

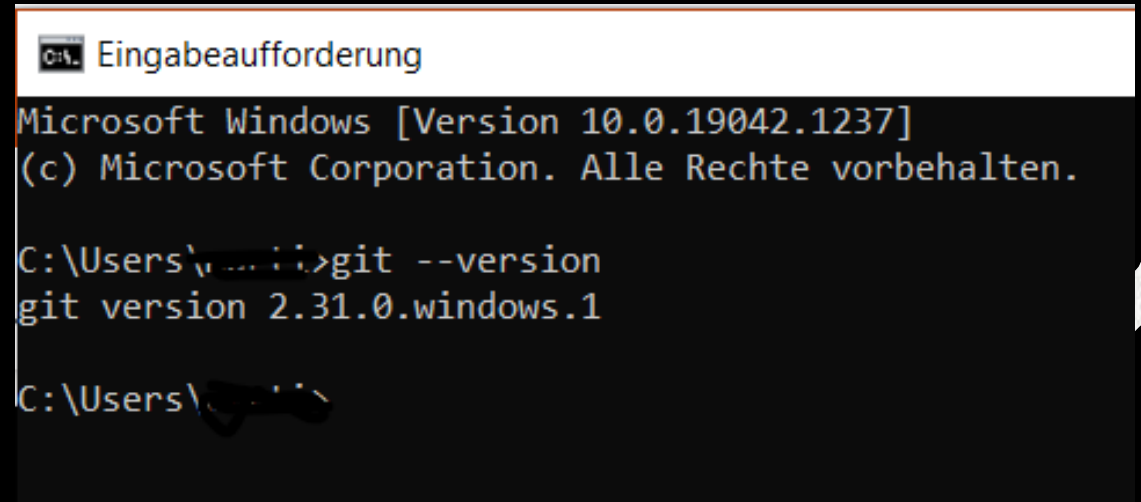
Installation

<https://git-scm.com/downloads>

(Linux: `sudo apt install git-all`)

Beim Installer: Standardeinstellungen OK

Test: `git --version`



```
Git - Eingabeaufforderung
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\w...>git --version
git version 2.31.0.windows.1

C:\Users\w...>
```

Einstellungen

Änderungen manuell im Editor:
fehlerträchtig!

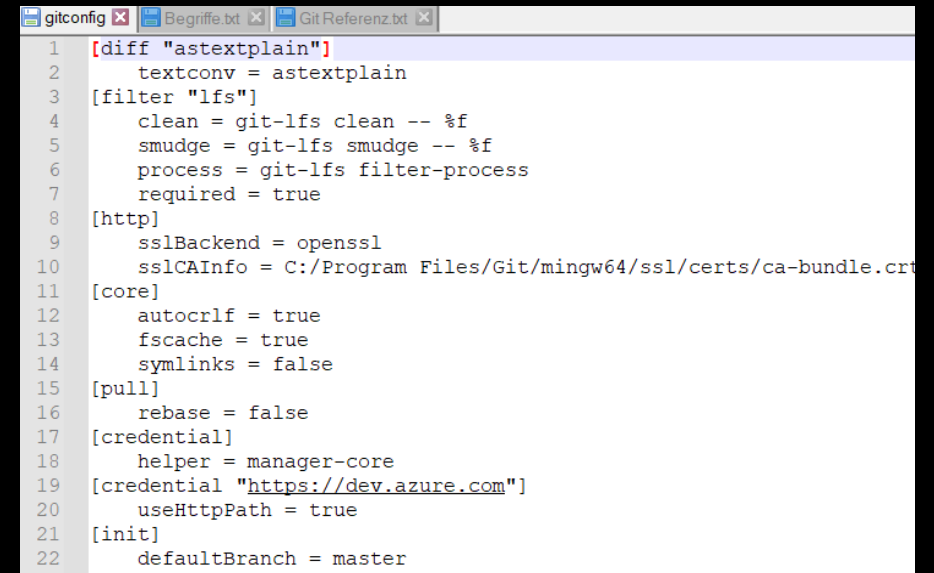
-> Git-Befehle nutzen:

- `git config --global user.name "My Name"`
- `git config --global user.email me@bsp.at`

