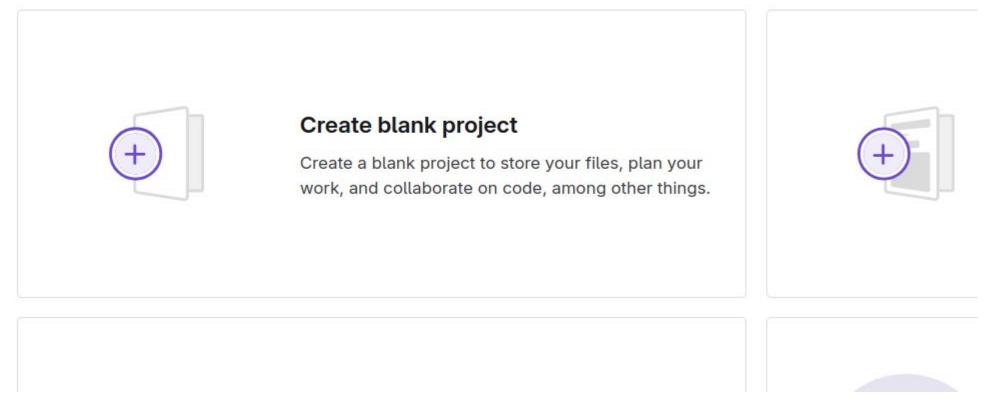


Repository anlegen



- 1. Loggt euch auf GitLab mit eurem Uni-Account ein. Link: https://gitlab.informatik.uni-bremen.de
- 2. Wählt oben links "Neues Projekt" aus. Danach "Leeres Projekt".



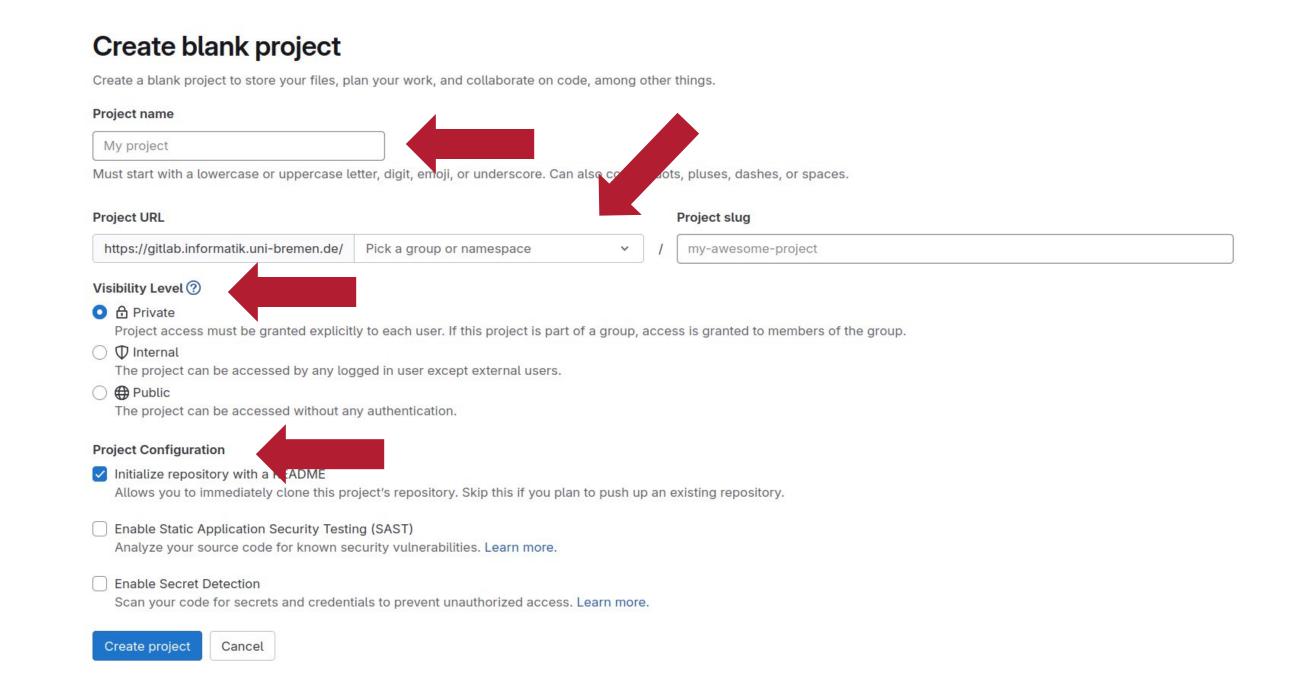




Repository anlegen



- 1. Loggt euch auf GitLab mit eurem Uni-Account ein. Link: https://gitlab.informatik.uni-bremen.de
- 2. Wählt oben links "Neues Projekt" aus. Danach "Leeres Projekt".
- 3. Wählt einen Projektnamen, Namespace, Visibility Level, und Projektkonfiguration.







- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.





- 1. Authentifizierung festlegen
 - HTTPS oder SSH
 - HTTPS entweder mit
 Passwort oder Auth-Token



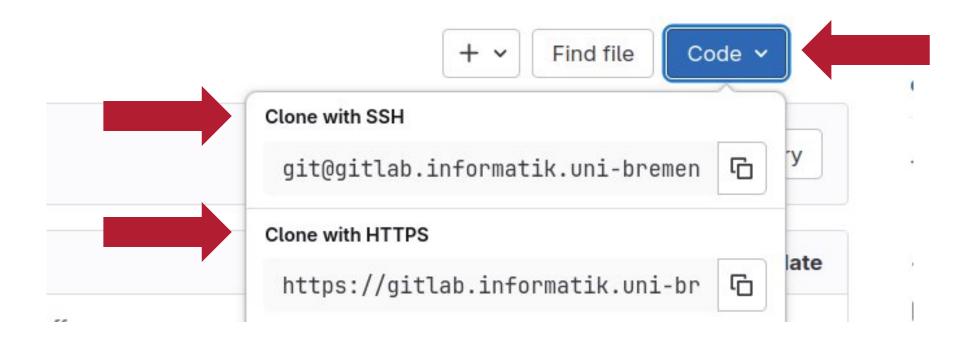


- 1. Authentifizierung festlegen
 - HTTPS oder SSH
- 2. Zielordner entweder in IDE oder Terminal öffnen.





