

# git - Der einfache Einstieg

eine einfache Anleitung, um git zu lernen. Kein Schnick-Schnack ;)



download the  
cheat sheet  
now. it's free!

von Roger Dudler

Dank an @tfnico, @fhd und Namics

g in english, español, français, indonesian, italiano, nederlands, polski, português, py

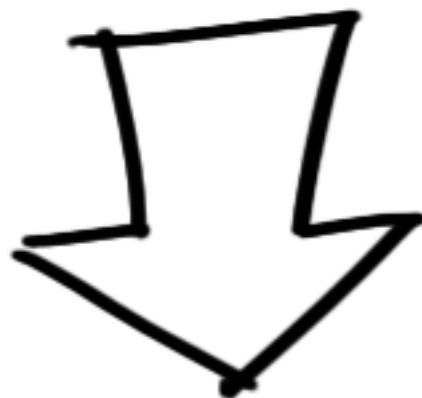
မြန်မာ, 日本語, 中文, 한국어

日本語, 中文, 한국어

Feedback auf github



want a simple  
but powerful  
git client for  
your mac?



## installation

git für OS X herunterladen

git für Windows herunterladen

git für Linux herunterladen

# neues repository erstellen

erstelle ein neues Verzeichnis, öffne es und führe

```
git init
```

aus, um ein neues git-Repository anzulegen.

# ein repository auschecken

erstelle eine Arbeitskopie, indem du folgenden Befehl ausführst:

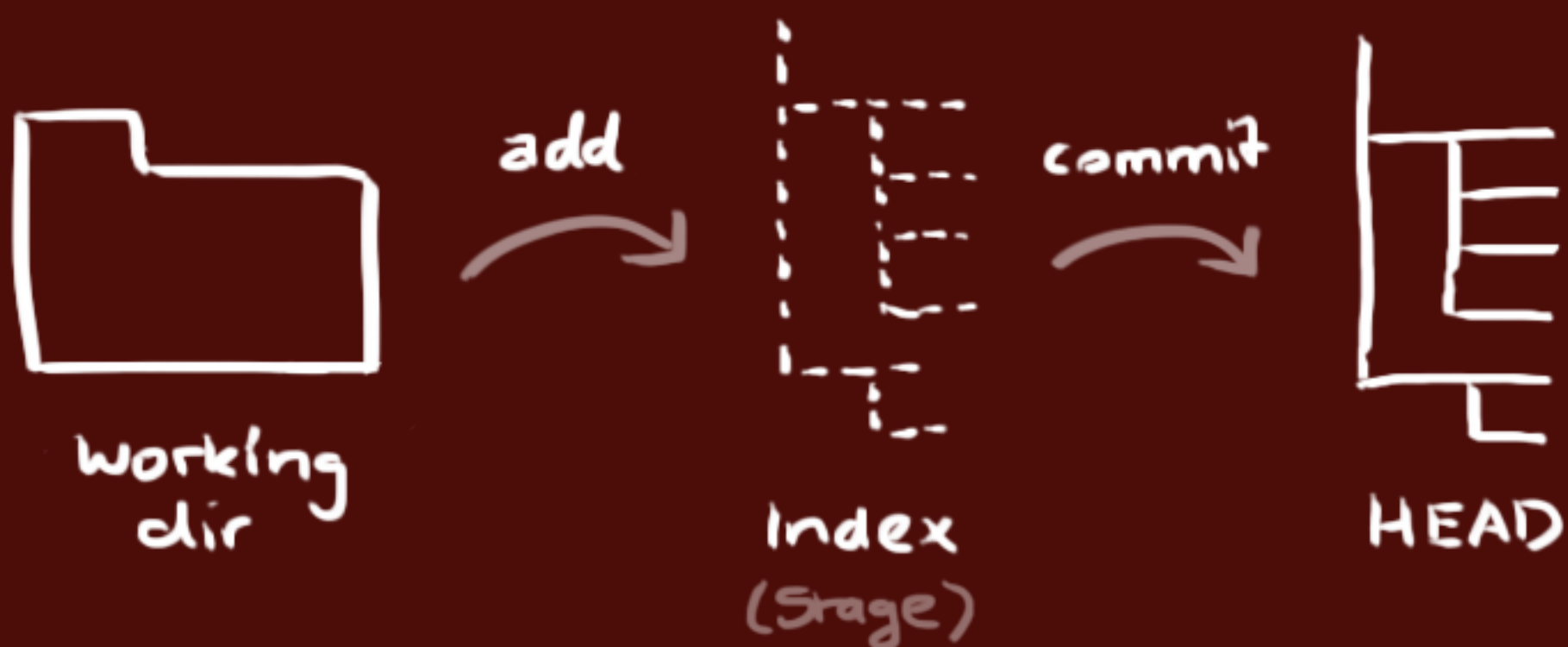
```
git clone /pfad/zum/repository
```

Falls du ein entferntes Repository verwendest, benutze:

```
git clone benutzername@host:/pfad/zum/repository
```

# workflow

Dein lokales Repository besteht aus drei "Instanzen", die von git verwaltet werden. Die erste ist deine **Arbeitskopie**, welche die echten Dateien enthält. Die zweite ist der **Index**, welcher als Zwischenstufe agiert und zu guter Letzt noch der **HEAD**, der auf deinen letzten Commit zeigt.



## add & commit

Du kannst Änderungen vorschlagen (zum **Index** hinzufügen) mit

```
git add <dateiname>
```

```
git add *
```

Das ist der erste Schritt im git workflow, du bestätigst deine Änderungen mit:

```
git commit -m "Commit-Nachricht"
```

Jetzt befindet sich die Änderung im **HEAD**, aber noch nicht im entfernten Repository.

## änderungen hochladen

Die Änderungen sind jetzt im **HEAD** deines lokalen Repositories. Um die Änderungen an dein entferntes Repository zu senden, führe:

```
git push origin master
```

aus. Du kannst *master* auch mit einem beliebigen anderen Branch ersetzen, mehr über Branches erfährst du später.

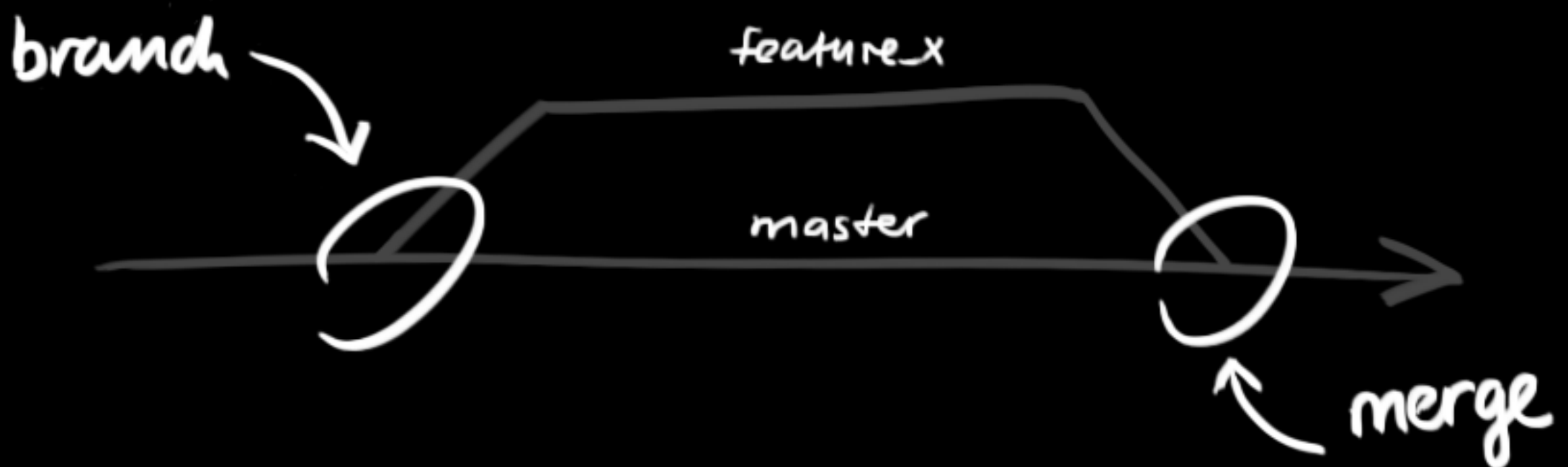
Wenn du dein lokales Repository nicht von einem entfernten geklont hast und du diese aber mit einem anderen Repository verbinden möchtest, musst du dieses mit

```
git remote add origin <server>
```