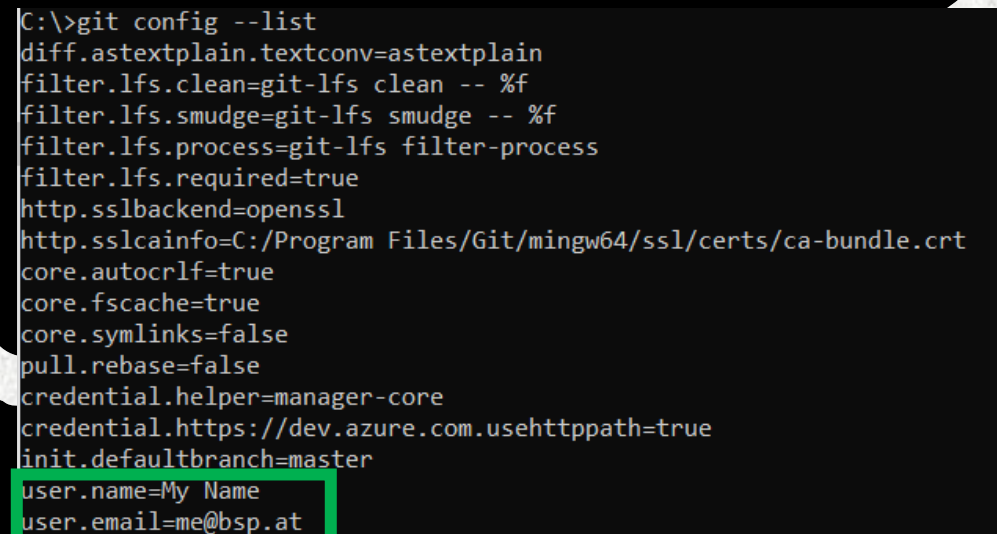
Prüfen:

- `git config --list`

C:\Programs\Git\etc -> gitconfig Datei:



```
1 [diff "astextplain"]
2   textconv = astextplain
3 [filter "lfs"]
4   clean = git-lfs clean -- %f
5   smudge = git-lfs smudge -- %f
6   process = git-lfs filter-process
7   required = true
8 [http]
9   sslBackend = openssl
10  sslCAInfo = C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
11 [core]
12  autocrlf = true
13  fscache = true
14  symlinks = false
15 [pull]
16  rebase = false
17 [credential]
18  helper = manager-core
19 [credential "https://dev.azure.com"]
20  useHttpPath = true
21 [init]
22  defaultBranch = master
```



```
C:\>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=My Name
user.email=me@bsp.at
```

Git Hilfe

Auflistung Befehle:

HTML-Offline-Hilfe zu Befehl zB.
"config":

Gekürzte Darstellung in cmd

```
git help
```

```
git help config      oder  
git config --help
```

```
git config -h        oder  
git config -help
```



<https://pixabay.com/de/photos/fragen-unterzeichnen-design-kreativ-2341784/>

ODER:
Googlen ;-)

Kommandozeileninterpreter

Eingabeaufforderung, CLI, Konsole, cmd, Powershell, Shell,
Unix-Shell, Bash, Terminal (macOS, Linux), **Git Bash/Git CMD**

Win: cmd.exe vs. Powershell

```
cmd: Eingabeaufforderung
C:\Users\...>cd source
C:\Users\...\source>mkdir test
C:\Users\...\source>dir
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: 28B3-FAE3

Verzeichnis von C:\Users\...\source

08.10.2021  14:18    <DIR>          .
08.10.2021  14:18    <DIR>          ..
08.10.2021  12:06    <DIR>          cprsw
26.08.2021  11:46    <DIR>          htl_leonding
02.09.2021  19:35    <DIR>          repos
08.10.2021  14:18    <DIR>          test
               0 Datei(en),           0 Bytes
               6 Verzeichnis(se), 346 418 024 448

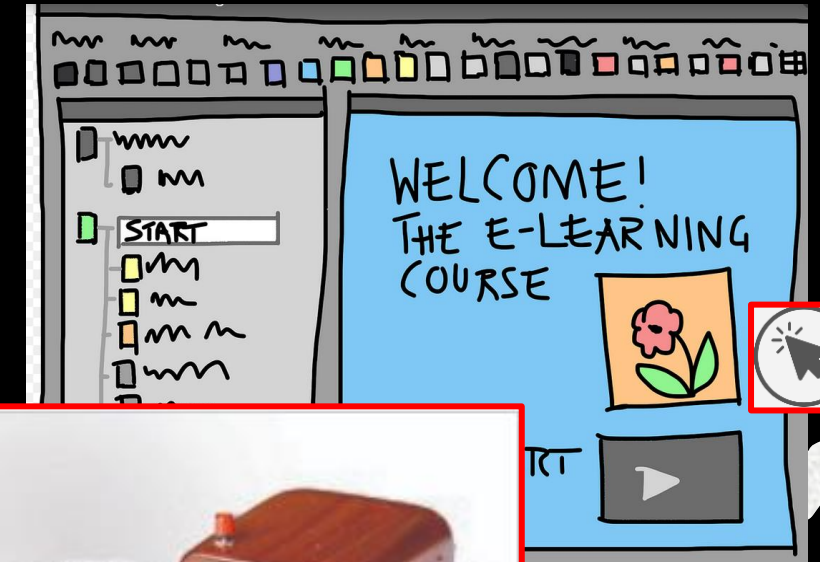
Windows PowerShell
PS C:\Users\...\source> cd .\source\
PS C:\Users\...\source> dir

Verzeichnis: C:\Users\...\source

Mode                LastWriteTime         Length Name
----                -
d-----          08.10.2021    12:06         cprsw
d-----          26.08.2021    11:46       htl_leonding
d-----          02.09.2021    19:35         repos
d-----          08.10.2021    14:18         test

PS C:\Users\...\source>
```