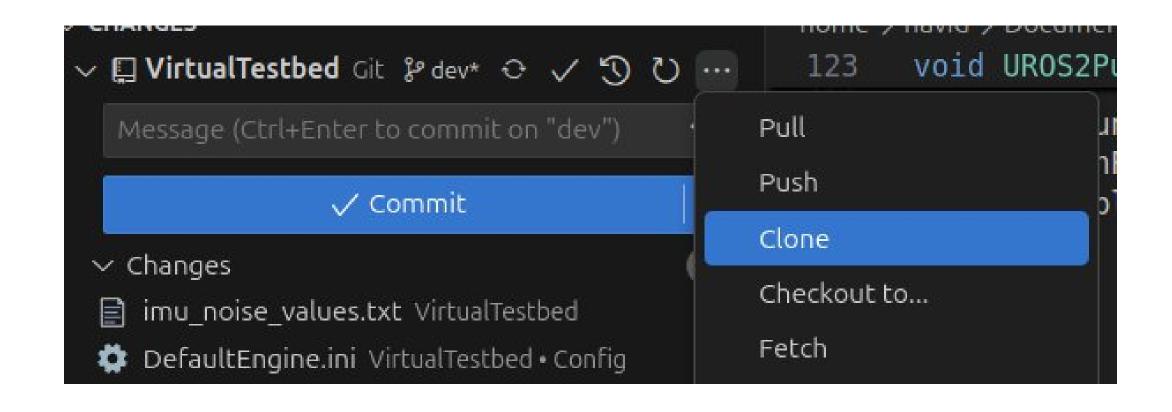
- 1. Authentifizierung festlegen
 - HTTPS oder SSH
- 2. Zielordner entweder in IDE oder Terminal öffnen.
- 3. Url des Repo kopieren.
 - oben rechts unter "Code"
 - Achtet auf den Url-Typen!







- 1. Authentifizierung festlegen
 - HTTPS oder SSH
- 2. Zielordner entweder in IDE oder Terminal öffnen.
- 3. Url des Repo kopieren.
 - Oben rechts unter "Code"
- 4. Repository entweder über IDE oder Terminal klonen.







- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.





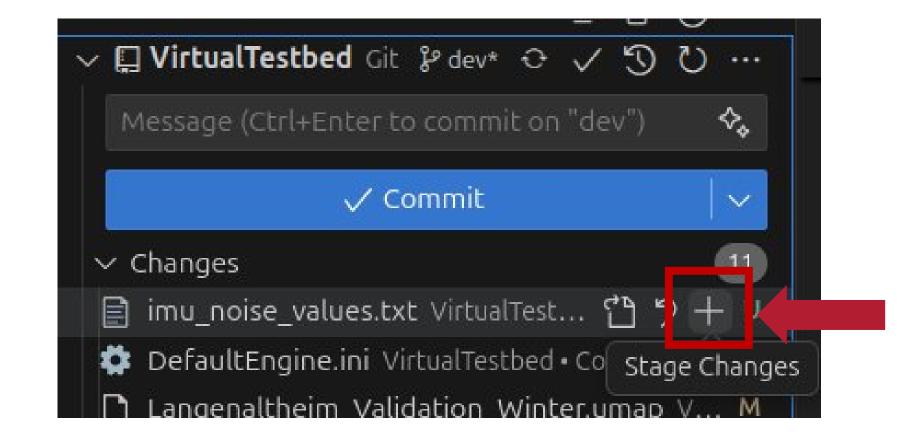
- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.

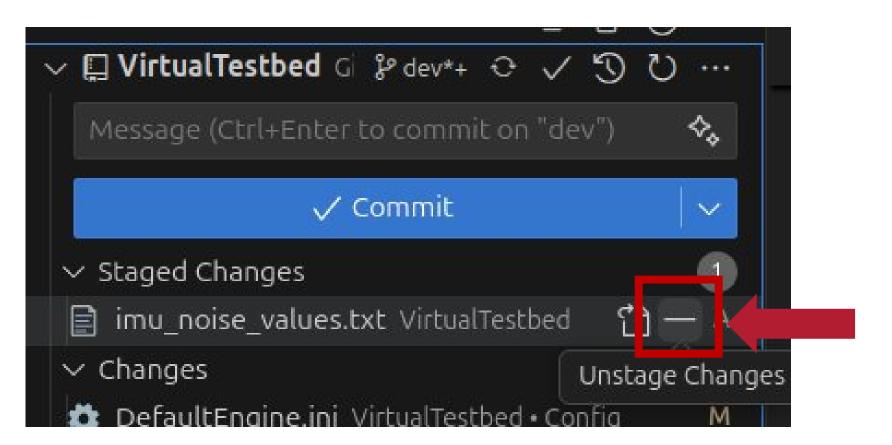


Änderungen stagen - git add <file path>



 Dateien zum stagen auswählen (+) oder entfernen (-).







Änderungen stagen - git add <file path>



- Dateien zum stagen auswählen (+) oder entfernen (-).
- 2. Man kann auch einzelne Abschnitte von Dateien stagen.
 - Ist aber teils "flimsig".

```
C++ ROS2Publisher.cpp
                                                         C++ IMUSensor.cpp (Working Tree) M X C++ ROS2UAV.cpp
🎢 VirtualTestbed > Source > VirtualTestbed > Sensors > IMU > 🖙 IMUSensor.cpp
                                                                        3 #include "IMUSensor.h"
    #include "DrawDebugHelpers.h"
    #include <rclcUtilities.h>
                                                                        8 #include <rclcUtilities.h>
    #include "../../Robots/ROS2Robot.h"
                                                                        9 #include "../../Robots/ROS2Robot.h"
    #include "../../Utils/DaytimeSimulator.h"
                                                                       10 #include "../../Utils/DaytimeSimulator.h"
    #include "../../Utils/CoordinateHelper.h"
                                                                       #include "../../Utils/CoordinateHelper.h"
    #include "../../CSVReader.h"
                                                                       #include "../../CSVReader.h"
    #include "Misc/DateTime.h"
                                                                       13 #include "Misc/DateTime.h"
    #include "../../Robots/ROS2UAV.h"
                                                                       #include "../../Robots/ROS2UAV.h"
    #include <vector>
                                                                       16 #include <vector>
    // Sets default values
                                                                       18 // Sets default values
    UIMUSensor::UIMUSensor()
                                                                          UIMUSensor::UIMUSensor
                                                                                                      Go to Definition
                                                                                                      Go to Declaration
        // Set this component to call TickComponent() every
                                                                               // Set this compor
        PrimaryComponentTick.bCanEverTick = true;
                                                                               PrimaryComponentTi
                                                                                                      Go to Type Definition
                                                                                                      Go to References
                                                                                                                                   Shift+F12
        // Set the default topic name:
                                                                               // Set the default
                                                                               TopicName = "imu 0
        TopicName = "imu 0";
        FrameID = TopicName;
                                                                               FrameID = TopicNam
                                                                                                      Find All References
        CoordsComponent = CreateDefaultSubobject<UCoordinate
                                                                               CoordsComponent =
                                                                                                      Show Call Hierarchy
                                                                                                                                 Shift+Alt+H
                                                                                                      Add Selection to Chat
        // Set publisher settings:
                                                                               // Set publisher s
                                                                                                      Add File to Chat
        PublicationFrequencyHz = 0;
                                                                               PublicationFrequen
                                                                                                      Open Inline Chat
        MsgClass = UROS2ImuMsg::StaticClass();
                                                                               MsgClass = UR0S20d
                                                                                                      Explain
        UpdateDelegate.BindDynamic(this, &UIMUSensor::Messag
                                                                               UpdateDelegate.Bin
                                                                                                      Generate Code
                                                                                                      Rename Symbol
    void UIMUSensor::MessageUpdate(UROS2GenericMsg* TpcMessa
                                                                                                      Change All Occurrences
                                                                                                                                    Ctrl+F2
                                                                           void UIMUSensor::Messa
        UROS2ImuMsg* Message = Cast<UROS2ImuMsg>(TpcMessage)
                                                                               // UROS2ImuMsg* Me
                                                                                                                                 Ctrl+Shift+I
                                                                                                      Format Document
                                                                               UROS2OdomMsg* Mess
                                                                                                      Format Document With..
        Message->SetMsg(currentMsg);
                                                                               Message->SetMsg(cu
                                                                                                      Format Selection
                                                                                                                                Ctrl+K Ctrl+F
                                                                                                                                Ctrl+Shift+R
                                                                           void UIMUSensor::InitP
    void UIMUSensor::InitPublisher() {
        Super::InitPublisher();
                                                                               Super::InitPublish
                                                                                                      Stage Selected Ranges
                                                                                                                             Ctrl+K Ctrl+Alt+S
        AROS2Robot* ownerRobot = static cast<AROS2Robot*>(Ge
                                                                                                                                Ctrl+K Ctrl+R
                                                                                                      Revert Selected Ranges
 mavid ~ > Documents > Workspace > vamex3-vtb > ⅓ dev + 10 ... 1 ▶ 19 git status
                                                                                                                                     Ctrl+X
```

