

Little Known Git Commands

04.09.2025

145 Git Commands

- 75 Procelain Commands
 - add, commit, push, pull, ...
- 70 Plumbing Commands
 - apply, update-ref, cat-file, for-each-ref, ...

Configuration



Globale Dateien

- gitconfig

```
$HOME/.gitconfig
```

```
$XDG_CONFIG_HOME/git/config
```

- gitignore

```
$HOME/.gitignore
```

```
$XDG_CONFIG_HOME/git/ignore
```

Kosmetik

- Farbschema der Git Kommandos

```
git config --global color.status.untracked "yellow reverse"
```

- Gitconfig

```
[color "status"]  
    untracked = yellow reverse
```

- 16 Basis Farben
- 256 ANSI Farben (0 – 255, "#ff00ff ")
- Attribute (bold, dim, ul, blink, reverse, italic, strike)

Alias

- Alias für Git-Kommandos
- '!' Leitet ein Shell-Kommando ein

```
[alias]
```

```
ol = log -oneline
```

```
mybranch = !echo "I am on branch $(git branch --show-current)"
```

Bedingte Konfiguration

- ~/.config/git/private

```
[includeIf "gitdir:/home/stefan/private"]  
    path = ~/.config/git/private
```

- ~/.config/git/private

```
[user]  
    email = private@email.com
```

Globales Ignorieren

- Ignorieren von benutzerspezifischen Dateien
 - Editor/DIE Spezifische Dateien (z.B. *.swp/*.swo)

*.swp

*.swo

Git Attributes

- Datei .gitattributes im Repository

```
pattern attribute_1 attribute_2
```

- Configuration der Line Endings innerhalb des Repositories

```
* text=auto
```

- **Nicht** mehr empfohlen
 - Muss von jedem Benutzer konfiguriert werden

```
git config --global core.autocrlf true
```

diff, log, blame

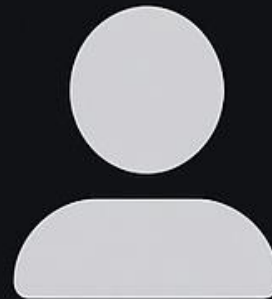
DIFF



LOG



BLAME



- Aufteilen von Diffs in Worte anstatt Zeilen

```
git log --oneline
git log --format=<full, fuller, oneline, short, reference, ...>
git log --graph
git log --reverse
git log --name-only
git log --name-status
git log --author="<name>"
git log -- <file>
```

Log eingrenzen

- Zeilen einer Datei

```
git log -L<start>,<end>:<file>
```

- Veränderungen in einer Funktion
 - Form von <funcname> lässt sich über gitattribute steuern

```
git log -L:<funcname>:<file>
```

- Grep

```
git log --grep=<regex>
```

Word-Diff

- Aufteilen von Diffs in Worte anstatt Zeilen

```
git diff --word-diff  
git show --word-diff  
git blame --word-diff
```

- Regeln zum Splitten von Worten am Beispiel von TeX
 - `\command` nicht splitten

```
[diff "tex"]  
wordRegex = "\\\[a-zA-Z]+|[\{\}]|\\.|[\^\{\}[:space:]]+"
```

Blame

- Ignoriere Whitespace

```
git blame -w
```

- Nachverfolgen von kopierten und verschobenen Passagen

```
git blame -C -C -C
```

Grep

- Sucht in tracked Dateien

```
git grep
```

Notes

- Notizen zu Commits, ohne den Commit zu verändern

```
git notes add
```

```
git notes add <commit-ish> -m "<text>"
```

Veränderte Historie



HEAD erweitern

- Den letzten Commit erweitern

```
git commit --amend --no-edit
```

Ältere Commits erweitern

- Einen beliebigen Commit ergänzen

```
git commit --fixup=<commit-ish>
```

- Fixup-Commits anwenden
 - Wie squash, aber ohne die Message zu verändern

```
git rebase --auto-squash -i <commit-ish>
```

Rebase Konfigurieren

- Einen beliebigen Commit ergänzen

```
git rebase --autosquash --autostash
```

- Implizites autosquash u konfigurieren

```
git config --global rebase.autosquash true  
git config --global rebase.autostash true
```

Stacked Branches

- Rebase von stacked Branches

```
git rebase --update-refs <branch>
```

Wiederholte Merge Konflikte

- REuse REcorded RESolution

```
git config --global rerere.enabled true  
git config --global rerere.autoUpdate true
```