hinzufügen. Jetzt bist du bereit, deine Änderungen hochzuladen

# branching

Branches werden benutzt, um verschiedene Funktionen isoliert voneinander zu entwickeln. Der *master*-Branch ist der "Standard"-Branch, wenn du ein neues Repository erstellst. Du solltest aber für die Entwicklung andere Branches verwenden und diese dann in den Master-Branch zusammenführen (mergen). Auch das lernst du später.



Erstelle einen neuen Branch mit dem Namen "feature\_x" und wechsle zu diesem:

```
git checkout -b feature_x
```

Um zum Master zurück zu wechseln:

```
git checkout master
```

Und um den eben erstellten Branch wieder zu löschen:

```
git branch -d feature_x
```

Ein Branch ist *nicht für andere verfügbar*, bis du diesen in dein entferntes Repository hochlädst:

```
git push origin <branch>
```

## update & merge

Um dein lokales Repository mit den neuesten Änderungen zu aktualisieren, verwende:

```
git pull
```

in deiner Arbeitskopie, um die Änderungen erst *herunterzuladen (fetch)* und dann mit deinem Stand *zusammenzuführen (merge)*.

Wenn du einen anderen Branch mit deinem aktuellen (z.B. master) zusammenführen willst, benutze:

```
git merge <branch>
```

In beiden Fällen versucht git die Änderungen automatisch zusammenzuführen. Unglücklicherweise ist dies nicht immer möglich

und endet in *Konflikten*. Du bist verantwortlich, diese *Konflikte* durch manuelles Editieren der betroffenen Dateien zu lösen. Bist du damit fertig, musst du das git mit folgendem Befehl mitteilen:

```
git add <dateiname>
```

Bevor du Änderungen zusammenführst, kannst du dir die Differenzen auch anschauen:

```
git diff <quell_branch> <ziel_branch>
```

## tagging

Es wird empfohlen, für Software Releasestags zu verwenden. Dies ist ein bekanntes Konzept, das es schon mit SVN gab. Du kannst einen neuen Tag namens *1.0.0* mit folgendem Befehl erstellen:

```
git tag 1.0.0 1b2e1d63ff
```

*1b2e1d63ff* steht für die ersten 10 Zeichen der Commit-Id, die du mit deinem Tag referenzieren möchtest. Du erhältst die Liste der Commit-IDs mit:

```
git log
```

Du kannst auch weniger Zeichen verwenden, es muss einfach eindeutig



sein.

# änderungen rückgängig machen

Falls du mal etwas falsch machst (was natürlich nie passiert ;) ) kannst du die lokalen Änderungen mit:

```
git checkout -- <filename>
```

auf den letzten Stand im HEAD zurücksetzen. Änderungen, die du bereits zum Index hinzugefügt hast, bleiben bestehen.

Wenn du aber deine lokalen Änderungen komplett entfernen möchtest, holst du dir den letzten Stand vom entfernten Repository mit folgenden

Befehlen:

```
git fetch origin
```

```
git reset --hard origin/master
```



# nützliche tricks

Eingebaute git-GUI:

```
gitk
```

Farbige Konsolenausgabe:

```
git config color.ui true
```

Eine Zeile pro Commit in der Logausgabe:

```
git config format.pretty oneline
```

Interaktives Hinzufügen von Änderungen:

```
git add -i
```

## links

### grafische clients

GitX (L) (OS X, Open Source)

Tower (OS X)

Source Tree (OS X, kostenlos)

GitHub for Mac (OS X, kostenlos)

GitBox (OS X)

# anleitungen

Git Community Book

Pro Git

Think like a git

GitHub Help

A Visual Git Guide

Clarify

Building websites? Working with Designers? Try Clarify.