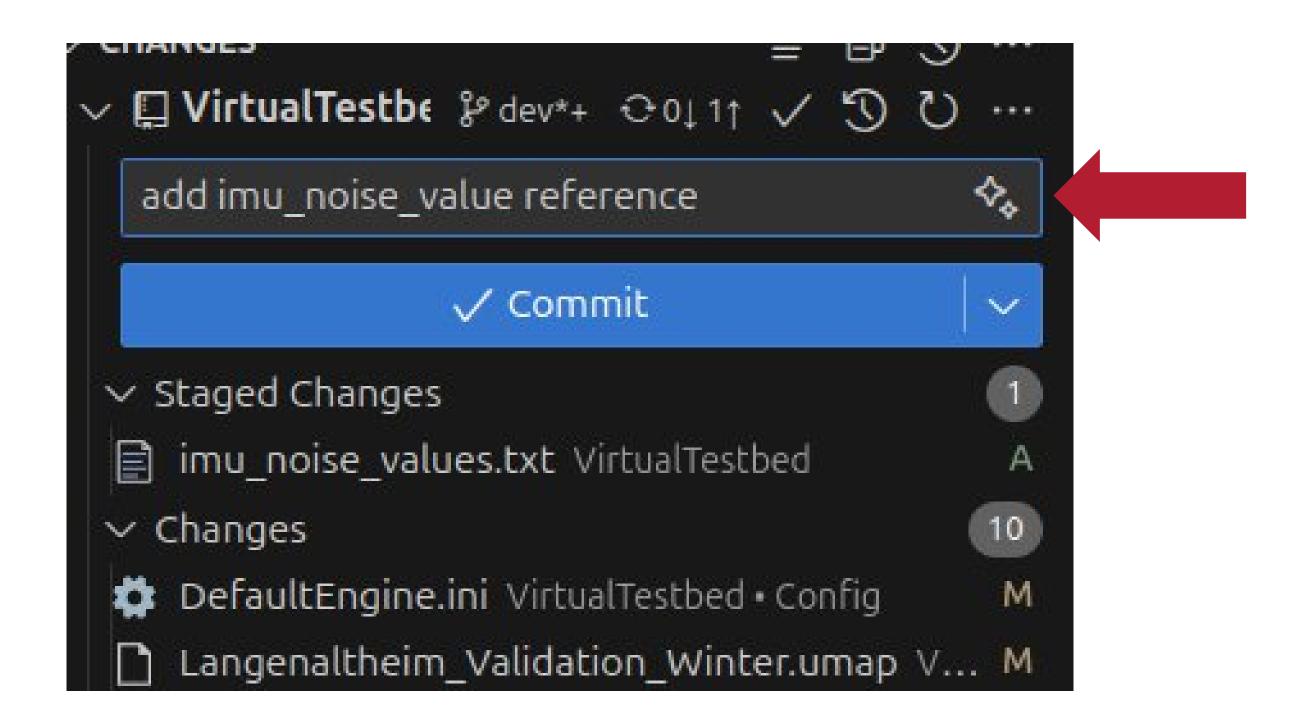
Änderungen zwischen zwei Revisionen derselben Datei im VS Code Diff-Editor. Es ist möglich einzelne Bereiche zu markieren und separat zu stagen.



Änderungen commiten - git commit -m "<message>"



- Gebt eine deskriptive
 Commit-Message für eure Änderungen an.
 - Das ist nicht optional!!!