GUI (Graphical User Interface)



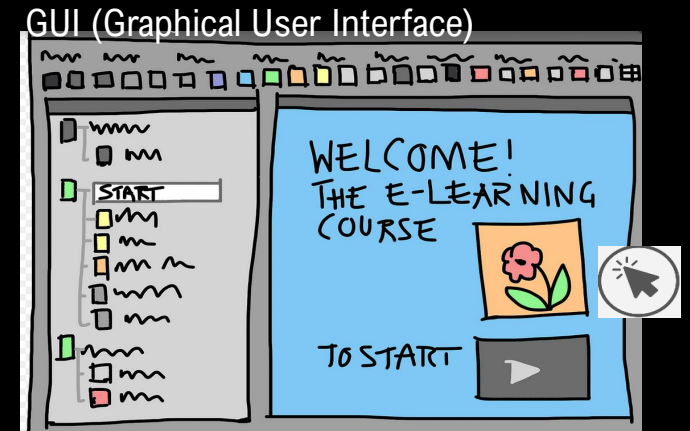
Der erste Prototyp einer Computermouse wurde 1963 von William English nach Zeichnungen von Douglas Engelbart gebaut.

[https://de.wikipedia.org/wiki/Maus_\(Computer\)](https://de.wikipedia.org/wiki/Maus_(Computer))

"X-Y-Positions-Anzeiger für ein Bildschirmsystem"

Kommandozeileninterpreter

- Bei jedem Betriebssystem vorinstalliert
- Schriftliche Eingabe der Befehle
- Funktionen für Steuerung des Betriebssystems
- GUIs beschränken sich oft auf wesentliche Funktionen
- GUI anfangs intuitiver, jedoch Einarbeitung in jede neue GUI notwendig (bei CLI nicht der Fall)



Win: cmd.exe vs. Powershell

```
cmd: Eingabeaufforderung

C:\Users\...>cd source
C:\Users\...\.source>mkdir test
C:\Users\...\.source>dir
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: 28B3-FAE3

Verzeichnis von C:\Users\...\.source

08.10.2021  14:18  <DIR>      .
08.10.2021  14:18  <DIR>      ..
08.10.2021  12:06  <DIR>      cprsw
26.08.2021  11:46  <DIR>      htl_leonding
02.09.2021  19:35  <DIR>      repos
08.10.2021  14:18  <DIR>      test
               0 Datei(en),           0 Bytes
               6 Verzeichnis(se), 346 418 024 448 Bytes frei

C:\Users\...\.source>
```

```
Windows PowerShell

PS C:\Users\...> cd .\source\
PS C:\Users\...\.source> dir

Verzeichnis: C:\Users\...\.source

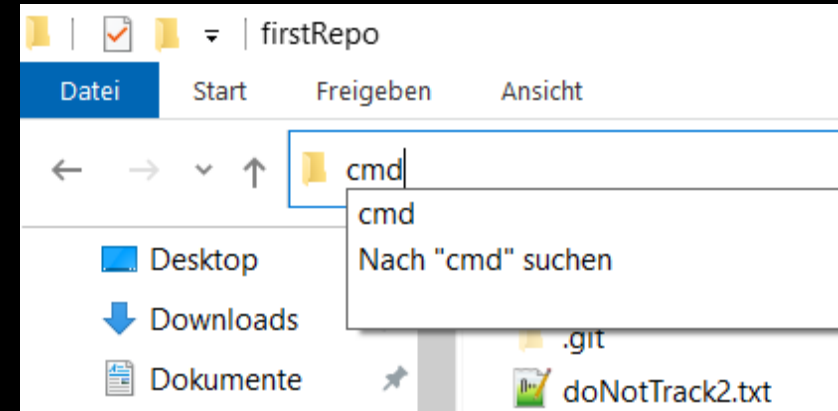
Mode                LastWriteTime         Length Name
----                -
d-----          08.10.2021   12:06             cprsw
d-----          26.08.2021   11:46             htl_leonding
d-----          02.09.2021   19:35             repos
d-----          08.10.2021   14:18             test

PS C:\Users\...\.source>
```

Eingabeaufforderung, CLI, Konsole, cmd, Powershell, Shell, Unix-Shell, Bash, Terminal (macOS, Linux), **Git Bash/Git CMD**

CMD <-> Explorer - im selben VZ öffnen

- Im Explorer in Verzeichnis in Adressleiste "cmd" (oder "powershell" etc.) eingeben um diese im selben Verzeichnis zu öffnen
- In CMD (oder Powershell) "explorer ." (mit Punkt!) eingeben um CLI im aktuellen Verzeichnis zu öffnen



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\...\source\cprsw\firstRepo>explorer .