A Project by Roger Dudler, Author of The Git Simple Guide



# kommentare

31 Comments

git - the simple guide

<sup>1</sup> Login ▾

Recommend 11

Tweet

Share

Sort by Newest ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS <sup>?</sup>



Name



Ann • 9 days ago

How can I create a release? I don't want to push o pull nothing just create a release? Is there no way to do it?

^ | ▾ • Reply • Share ›



Ann • 9 days ago

How can I create a release? I don't to push o pull nothing just create a release? Is there no way to do it?

^ | ▾ • Reply • Share ›



Ronny M. • a year ago

Hallo,

welchen Befehl muss ich absetzen, wenn ich nur meine lokalen Dateien aktualisieren will. Ich habe mit "git clone https://github.com/XXX/YYYY" mir ein PHP Script geholt. Zu diesem gibt es nun Aktualisierungen. Mit "git pull xhttps://github.com/XXX/YYYY" bekomme ich diese aber nicht... Ich erhalte die Fehlermeldungt "fatal: Kein Git-Repository (oder irgendein Elternverzeichnis): .git".

Muss ich mir extra eine \*.git Datei erstellen, damit ich die Änderungen, welche im gleichen Ordner ankommen sollen, erhalten kann?


Ich möchte nur die jeweils aktuellen Versionen haben, selbst aber zur Zeit keine Änderungen senden.

Vielen Dank für Eure Hilfe,  
LG Ronny

^ | v · Reply · Share ›

 **Ilmu-Kiv Jolia** → Ronny M. · a year ago  
Hast du git init ausgeführt?

^ | v · Reply · Share ›

 **Florian Wittmann** · a year ago  
Danke für diesen wundervollen Überblick! Wer noch ein wenig mehr wissen will, dem kann ich meinen Git Videokurs ans Herz legen:  
<https://www.udemy.com/git-v...>

Was lernst du alles im Kurs?

- Du lernst Git Repositories zu initialisieren oder von einem bestehenden Repository zu klonen.
- Du kannst am Ende Dateien committen und die History, den Verlauf von Commits betrachten.
- Du hast einen Überblick über die Branchesfunktionalität, kannst sie erstellen, mergen und eventuell auftretende Konflikte beheben.
- Mit Remotes kannst du mit anderen Entwicklern zusammenarbeiten und gemeinsam an Projekten arbeiten.
- Außerdem lernst du, wie du deine Commits nachträglich bearbeiten, löschen oder zusammenstauchen kannst.
- Du kannst eigene Aliase definieren, um dir lästige Tipparbeit zu sparen und produktiver zu arbeiten.

Über den Link oben gibt es den Kurs zur Zeit auch deutlich günstiger :-)

^ | v · Reply · Share ›

 **Edgar Wahl** · a year ago  
Also ein Version Controllsystem funktioniert in der Form:

Programmname = Master (Der Master verwaltet das Programm als Programmiercode).

Der branch ( Das ist der teil der die Version ausmacht). Programmname (winword=master). winword ist ein ordner auf Ihrem Computer.  
als Unter ordener (Sinnbild) 0.01 ist der der erste Code der geschrieben wurde.  
als Unter ordener 0.02 wird nun die Oerfläche programmiert

0.03 ist dann das Prebuild beider zusammengeführter Teile.


Die Branch ist praktisch die Niederlassung des programmes in einem Bereich.

git und svn machen dasselbe jedoch unterschiedliche Konzepte in der Verwaltung.

^ | v · Reply · Share ›

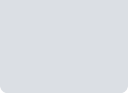
 **Edgar Wahl** · a year ago  
Kann man eigentlich mit git nur fehlende dateien zwischen github und lokal abgleichen

^ | v · Reply · Share ›

 **patrick** · a year ago  
cooles ding!

^ | v · Reply · Share ›

 **Lasse Schultebraucks** · 2 years ago



Für Anfänger vielleicht ein wenig schwierig das Prinzip hinter Git und allgemein VCS zu verstehen, aber als Erfahrender gut zu verstehen und als Referenz zu benutzen

^ | v · Reply · Share ›



**Mueller** · 2 years ago

Völlig "unsable for beginners",  
denn ich kenne die Begriffe und den Sinn und Zweck dahinter gar nicht.