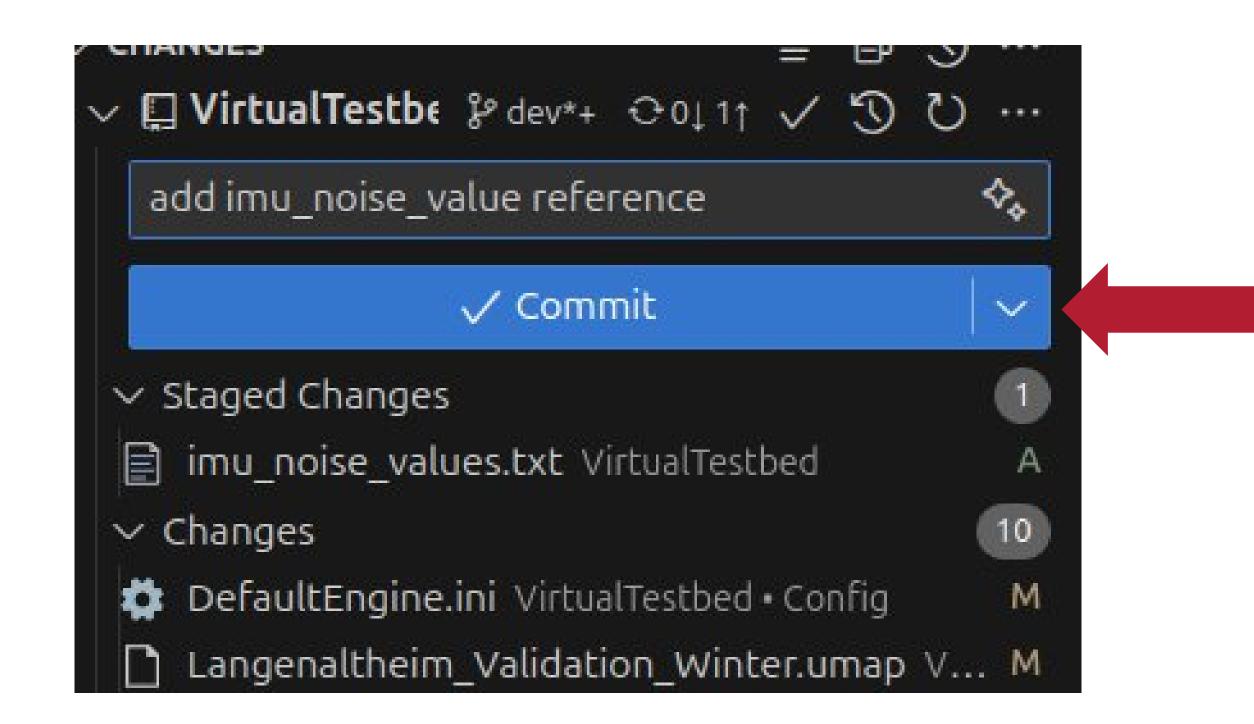




Änderungen commiten - git commit -m "<message>"



- Gebt eine deskriptive
 Commit-Message für eure Änderungen an.
 - Das ist nicht optional!!!
- 2. Mit dem "Commit"-Button geht euer Commit auf die lokale Commit-History.

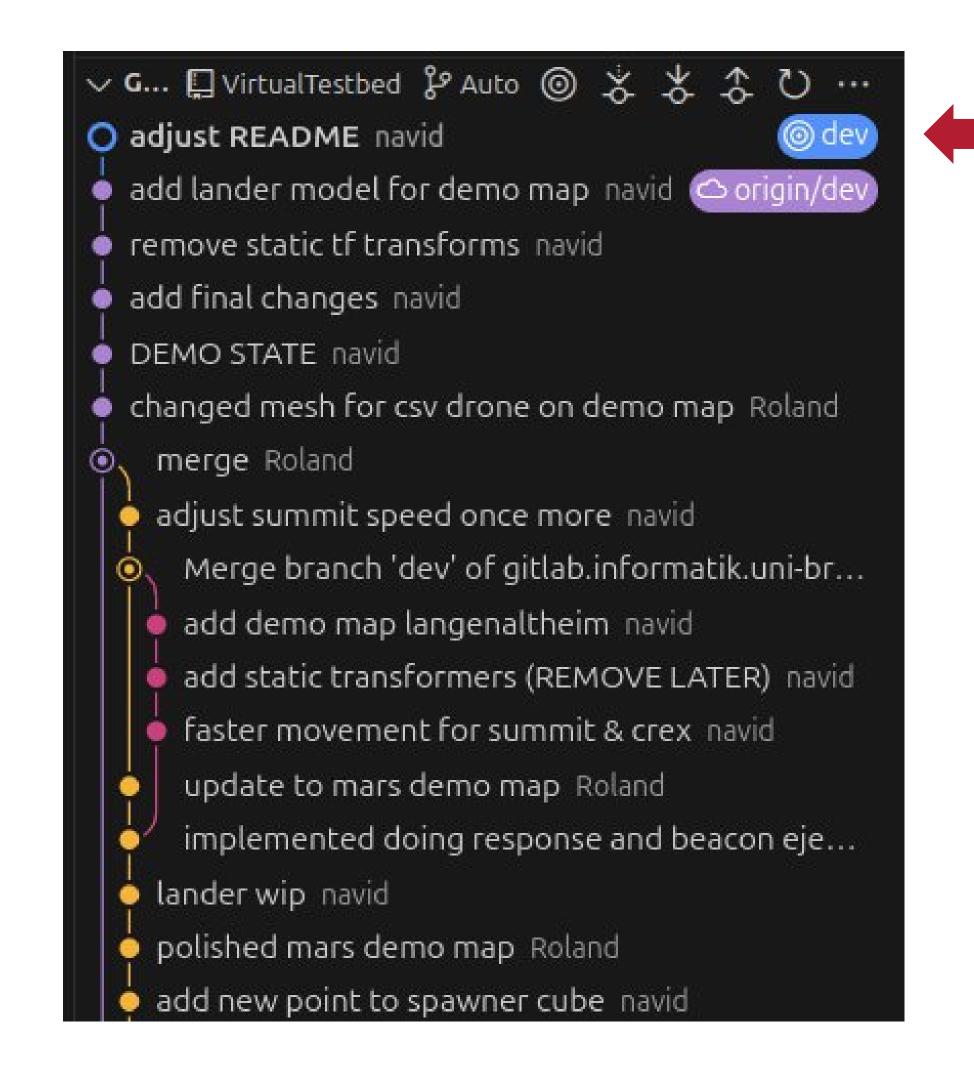




Änderungen commiten - git commit -m "<message>"



- Gebt eine deskriptive
 Commit-Message für eure Änderungen an.
 - Das ist nicht optional!!!
- 2. Mit dem "Commit"-Button geht euer Commit auf die lokale Commit-History.
 - Aber noch nicht aufs Remote Repository (GitLab)! Heißt i.d.R. origin/.

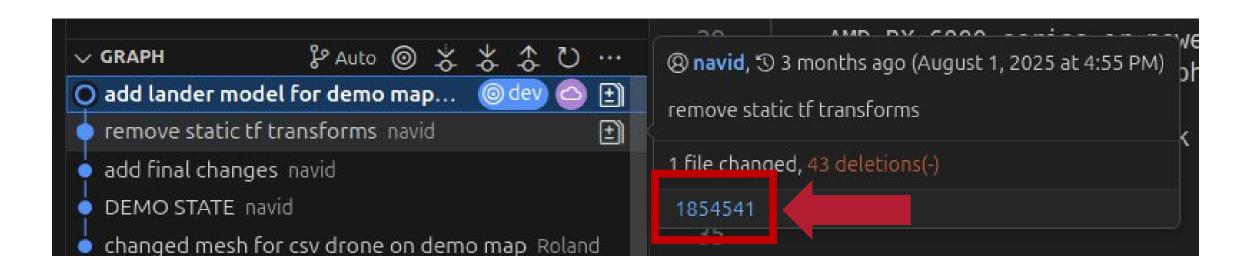




Commit Hashes & History



- Jeder Commit hat einen einzigartigen Hash.
- Wird häufig genutzt, um diese zu referenzieren.
 - I.d.R. reicht der kurze Hash.
- Wird immer in der Commit History mit aufgelistet.
 - git log









