

# git - Der einfache Einstieg

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

[Twittern](#)

4,747

von Roger Dudler

Dank an @tfnico, @fhd und Namics

› Anleitung in english, español, français, italiano, nederlands, português, русский, tür

□□□□□, 日本語, 中文, 한국어

日本語, 中文, 한국어

Feedback auf github

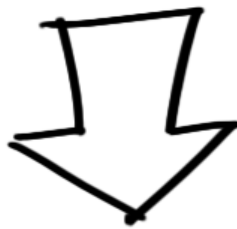


**Are You a Front-End Developer?**

by Roger Dudler, Author of the Git Simple Guide

Try Frontify

Now Free with  
**Github** Integration!



## 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:

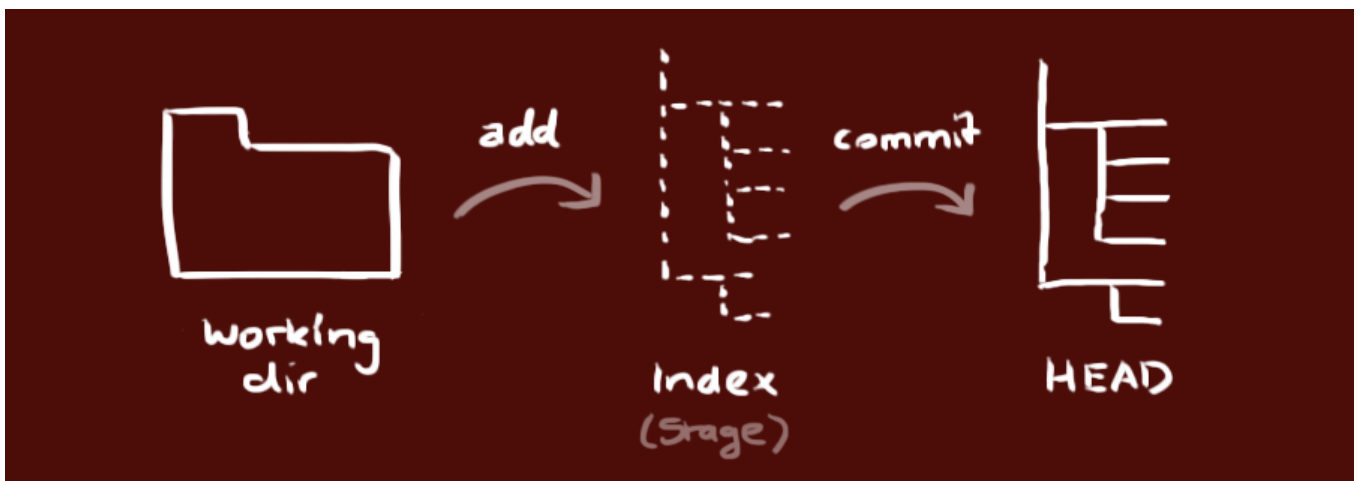
```
git clone /pfad/zum/repository
```

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.