^ | v · Reply · Share ›



**Becky** · 2 years ago

Danke! :-) schön aufgearbeitet

^ | v · Reply · Share ›



**Basti** · 2 years ago

Sehr schöner Guide! Was ich etwas vermisse: Es werden zwar alle Befehle gezeigt die nötig sind, aber es wäre wirklich schön zu wissen wofür ein Begriff im Befehl steht (z.B. bei: git remote add origin <server>). Denn das wird noch nicht so richtig klar und ich denke, dass würde zum Gesamtverständnis beitragen.  
Ansonsten Top, weiter so! :)

^ | v · Reply · Share ›



**Pit** · 2 years ago

Hi, der Linux-Download-Link funktioniert nicht mehr.

^ | v · Reply · Share ›



**Simon Sobisch** · 2 years ago

Danke. Kannst du das stashen noch ergänzen? Oder zählt das für dich zu "schnick-schnack"?

Wenn ich das richtig verstanden habe ist das zum Aufheben für nicht fertige Änderungen, die man noch nicht in den Index hinzufügen möchte und ist u.a. notwendig, wenn man ein `pull` ausführen möchte.

Eine der besten Erweiterungen von git scheint mir `bisect` zu sein, das wäre sicherlich auch hilfreich. Vielleicht ergänzt du beides in einem Bereich "schnick-schnack" oder "Für Fortgeschrittene" unter Verwendung der Begriffe (damit man über die Suche auch dahin kommt)?

^ | v · Reply · Share ›



**Markus** · 2 years ago

Super für den Einstieg in Git!

^ | v · Reply · Share ›



**Yo** · 3 years ago

nice 1

^ | v · Reply · Share ›



**Michl** · 3 years ago

Besten Dank nochmal für deine wirklich gute Anleitung.

^ | v · Reply · Share ›



**Bonze** · 3 years ago

"git add \*" is different than "git add .", i would recommend the latter. explained here: <http://stackoverflow.com/qu...>

^ | v · Reply · Share ›



**kleinerpimml** · 3 years ago


echt geil alter! Ich hoffe du retttest mir morgen meinen Arsch!

^ | v · Reply · Share ›




**Bestrapper** · 3 years ago

THANK YOU

- ^ | v · Reply · Share ›
- 

c3po · 3 years ago

this was exactly what I was searching for, Thanks!

^ | v · Reply · Share ›
- 


Sebastian · 3 years ago

Danke! Super für den Einstieg

^ | v · Reply · Share ›
- 

DAU · 3 years ago

that was very helpful thanks alot !!!

^ | v · Reply · Share ›
- 


mik · 3 years ago

Thanks, gute Hilfe .... hatte nach X Versuchen immer noch kein Plan. Das half ;-)

^ | v · Reply · Share ›
- 


Martin Lehmann · 3 years ago

Herzlichen Dank, passt genau für einen SVN->GIT-Wechsler!

^ | v · Reply · Share ›
- 


Steffen · 3 years ago

Danke!!! :)

^ | v · Reply · Share ›
- 

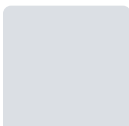
Steffen ↗ Steffen · 3 years ago

sehr gut für Anfänger!

^ | v · Reply · Share ›
- 

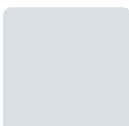
inka · 3 years ago

Danke! Ich hab echt keine Ahnung, wie Du Dich in DAUs wie mich so reinversetzen konntest, dass diese Anleitung zustande kam, aber herzlichen Glückwunsch, alle Kollegen sind dran gescheitert. :D

^ | v · Reply · Share ›
- 


Vincent · 3 years ago

Sehr gut für den Einstieg in GIT. Jedem Anfänger empfehle ich diese Seite.

^ | v · Reply · Share ›
- 

Marius · 3 years ago

Vielen Dank!!

^ | v · Reply · Share ›
- 

fabi · 3 years ago

wow, i guess this guid all i ever wanted. thank u

^ | v · Reply · Share ›

## The Top 5 MacOS Tips

Learn these simple tricks of your Mac

Learn More

Sponsored by [mackeeper.com](#)