C:\Users\...\source\cprsw\firstRepo>
```

Kommandozeileninterpreter

Befehle nutzen

Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

1. Suche mittels Eingabe von "help" den Befehl für "Löscht den Bildschirminhalt" und führe ihn aus.
2. Was bedeutet cd – Suche mittels "help cd"
3. Navigiere mittels "cd .." bis zu deinem root-Verzeichnis/Laufwerksbuchstaben
4. Lasse dir den Inhalt des Ordners auflisten mittels "dir"
5. Navigiere mittels "cd ordnerXY" 4x in die Tiefe (Info mittels "dir")
6. Navigiere mittels "cd\" wieder zu deinem root
7. Betätige die Pfeil-auf/-ab-Tasten – was geschieht hier?

Exkurs: Command Line – Befehle:

Command	Beschreibung
cd cd.. cd\	Verzeichnis wechseln
mkdir	Ordner erstellen
dir (oder Linux: ls)	Liste Ordnerinhalt
dir /adh (oder Linux: ls -a)	Versteckte Ordner anzeigen lassen

Git Repository

- Der "Ablageort" des Projekts

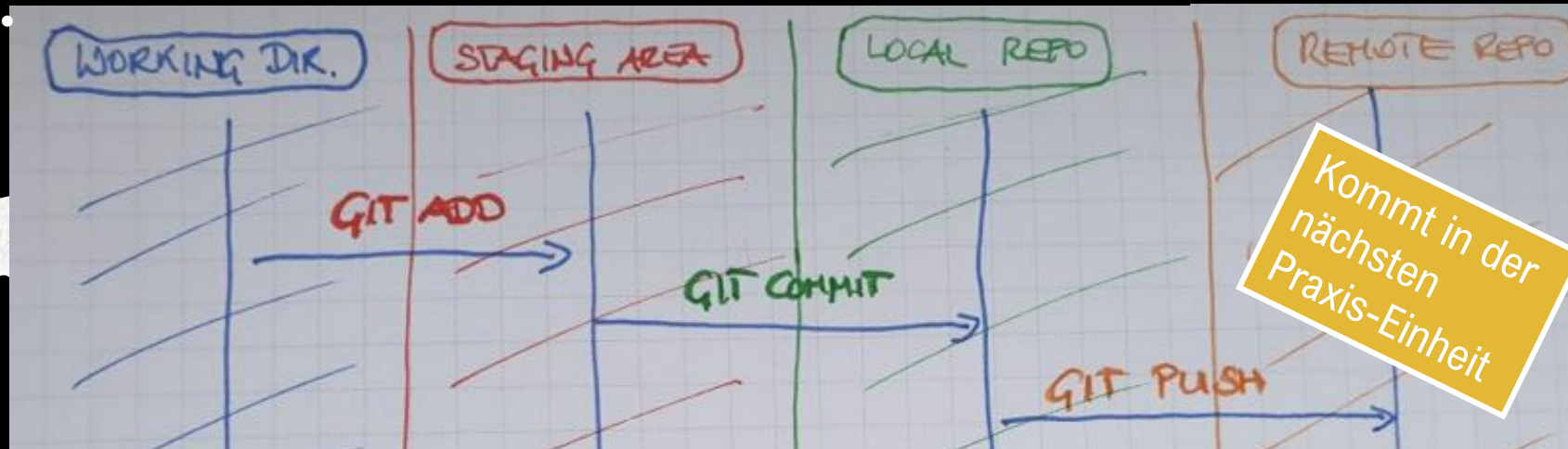
- Aktuelle Dateien des Projekts
- Alle früheren Versionen
- Daten, um Änderungen nachzuverfolgen

Möglichkeit 1: LOKAL

- Nutze Repo nur für mich selbst
- Start eines neuen Projekts (erst später Kollaboration über Remote Repo)

Möglichkeit 2: VERTEILT

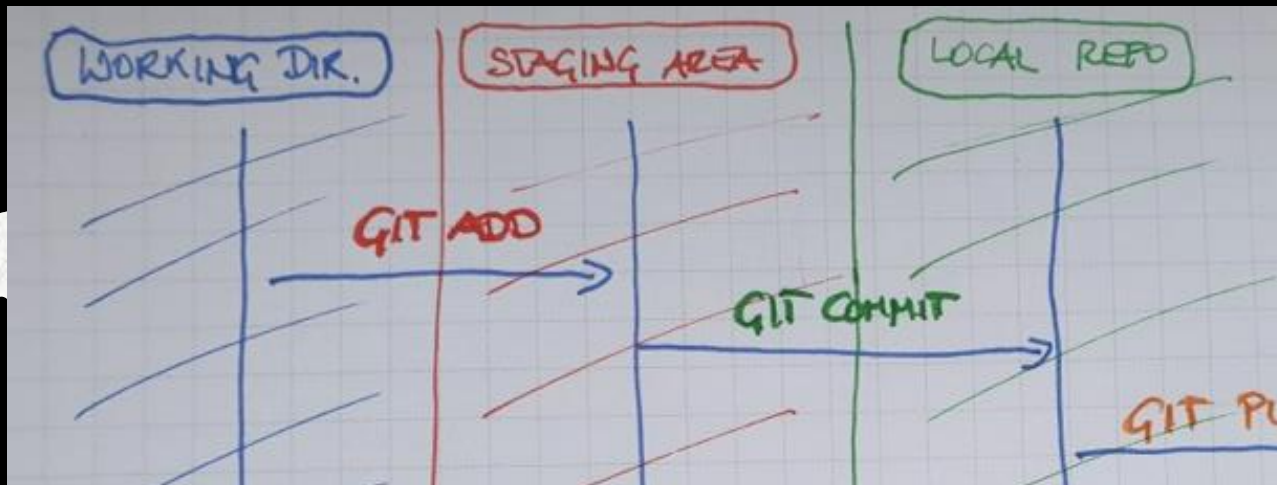
- an bestehendem Repo teilnehmen (klonen)
- Gemeinsames Projekt



Git Repository (lokal)

- Die 3 Zustände

- **Modified:** Datei geändert, aber noch nicht in lokale Datenbank eingecheckt
- **Staged:** eine geänderte Datei für nächsten Commit vorgemerkt
- **Committed:** Daten sicher in der lokalen Datenbank (Git-Verzeichnis) gespeichert



Lokales Repo erzeugen

Leeres Repo erzeugen: `git init`

-> *Erstellt ein leeres Repo im aktuellen Ordner*

Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

1. Erstelle einen lokalen Ordner für CPRSW
2. Erstelle darin einen Ordner "firstRepo" für dein erstes Repo
3. Navigiere zum Ordner und Erzeuge ein leeres Repo
4. Lasse dir den Inhalt des Repos anzeigen (Versteckte Dateien!) -> der versteckte **.git**-Ordner wird angezeigt

Exkurs: Command Line – Befehle:

Command	Beschreibung
<code>cd</code> <code>cd..</code> <code>cd\</code>	Verzeichnis wechseln
<code>mkdir</code>	Ordner erstellen
<code>dir</code> (oder Linux: <code>ls</code>)	Liste Ordnerinhalt
<code>dir /adh</code> (oder Linux: <code>ls -a</code>)	Versteckte Ordner anzeigen lassen