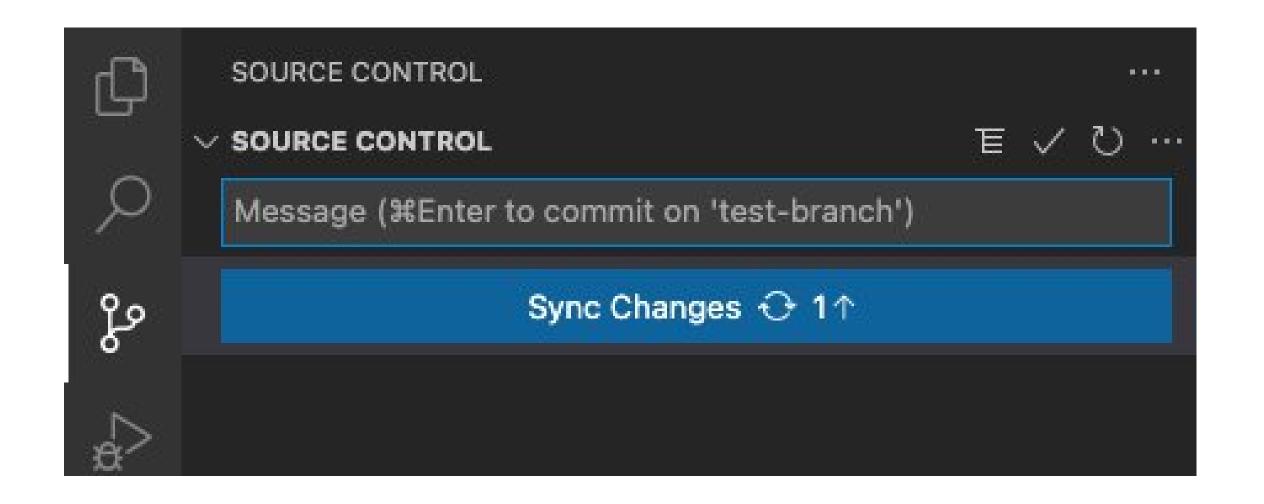
- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.

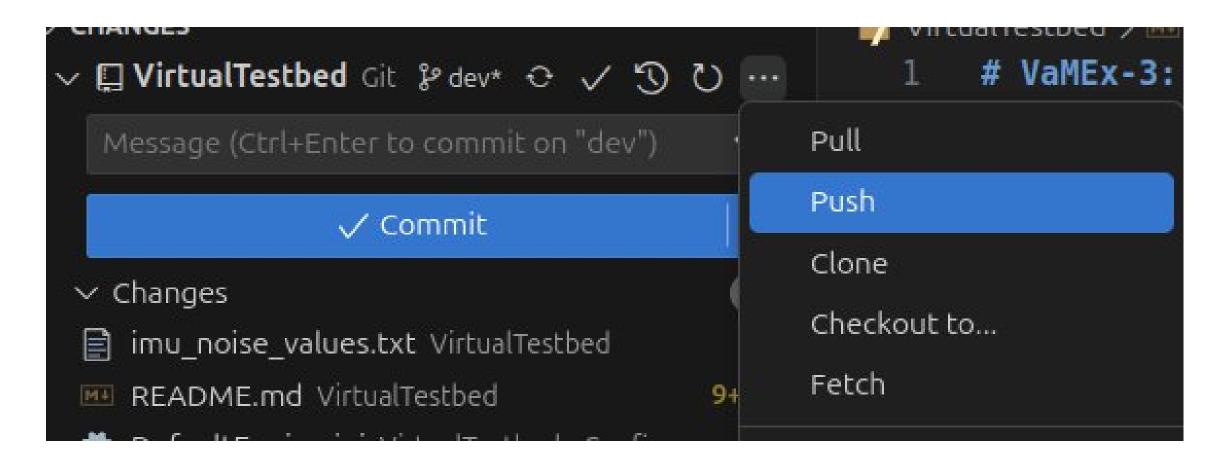


Commits pushen - git push



- 1. Pusht die Änderungen mit eine der zwei Optionen:
 - "Sync Changes"; wird angezeigt, wenn keine lokalen Änderungen mehr zu committen sind
 - dem Source-Control
 Kontextmenü







Commits pushen - git push



- 1. Pusht die Änderungen
- 2. Falls jemand anderes schon vorher gepusht hat, kann es Probleme geben.
 - Die Änderungen mussen vorher lokal integriert werden.
 - Gibt viele Ansätze:
 - Branch & Merge
 - Rebase
 - •





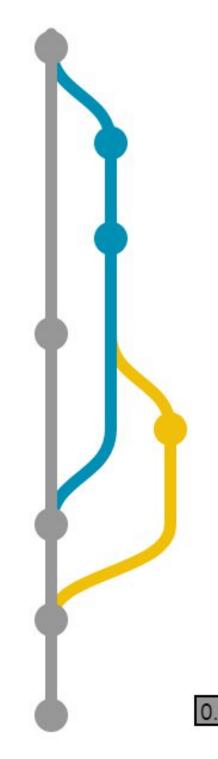
- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.



Branches



- "Abzweigungen" vom aktuellen Stand
 - Warum? Um eigenständig an etwas zu arbeiten, ohne ständig externe Änderungen integrieren zu müssen.
- Rückintegration mit einem "Merge"
 - Erzeugt einen Merge-Commit.
 - Ggf. muss man Anpassungen vornehmen, im Falle eines Merge-Konflikts