Worktrees

- Gemeinsames Repository
- Kopie der Arbeitskopie

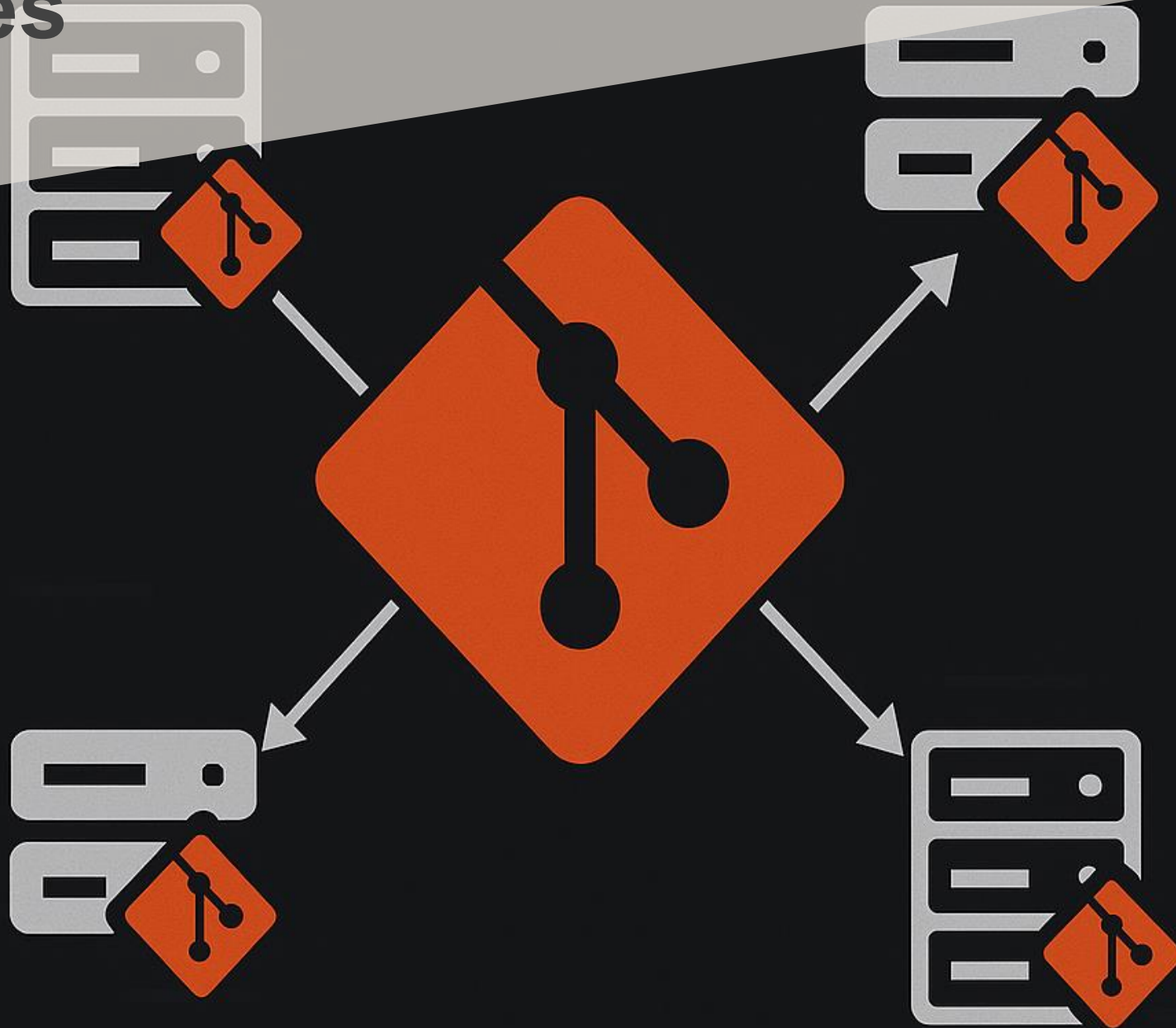
```
git worktree add <path>
```

```
git worktree add -b <new-branch> <path> <commit-ish>
```

```
git worktree list
```

```
git worktree remove <path>
```

Remotes



Sicheres pushen

- Sichereres Force Push nach einer Veränderung der History
 - Kontrolliert ob sich die remote ref geändert hat

```
git push --force-with-lease
```