```
C:\Users\... \source\cprsw\firstTest>dir /adh
Datenträger in Laufwerk C: ist OS
Volumeserienummer: 28B3-FAE3

Verzeichnis von C:\Users\... \source\cprsw\firstTest

22.09.2021  22:42    <DIR>          .git
               0 Datei(en),               0 Bytes
               1 Verzeichnis(se), 346 609 205 248 Bytes frei
```

Der .git Ordner

- Wird bei `git init` erzeugt

Enthält alle Informationen, die zur **Verwaltung** des Repos nötig sind

- Commit-Verlauf/Logs
- Konfigurationsdatei (lokal)
- aktivierbare Beispielskripte für Zusatzfunktionen (hooks))

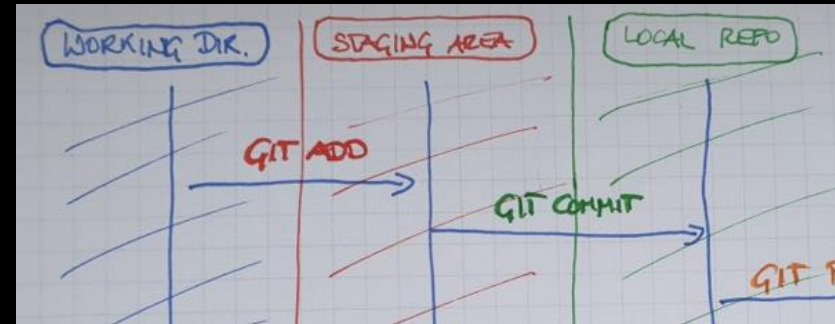
```
C:\Users\... \source\cprsw\firstTest>dir /adh
Datenträger in Laufwerk C: ist OS
Volumeserienummer: 28B3-FAE3

Verzeichnis von C:\Users\... \source\cprsw\firstTest

22.09.2021  22:42    <DIR>          .git
                0 Datei(en),                0 Bytes
                1 Verzeichnis(se), 346 609 205 248 Bytes frei
```