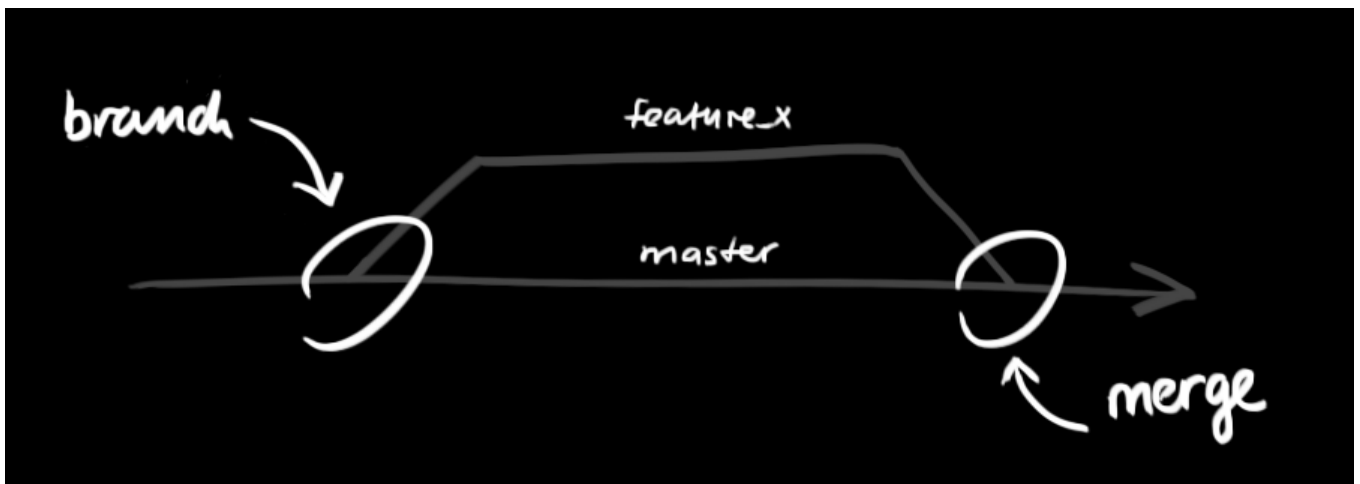
Wenn du dein lokales Repository nicht von einem entfernten geklont  
hast und du diese aber mit einem anderen Repository verbinden  
möchtest, musst du dieses mit

```
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

33 Comments

git - the simple guide

 hanspeter schlap... ▾

Sort by Newest ▾

Share  Favorite 

Join the discussion...

**Chris** • 2 months ago

Gibt es irgendwo eine Anleitung für einen guten Workflow? Ich habe nun ein PHP-Projekt in einem lokalen Ordner und auf Bitbucket. Wie kann ich nun auf den Dev-Server arbeiten? Der ist nur per FTP zugänglich. Wie kommen die Files wenn ich CTRL+S drücke im Editor auf den Server?

  • Reply • Share ›**Nico Schneider** • 3 months ago

git add -p ist empfehlenswert!

  • Reply • Share ›**Simon** • 3 months ago

Extrem gutes Tutorial! Vielen Dank!

  • Reply • Share ›**tk** • 3 months ago

Hat mir sehr geholfen. Danke!

1   • Reply • Share ›**Geziefer** • 4 months ago

Sehr gut!

1   • Reply • Share ›**Thommy** • 5 months ago

statt 'git add \*' sollte man immer 'git add .' verwenden, da die Form mit '\*' z.B. .gitignore nicht beachtet --> es könnten also Dateien ins Repository übernommen werden, die da nicht hingehören.

2   • Reply • Share ›**Konrad Abe** • 5 months ago

Perfektes Tutorial für Einsteiger bzw. Referenz zum Nachschlagen. Danke!

1   • Reply • Share ›**Jaydot** • 6 months ago

git add \*

funktioniert bei mir übrigens nicht. Stattdessen nutze ich

git add .

um alle Änderungen auf einmal zu stashen.

Vielleicht hilft's ja irgendwem.

  • Reply • Share ›**bluevirus** → Jaydot • 5 months ago

git add --all tut's auch

  • Reply • Share ›**Jaydot** • 6 months ago

Besten Dank!

Source Tree gibt's übrigens auch für Windows. Funktioniert (logischerweise) ganz wunderbar mit Bitbucket :-)

  • Reply • Share ›**top** • 7 months ago



Die beste Anleitung die ich kenne!

^ | v • Reply • Share ›



**Armigo** • 8 months ago

Einfach nur hervorragend! Echt klasse!

^ | v • Reply • Share ›



**Sven** • 9 months ago

Schöne Anleitung, danke!

^ | v • Reply • Share ›



**mtk** • 10 months ago

Toll, Danke!

^ | v • Reply • Share ›



**Rico** • a year ago

Schöne Anleitung, besten Dank!

1 ^ | v • Reply • Share ›



**Jeffrey** • a year ago

Vielen Dank für die Tolle Anleitung hat mir den Einstieg erleichtert

1 ^ | v • Reply • Share ›



**Flo** • a year ago

Sehr schöne Anleitung. Da macht sogar branchen und resetten Spaß. :D

1 ^ | v • Reply • Share ›



**Jo Schneider** • a year ago

Das ist mal eine Anleitung, die Spaß macht. Ergänzung:

Der Befehl „git branch“ zeigt alle existierenden Zweige an, wobei der aktuelle Branch mit einem Sternchen gekennzeichnet ist.

^ | v • Reply • Share ›



**Amarok** • a year ago

Eine List von -Non-OSX-Clients mit GUI wäre echt supper ;-)

Davon abgesehen sucht diese Seite allerdings ihres Gleichen in ihrer Einfachheit, Übersichtlichkeit und Anwendbarkeit :-)

4 ^ | v • Reply • Share ›