Push und CI/CD

- Durch Push CI/CD Aktionen ausführen
 - GitLab

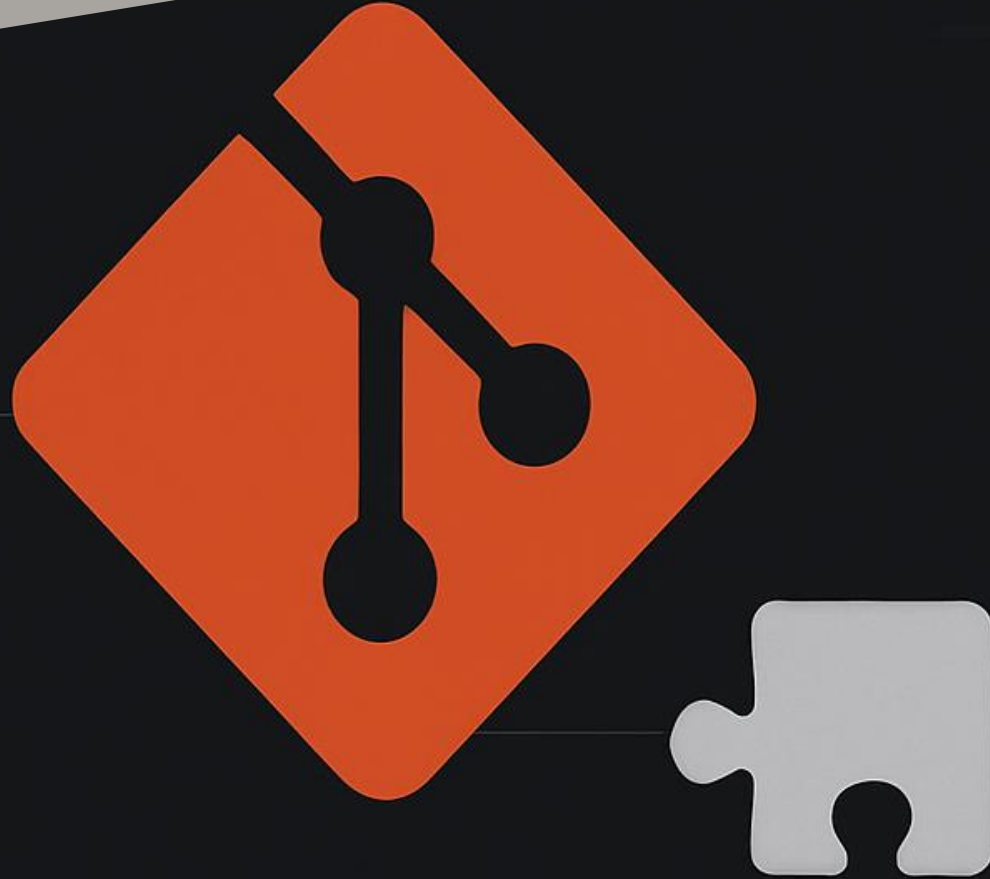
```
git push -o ci.skip
```

```
git push -o merge_request.create
```

```
git push -o merge_request.assign=<user>
```

- GitHub, Azure Devop, Bitbucket
 - [skip ci] in commit Message

Addons



Eigene Skripte

- Per alias
- Executable im Suchpfad

```
git-hello
```

- Aufruf

```
git hello
```

Recover

- Verlorene Objekte wiederfinden

```
git recover
```

- Verlorene Objekte wiederherstellen

```
git recover <sha> --filename <filename>
```

- Quelle:
 - <https://github.com/ethomson/git-recover>

Absorb

- Automatisches erzeugen von Fixup-Commits

```
git absorb
```

- Quelle:
 - <https://github.com/tummychow/git-absorb>

Grosse Repos



Frühlingsputz

- Referenzen auf gelöschte Remote-Branches

```
git fetch --prune
```

- Automatisches Prune

```
git config --global fetch.prune true
```

Maintenance

- Schedule Unterhalts Aufgaben
 - Einmalig pro Repository ausführen

```
git maintainance start
```

- Strategie: Incremental
 - gc: disabled
 - commit-graph: hourly
 - prefetch: hourly
 - loose-objects: daily
 - incremental-repack: daily

Langsames git status?

- git status muss auf alle Dateien ein stat aufrufen
- File System Monitor

```
git config core.untrackedcache true  
git config core.fsmonitor true
```

Fragen?