```
08.10.2021  19:20          31 COMMIT_EDITMSG
08.10.2021  16:19        130 config
08.10.2021  16:19         73 description
08.10.2021  16:19         23 HEAD
08.10.2021  16:19    <DIR>        hooks
08.10.2021  19:20        145 index
08.10.2021  16:19    <DIR>        info
08.10.2021  18:31    <DIR>        logs
08.10.2021  19:20    <DIR>        objects
08.10.2021  16:19    <DIR>        refs
                5 Datei(en),           402 Bytes
                5 Verzeichnis(se), 339 866 800 128 Bytes frei
```

Status prüfen



Status abfragen: `git status`

-> fragt den Status des Repos ab, in dessen Verzeichnis wir uns befinden

Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

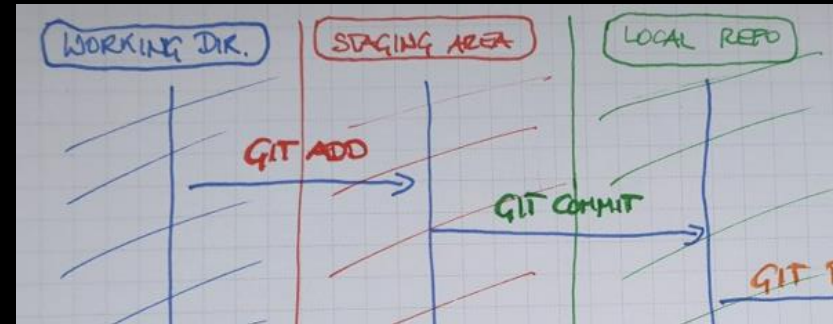
1. Navigiere zum Ordner deines leeren Repos "firstRepo"
2. Frage den Status ab: Repo ist leer -> "No commits yet"

```
C:\Users\...source\cprsw\firstTest>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Inhalt adden



Dateien hinzufügen: `git add file.txt`
`git add --all`
oder `git add *.java`

-> fügt einzelne/alle Dateien/alle .java-Dateien in den Staging-Bereich hinzu

Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

1. Navigiere zu deinem (leeren) Repo "firstRepo"
2. Füge eine Datei "firstFile.txt" hinzu
 1. per Datei-Explorer ODER
 2. Mittels cmd "copy con firstFile.txt" -> Enter
-> optionaler Texteingabe -> Cmd+Z (quit)
3. Frage den Status ab
4. Adde die Datei
5. Frage den Status ab

```
C:\Users\...1\source\cprsw\firstRepo>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    firstFile.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
C:\Users\...1\source\cprsw\firstRepo>git add firstFile.txt

C:\Users\...1\source\cprsw\firstRepo>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   firstFile.txt
```


Inhalt committen

Dateien committen:

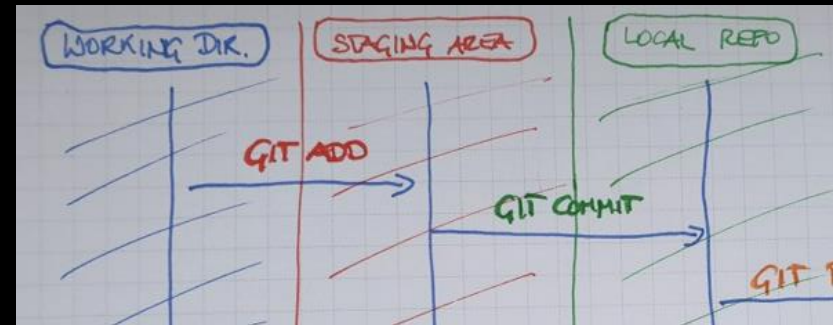
```
git commit -m "First file added"
```

-> alle mittels "add" hinzugefügten Inhalte werden committed (=in den Versionsverlauf aufgenommen)

Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

1. Navigiere zu deinem Repo "firstRepo" mit der neu geaddeten Datei
2. Führe eine commit durch (inkl. Message)
3. Führe eine Status-Abfrage durch



E-D
commit: überlassen, übergeben

```
C:\Users\... \source\cprsw\firstRepo>git commit -m "First file added"
[master (root-commit) 78a25a5] First file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 firstFile.txt

C:\Users\... \source\cprsw\firstRepo>git status
On branch master
nothing to commit, working tree clean
```


Inhalt committen

Dateien committen (cli Text-Editor)

`git commit`

-> Message-Eingabe via Text-Editor (meist zu umständlich/nicht intuitiv – Aneignen der Befehle notwendig)

```
Eingabeaufforderung - git commit

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   test1.txt
#
C:/Users/marti/source/cprsw/tempRepo/.git/COMMIT_EDITMSG [unix] (20:17 25/09/2021)
```

Bspw.: VIM
quit: Eingabe von ":q"

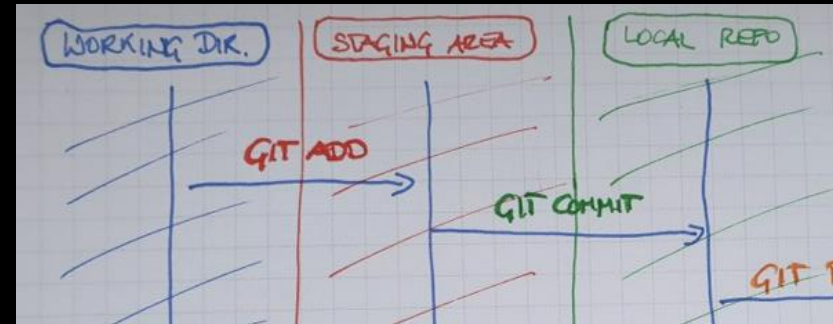
Inhalt verändern

Dateien adden und committen:

1. `git add firstFile.txt`
2. `git commit -m "Zeile hinzugefügt"`

ODER

`git commit -a -m "Zeile hinzugefügt"`



Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

1. Füge Inhalt in die Datei "firstFile.txt" Zeilen ein
2. Führe eine Status-Abfrage durch
3. Adde die Datei
4. Erneute Status-Abfrage
5. Committe die Datei

```
C:\Users\... \source\cprsw\firstRepo>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   firstFile.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
C:\Users\... \source\cprsw\firstRepo>git add firstFile.txt

C:\Users\... \source\cprsw\firstRepo>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   firstFile.txt

C:\Users\... \source\cprsw\firstRepo>git commit -m "Zeile hinzugefügt"
[master 4952e6d] Zeile hinzugefügt
1 file changed, 1 insertion(+)
```

Commit-Historie

```
git log
commit 4ce9ccbdac7edce67d98bba37cf8e50375bbcc80
Author: My Name <me@bsp.at>
Date:   Fri Oct 8 20:20:18 2021 +0200

    removed from index doNotTrack2

commit 86cf9d4d831a932ca121537ffcc1c16599f474aa
Author: My Name <me@bsp.at>
Date:   Fri Oct 8 20:18:28 2021 +0200

    doNotTrack2 added
```

(Modus beenden: "q")

ODER

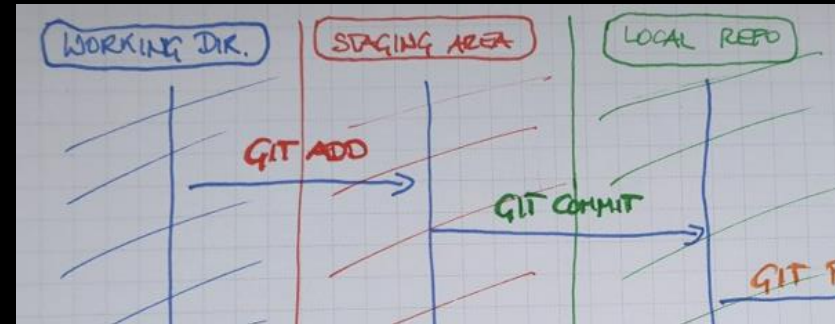
```
git log --pretty=oneline
```

```
C:\Users\m...i\source\cprsw\firstRepo>git log --pretty=oneline
29460d3f6b9006eebcee71fe569dc48118592366 (HEAD -> master) Zeile hinzugefügt.
7d1f49e7235842cc926761fe6164120816f96b5a First Commit
```

Dateien löschen

Vollständig (auch von der Festplatte):

```
1. git rm -f delFile.txt
```



Aufgaben:

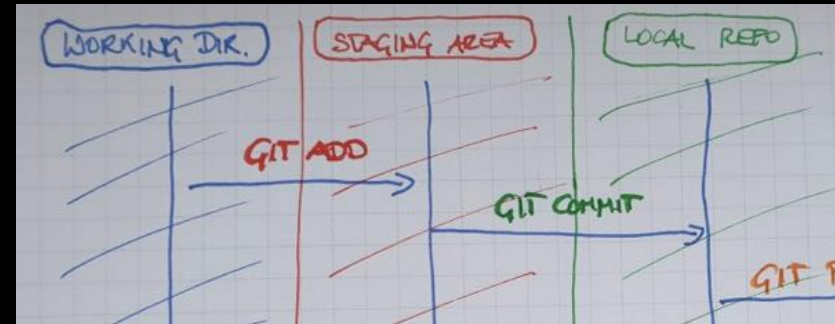
Nutze die Eingabeaufforderung (=Command line)

1. Erstelle eine neue Datei "delFile.txt" und "delFile2.txt"
2. Führe eine Status-Abfrage durch
3. Adde die Dateien
4. Lösche Datei "delFile.txt" vollständig (Probiere den Befehl zuerst ohne "-f")
5. Führe eine Status-Abfrage durch
6. Committe ("delFile2.txt" -> local Repo)
7. Lösche "delFile2.txt" vollständig
8. Erneute Status-Abfrage
9. Committe die "Löschung"

