

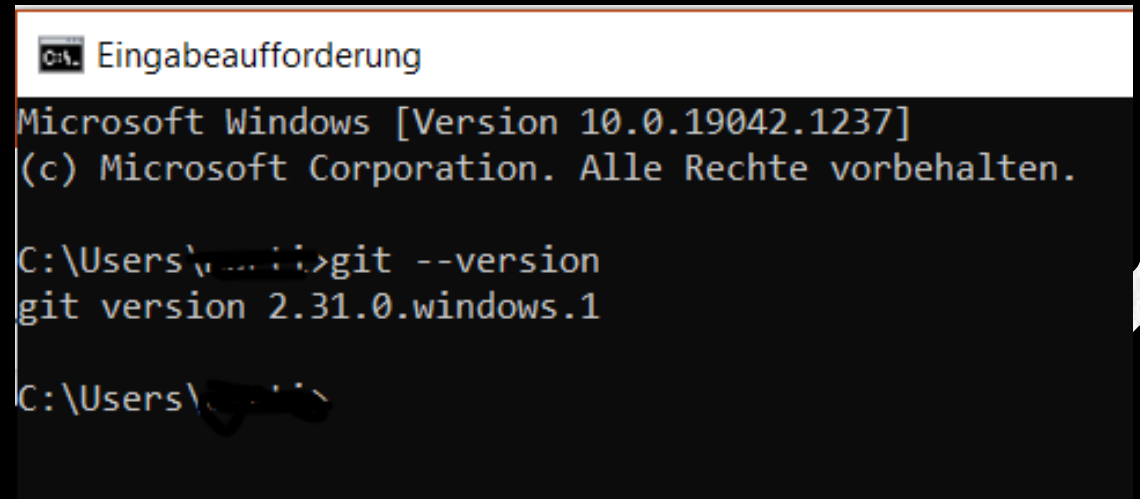
# Installation

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

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

Beim Installer: Standardeinstellungen OK

Test: `git --version`



```
C:\> Eingabeaufforderung

Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

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

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

# Einstellungen

Änderungen manuell im Editor:  
fehlerträchtig!

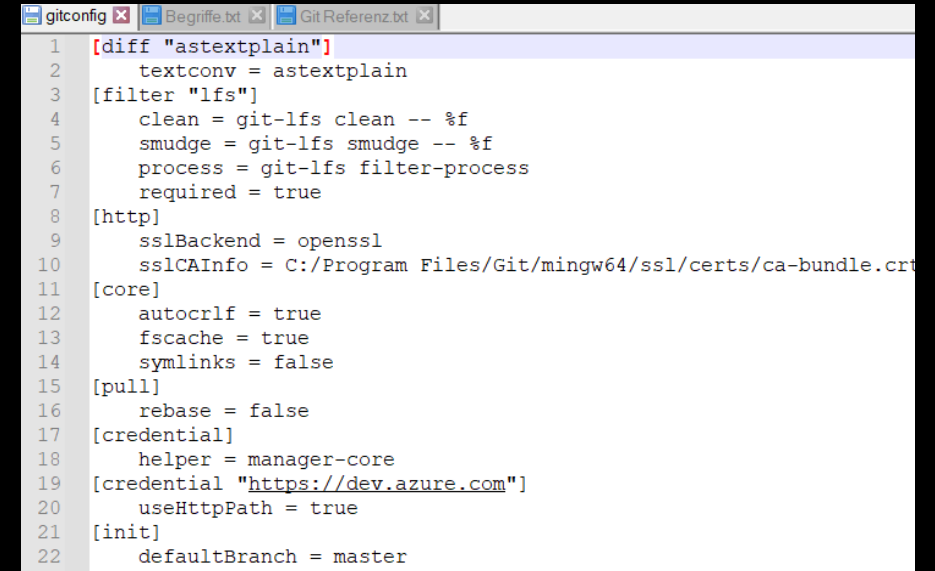
-> Git-Befehle nutzen:

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

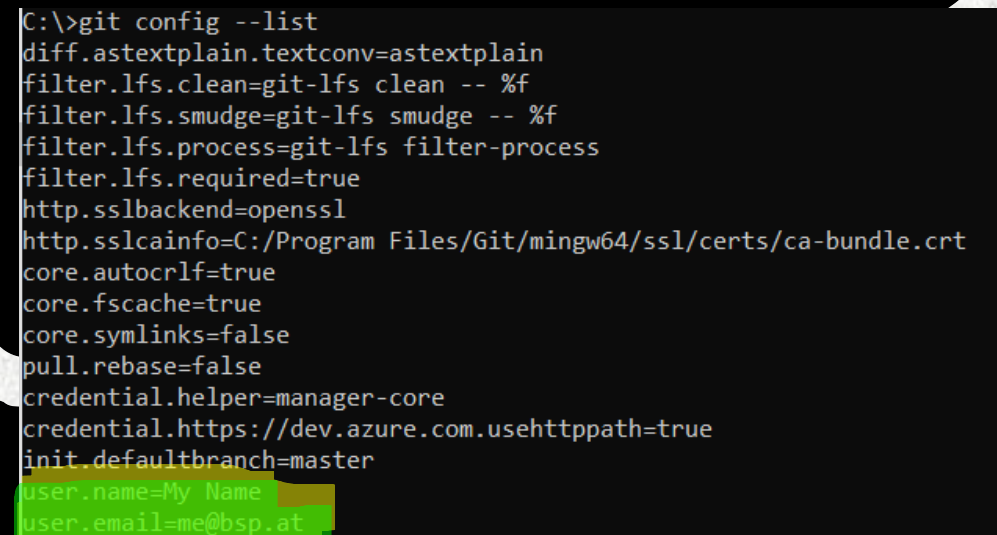
Prüfen:

- `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

ODER:  
Googlen ;-)