[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

[myfeature] eaf618c New cool feature - John Feature

[master] 8f1e0e7 Merge branch 'develop' into 'master' - John Doe

[master] 6a3dacc Merge branch 'myfeature' into 'master' - John Doe

[master] abcdef0 Release of version 0.1 - John Releaser

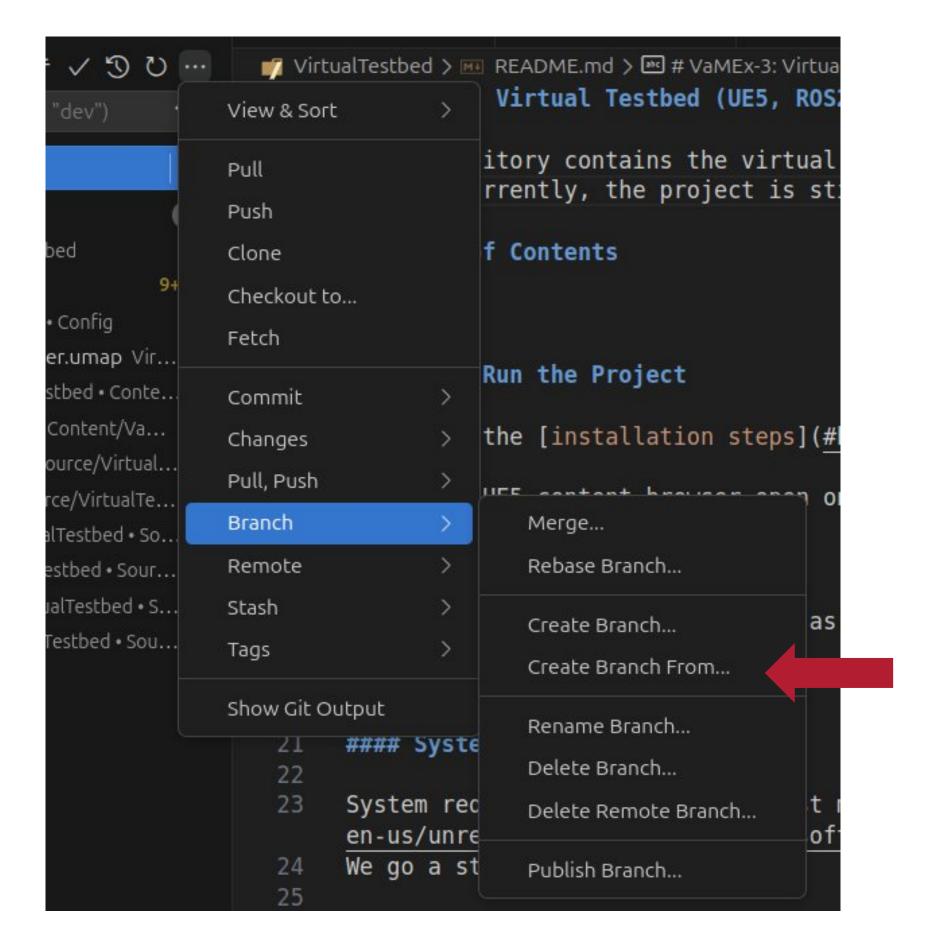
Quelle: GitHub/frappe/charts



Branch erstellen - git branch -c <branch name>



1. Erstellt einen neuen Branch basierend auf main.

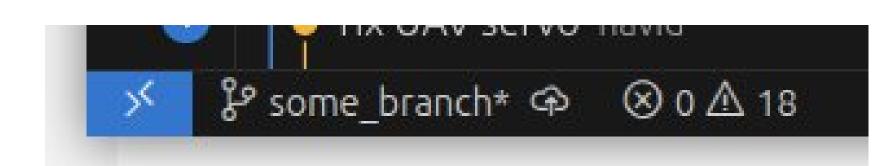


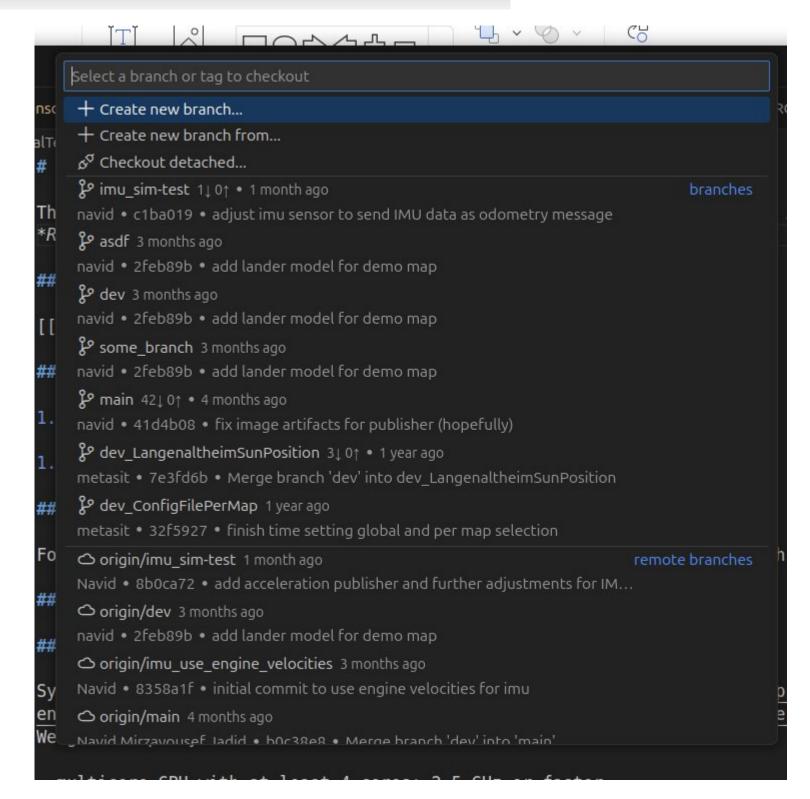


Branch erstellen - git branch -c <branch name>



- 1. Erstellt einen neuen Branch basierend auf main.
- 2. Der Branch kann mit dem Button unten links gewechselt werden.
 - git switch <branch>
 - Man kann darüber auch den neuen Branch erstellen.









- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.