Dateien löschen

Nur aus der Versionsverwaltung:

```
1. git rm --cached doNotTrack.txt
```

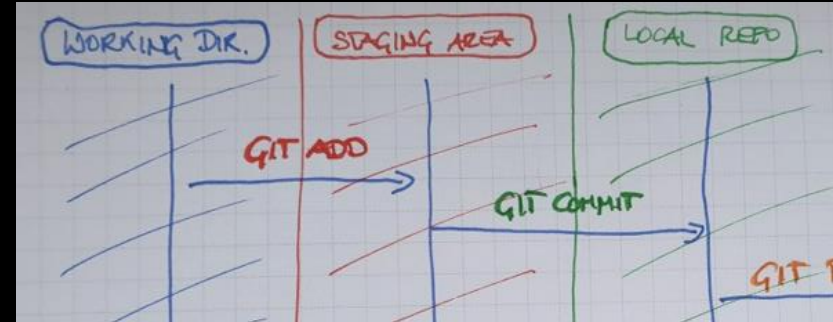


Aufgaben:

Nutze die Eingabeaufforderung (=Command line)

1. Erstelle eine neue Datei "doNotTrack.txt" und "doNotTrack2.txt"
2. Führe eine Status-Abfrage durch
3. Adde die Dateien
4. "Untracke" Datei "doNotTrack.txt"
5. Führe eine Status-Abfrage durch
6. Committe ("doNotTrack2.txt" -> local Repo)
7. "Untracke" doNotTrack2.txt"
8. Erneute Status-Abfrage
9. Committe das "Untracken"

Dateien umbenennen



- Über Git durchführen sobald in Staging Area oder Repo! Andernfalls wird git die Datei nicht mehr zuordnen können!

1. `git mv doNotTrack.txt doTrack.txt`

Aufgaben:

Nutze die Eingabeaufforderung (=Command line)

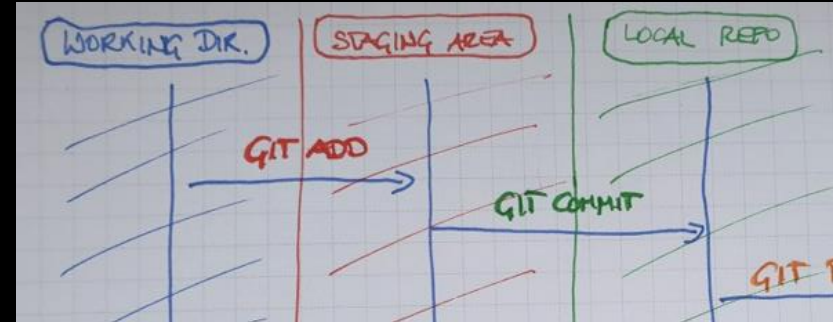
1. Adde und committe die Datei "doNotTrack.txt"
2. Benenne die Datei um in "doTrack.txt"
3. Führe eine Status-Abfrage durch
4. Committe die Änderung mit der Message "renamed files"

```
C:\Users\...source\cprsw\firstRepo>git mv doNotTrack2.txt doTrack.txt
```

```
C:\Users\...source\cprsw\firstRepo>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    doNotTrack2.txt -> doTrack.txt
```

```
C:\Users\...source\cprsw\firstRepo>git commit -m "renamed file"
[master f859db6] renamed file
1 file changed, 0 insertions(+), 0 deletions(-)
rename doNotTrack2.txt => doTrack.txt (100%)
```


Änderungen einer Datei restoren



- Möglich, solange nur "modified" (also in Working Directory) und noch nicht geadded
- Macht Änderungen rückgängig -> erzeugt Stand des letzten commits

1. `git restore firstFile.txt`

Aufgaben:

Nutze die Eingabeaufforderung (=Command line)

1. Stelle sicher, dass "firstFile.txt" committed ist
2. Ändere den Inhalt von "firstFile.txt"
3. Führe eine Status-Abfrage durch (Status "modified"!)
4. Führe ein Restore der "firstFile.txt" durch
5. Führe eine Status-Abfrage durch ("nothing to commit")
6. Öffne "firstFile.txt" -> die Änderungen nun wieder zurückgesetzt

```
C:\Users\...\source\cprsw\firstRepo>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   firstFile.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\...\source\cprsw\firstRepo>git restore firstFile.txt

C:\Users\...\source\cprsw\firstRepo>git status
On branch master
nothing to commit, working tree clean
```

Bestehendes GitHub Repo clonen

Bei Mitarbeit: Klonen bestehender Repos – Link von Projektverantwortlichen

```
git clone https://github.com/path/someRepo.git
```

- Richtet Repo lokal ein
- Lädt alle Inhalte des verlinkten Repos (inkl. gesamten Verlauf!)

Unterschied zu anderen Versionsverwaltungssystemen: Bei **traditionellen Client-Server-Systemen** (bspw. Subversion) wird **nicht der gesamte Verlauf dupliziert!**

Bestehendes GitHub Repo clonen

Repo clonen:

```
git clone https://github.com/MartinaReisHTL/CPRSW_StudentsShare.git
```

Aufgabe:

Nutze die Eingabeaufforderung (=Command line)

1. Navigiere zu deinem CPRSW-Ordner
2. Clone das obige Repo – der Ordner "CPRSW_StudentsShare" wird automatisch erstellt
3. Lasse dir den Inhalt des Repos anzeigen (Versteckte Dateien!)