```
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/>

# 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!)

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\marti\source\cprsw\firstTest>dir /adh
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: 28B3-FAE3

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

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

Exkurs: Command Line – Befehle:

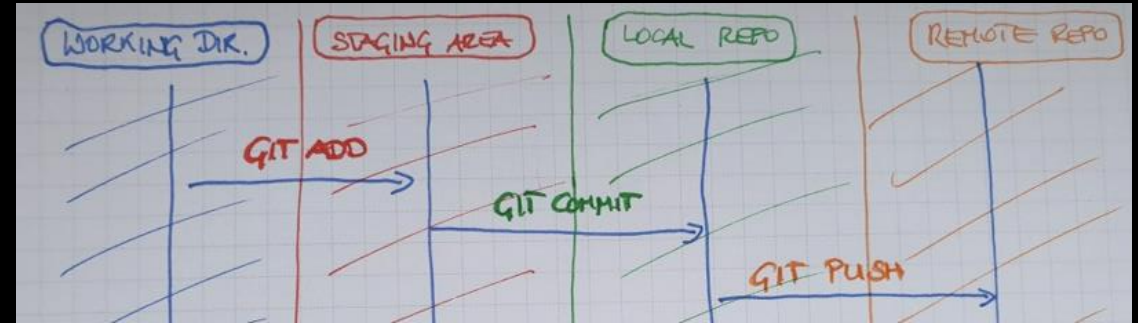
Command	Beschreibung
<code>cd</code> <code>cd..</code> <code>cd\</code>	Verzeichnis wechseln
<code>mkdir</code>	Ordner erstellen
<code>dir</code> (oder Linux: <code>ls</code> )	Liste Ordnerinhalt
<code>dir /adh</code> (oder Linux: <code>ls -a</code> )	Versteckte Ordner anzeigen lassen

```
C:\Users\... \source\cprsw\firstTest>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/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
3. Frage den Status ab
4. Adde die Datei
5. Frage den Status ab

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

C:\Users\... \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:

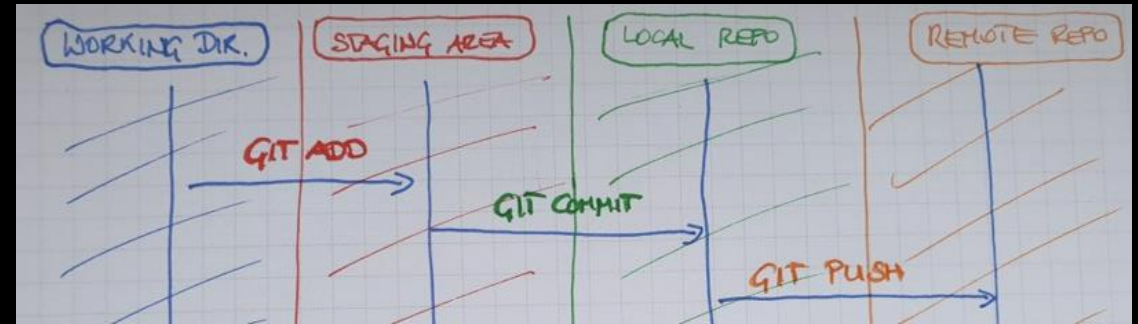
```
git commit -m "First file added"
```

-> 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

Exkurs:  
"to commit a crime" -  
"Eine Straftat begehen"