**chilloutcloud** ➔ **Amarok** • a year ago

Unter Windows benutze ich TortoiseGit als GUI

([https://code.google.com/p/tortoise...](https://code.google.com/p/tortoise-git/))

2 ^ | v • Reply • Share ›



**gitdummy** • a year ago

sehr übersichtlich und einfach. danke!

1 ^ | v • Reply • Share ›



**Jonathan Gruber** • a year ago

Danke, sehr hilfreich!

1 ^ | v • Reply • Share ›



**Timo** • a year ago

echt klasse

1 ^ | v • Reply • Share ›



**Willy Makend** • a year ago

Beste einföhrung in Git. Vielen Dank.

3 ^ | v • Reply • Share ›



**Strandurlaub** • 2 years ago

Ein echt gutes Tutorial, danke :)

Habe es über diese Seite gefunden: <http://www.nutgit.com/de/>

Mit dem selben Design :P

Best Tutorial zum Erstellen von Git Repositories - Windows und OSX

Dort bekommt man kostenlose Git Hosting. Wer eine grafische Oberfläche braucht sollte auf <http://www.github.com/> schauen, auf github sind private Git Repositories aber nicht kostenlos.

^ | v • Reply • Share ›



**u** • 2 years ago

Unter Änderungen hochladen: "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 (...)". Kann man das nicht klarer formulieren? Vorschlag: Klären, wer sind "diese" und "dieses"?

7 ^ | v • Reply • Share ›



**Danilo** • 2 years ago

Sehr gut! Den Abschnitt: grafische clients noch um Clients für Windows und Linux erweitern.

6 ^ | v • Reply • Share ›



**Boris** • 2 years ago

danke für die einföhrung. vlt solltest du beim git init auf '--bare' hinweisen, damit beim ersten push die "...to non-bare repository" meldung bzw. der resultierende Konfigurationsaufwand vermieden wird

5 ^ | v • Reply • Share ›



**tobi** • 2 years ago

1A! Besten Dank für diese kleine Einföhrung.

4 ^ | v • Reply • Share ›



**mosermu** • 2 years ago

super! vielen dank! als "englischbanause" bin ich dir echt dankbar für die coole anleitung :-)

3 ^ | v • Reply • Share ›



**Chris** • 2 years ago

GREAT WORK! When I started to use Git, i was really overwhelmed with all that "beginner" tutorials out there. This one shows just the basics, which are needed to start.

2 ^ | v • Reply • Share ›



**Johannes Vogel** • 2 years ago

This is simply one of the tutorials out there! Thx!

2 ^ | v • Reply • Share ›



**Chris M.** • 2 years ago

Good 5/5\*

2 ^ | v • Reply • Share ›

#### ALSO ON GIT - THE SIMPLE GUIDE

##### git - 簡単ガイド

1 comment • 2 years ago



**Mormont** — Git is git. I personally use it.

##### git - petit guide - no deep shit!

15 comments • 2 years ago



**SGH** — Très utile, juste ce qu'il faut pour Git et pas besoin de lire un bouquinencore Merci

#### WHAT'S THIS?

##### git - la guía sencilla

68 comments • 2 years ago



**Jose** — Una duda, digamos que ya tengo un repositorio en GitHub y ahora necesito subirlo a un ...

##### git - basit rehber - atla deve değil!

5 comments • 2 years ago



**onurozgurozkan** — Türkçeye çeviren arkadaşlara teşekkür ederiz. Bu tarz kaynakların daha çok çevirilmesi lazım.

✉ Subscribe

➦ Add Disqus to your site

🔒 Privacy