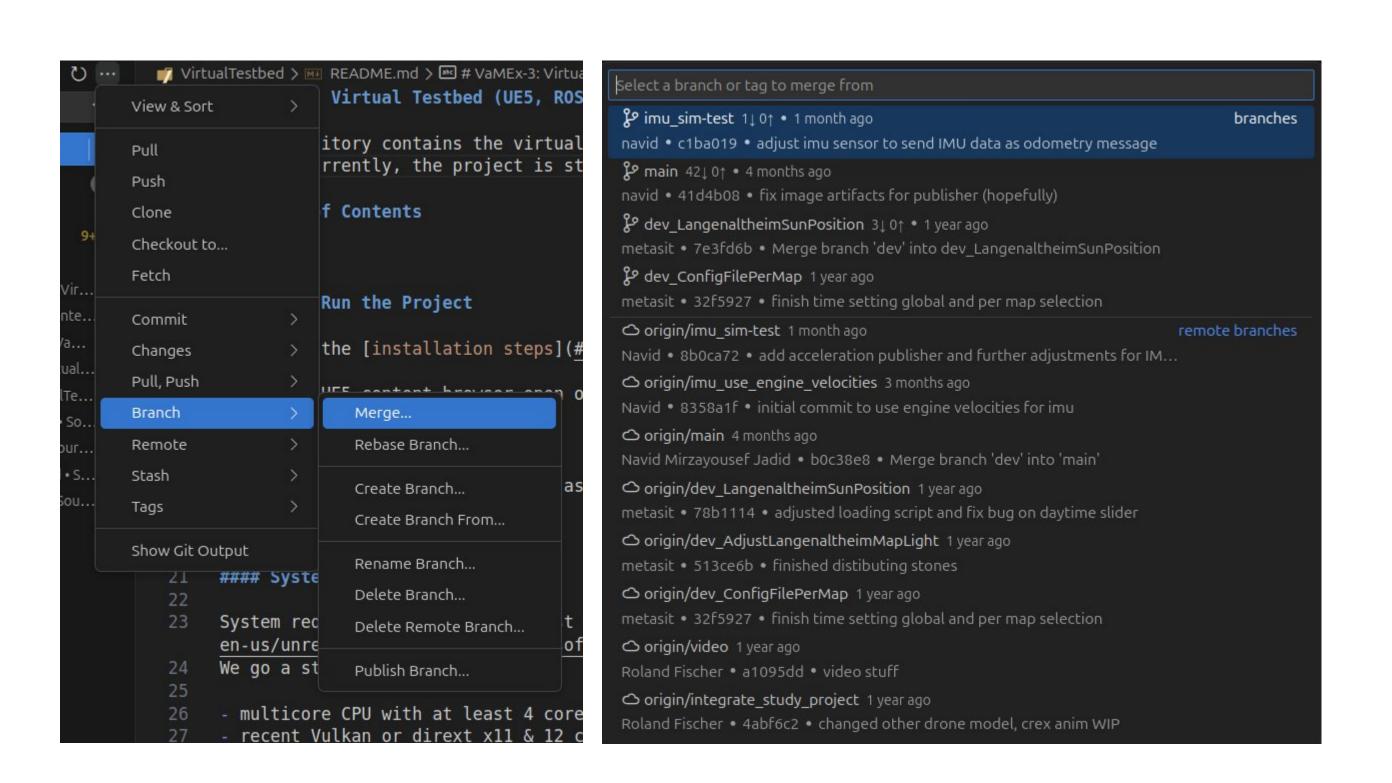
- 1. Repository auf GitLab anlegen.
- 2. Repository klonen (entweder per HTTPS oder SSH).
- 3. Eine Datei (z.B. Markdown; .md) dem lokalen Repo hinzufügen.
- 4. Der Datei etwas Text hinzufügen.
- 5. Änderungen "stagen" und mit einer Nachricht "committen".
- 6. Änderungen "pushen".
- 7. Branch erstellen.
- 8. Änderung an der Datei vornehmen, committen, und pushen.
- 9. Lokalen Branch in main mergen.



Branch mergen - git merge <branch name>



- 1. Wechselt auf main.
- 2. Merged euren neuen Branch in main.
 - Man merged (lokal) immer <u>in</u> den aktuellen Branch rein.





Merge-Konflikte



- Was wenn mehrere Personen an derselben Datei gearbeitet haben und in denselben Branch pushen / mergen?
 - Es entsteht ein "Merge-Konflikt".
 - Jetzt muss entschieden werden, welche Codeänderungen behalten werden, und welche nicht.
 - (Im Zweifelsfall kann man den Merge immer abbrechen.

```
    git merge --abort
```

Quelle: ihatetomatoes.net



Merge-Konflikte

- VS Code hat einen eingebaute Lösungen.
- (Inline Merge-Editor (oben)
 - Current Change: lokaler Stand
 - Incoming Change: einkommender Stand
 - **Both: Kombination**
- Three-Way Merge Editor (unten)
 - Links incoming, rechts current, unten Resultat.
- Optionen können per Click ausgewählt werden.
 Man kann auch direkt reinschreiben.

```
* Prints the welcome message
     Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
     <<<<< HEAD (Current Change)
     function printMessage(showUsage, message) 🕴
         console.log(message);
      Function printMessage(showUsage, showVersion) {
         console.log("Welcome To Line Counter");
         if (showVersion) {
               console.log("Version: 1.0.0");
     >>>>> theirs (Incoming Change)
         if (showUsage) {
              console.log("Usage: node base.js <file1> <file2> ... ");
                                                                      Resolve in Merge Editor
             ♦ You, 20 seconds ago Ln 11, Col 26 Spaces: 4 UTF-8 CRLF {} JavaScript 🔠 🔊
 rget.js ! 🎐 💹 Js Merging: target.js ! 🔘
                                                                      erge-git-playground > Js target.js > 🕅 printMessage
                                              Current 0 b7bd9b1 - main
                                                      * Prints the welcome message
     * Prints the welcome message
                                                        ction printMessage(showUsage, message) {
                                                         console.log(message);
           console.log("Version: 1.0.0");
                                                             console.log("Usage: node base.js <file</pre>
        ction printMessage(showUsage)
        console.log("Welcome To Line Counter");
       if (showUsage) {
           console.log("Usage: node base.js <file1> <file2> ... ");
```

Quelle: Microsoft

Ln 17, Col 31 Spaces: 4 CRLF {} JavaScript 🔠 🔊 🚨

ain! ♠ ઋ ⊗ 0 ⚠ 0 Not Logged In



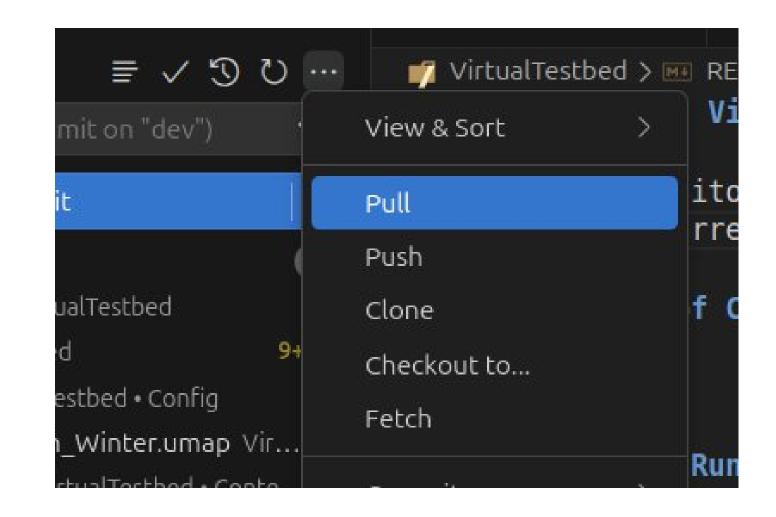
Änderungen / Commits ziehen - git pull



35

• git pull: Ziehe Änderungen vom Remote Repository

- Kann Probleme verursachen, wenn diesselbe Datei sowohl lokal als auch remote bearbeitet wurde.
 - Wenn die Änderungen lokal schon committed wurden, gibt es einen Merge-Konflikt.
 - Falls noch nicht, kann man die lokalen Änderungen erstmal "stashen" und danach wieder integrieren.