```
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 (Text-Editor)

`git commit`

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

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

```
cat 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)
```

## Abfrage des eingestellten Editors:

- `git config --list`

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



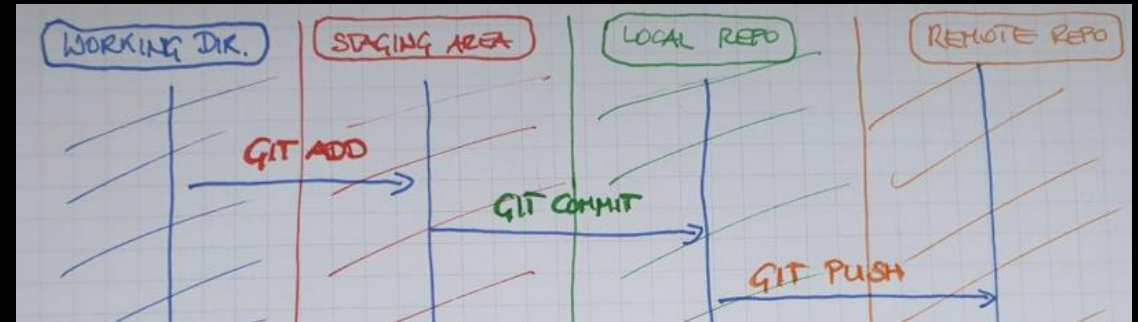
# Inhalt verändern

Dateien adden und committen:

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

ODER

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



```
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")
```

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\marti\source\cprsw\firstRepo>git add firstFile.txt

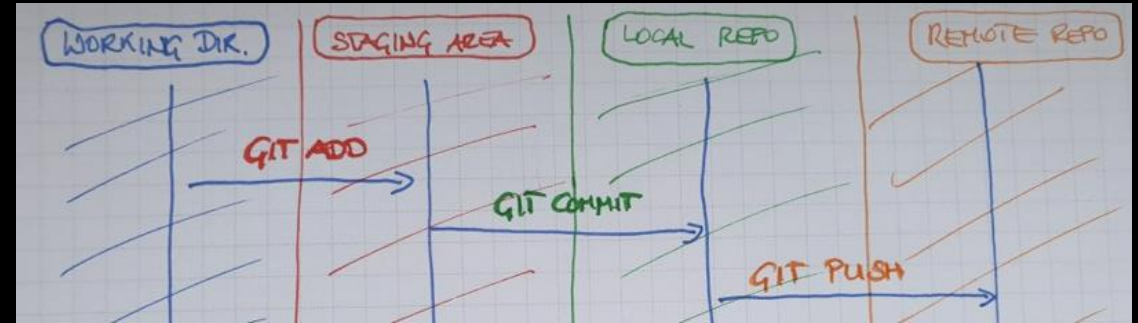
C:\Users\marti\source\cprsw\firstRepo>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   firstFile.txt

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

# Dateien löschen

Vollständig (auch von der Festplatte):

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



Aufgaben:

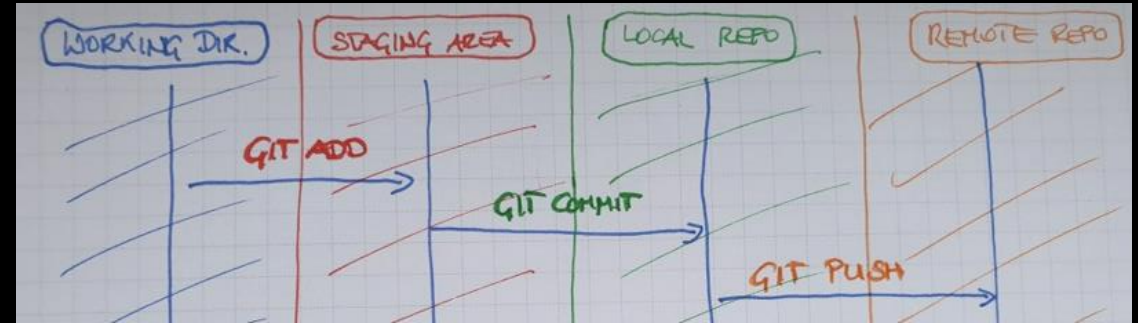
Nutze die Eingabeaufforderung (=Command line)

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"