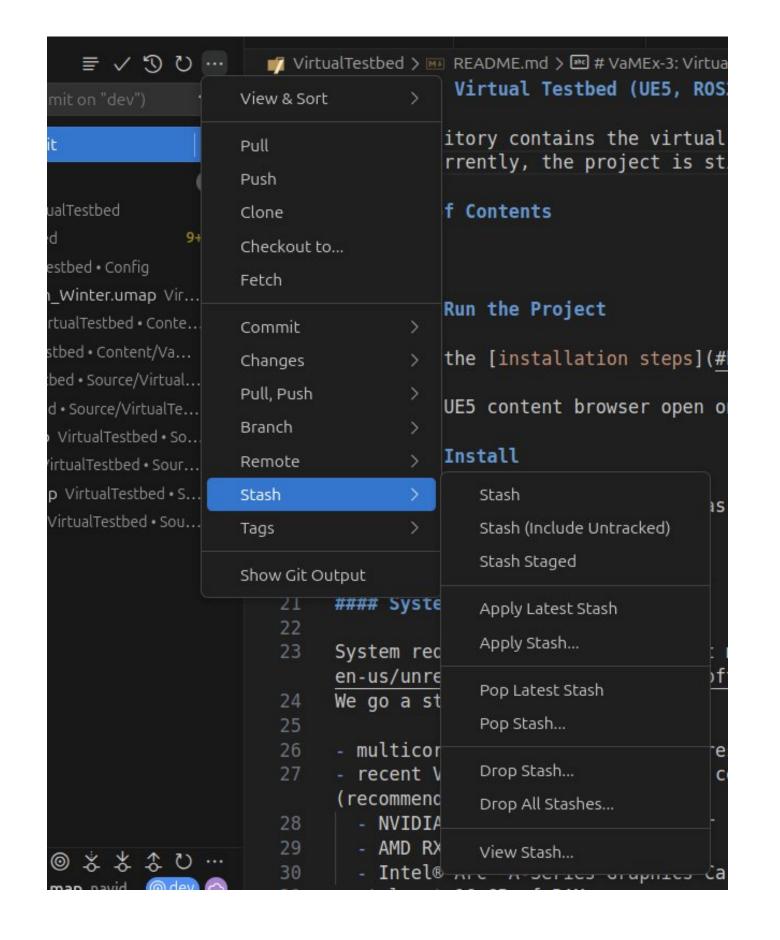




Stashes



- Manchmal möchte man lokale Änderungen kurzzeitig "verstauen" bzw. "wegstecken" (—> "stashen").
- git stash : Stecke die lokalen Änderungen in einen Stash.
 - Achtung: Untracked Files werden nicht standardmäßig mitgestasht.
- git stash list: Liste die vorhandenen Stashes auf.

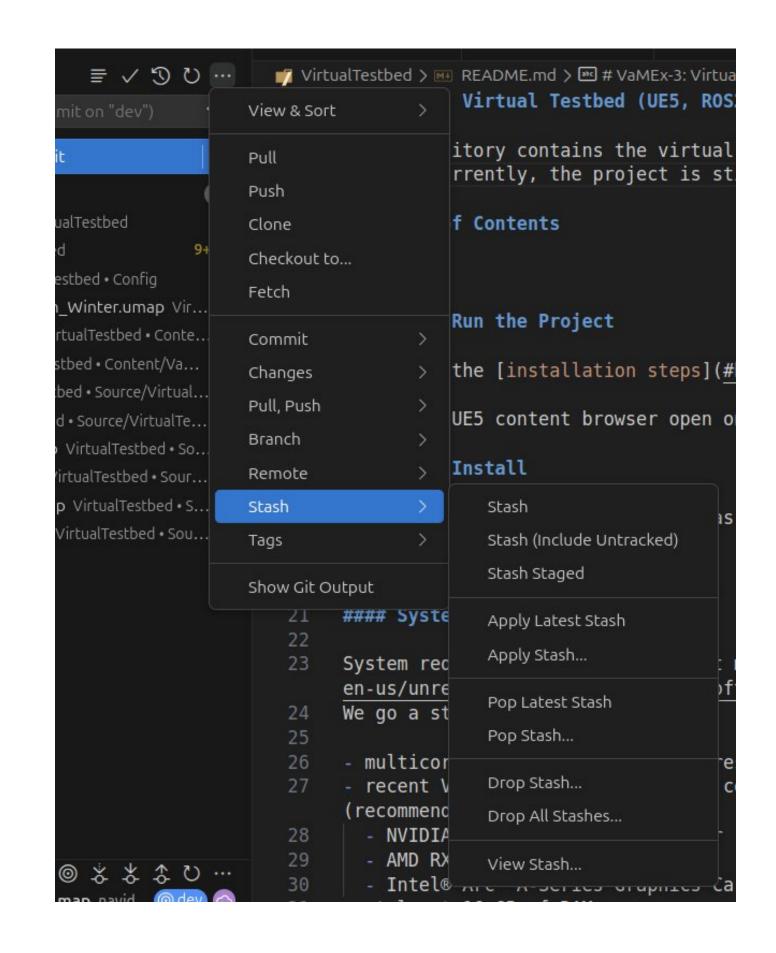




Stashes



- Entferne den letzten Eintrag aus der Liste der Stashes und wende die Änderungen an.
 Optional kann mit [stash number] ein früherer Stash ausgewählt werden.
- git stash apply [stash number] :
 Wie stash pop , behält aber den
 Eintrag.
- Gibt es bei pop oder apply einen Merge-Konflikt, wird der Eintrag immer behalten.





Weitere Befehle



- git checkout <commit hash> : Wechsel zum Stand des spezifizierten Commits auf dem aktuellen Branch.
- git checkout : Wechsel zum aktuellesten Stand des aktuellen Branches.
- git checkout <branch name> : Wechsel zum spezifizierten Branch.
- qit checkout -b <branch name> : Erstelle einen neuen Branch und wechsel zu diesem.
- git revert <commit hash>: Kehre die Änderungen aus dem gelisteten Commit zurück. Erstellt einen Revert-Commit, behält den originalen Commit aber in der History.



Einige Worte zum Schluss



- Git / Version Control ist sehr m\u00e4chtig und gleichzeitig unverzichtbar in Softwareentwicklung!
- Dies bleibt nur eine Einführung! Ihr lernt noch viel in der Eigenarbeit.
 - Für die meisten Sachen gibt es online Tutorials oder Foreneinträge.
 - Chat Bots können auch helfen.
 - Seid in beiden Fällen vorsichtig! Falsche Entscheidungen können euch Stunden an Arbeit kosten.



Zusätzliche Resourcen



- Git Explained in 100 Seconds (Youtube)
- Learn Git in 15 Minutes (<u>Youtube</u>)
- A Grip On Git (https://agripongit.vincenttunru.com/)
 - kurze visuelle Einführung in Git
 - < ~11 min read</pre>
- Learn Git Branching (https://learngitbranching.js.org/)
 - interaktives Tutorial für git im Webbrowser