git - the simple guide

just a simple guide for getting started with git. no deep shit ;)

Tweet

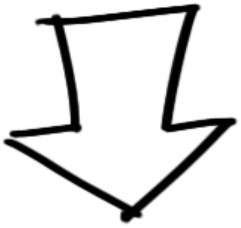
by Roger Dudler

credits to @tfnico, @fhd and Namics

ı deutsch, español, français, indonesian, italiano, nederlands, polski, português, pycc

ြမန္မာ, 日本語, 中文, 한국어 Vietnamese

please report issues on github



setup

Download git for OSX

Download git for Windows

Download git for Linux

create a new repository

create a new directory, open it and perform a

```
git init
```

to create a new git repository.

checkout a repository

create a working copy of a local repository by running the command

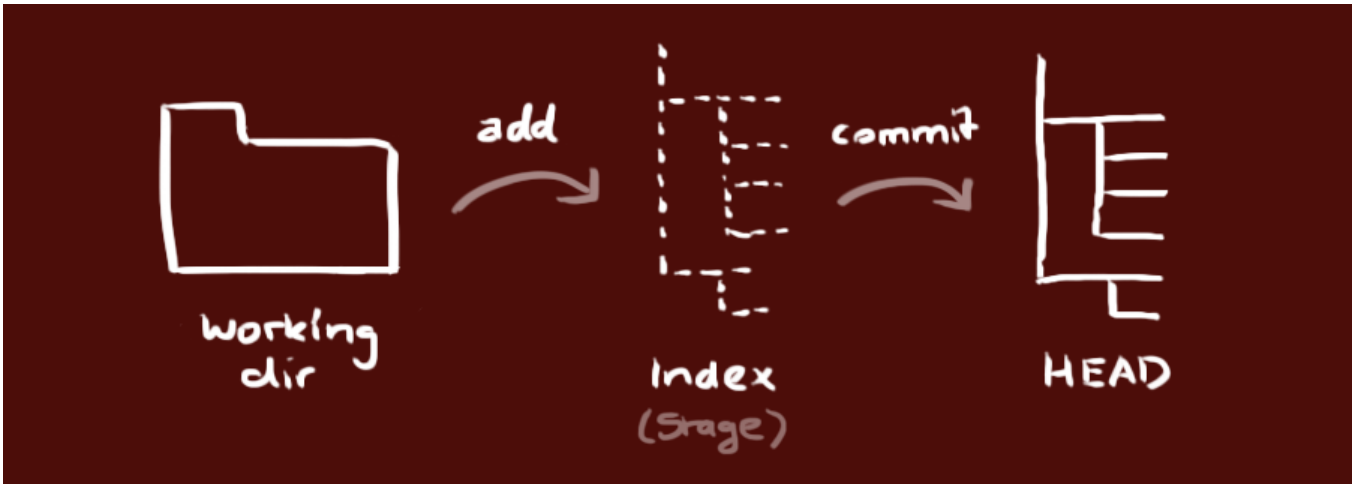
```
git clone /path/to/repository
```

when using a remote server, your command will be

```
git clone username@host:/path/to/repository
```

workflow

your local repository consists of three "trees" maintained by git. the first one is your `Working Directory` which holds the actual files. the second one is the `Index` which acts as a staging area and finally the `HEAD` which points to the last commit you've made.



add & commit

You can propose changes (add it to the **Index**) using

```
git add <filename>
```

```
git add *
```

This is the first step in the basic git workflow. To actually commit these

changes use

```
git commit -m "Commit message"
```

Now the file is committed to the **HEAD**, but not in your remote repository yet.

pushing changes

Your changes are now in the **HEAD** of your local working copy. To send those changes to your remote repository, execute

```
git push origin master
```

Change *master* to whatever branch you want to push your changes to.

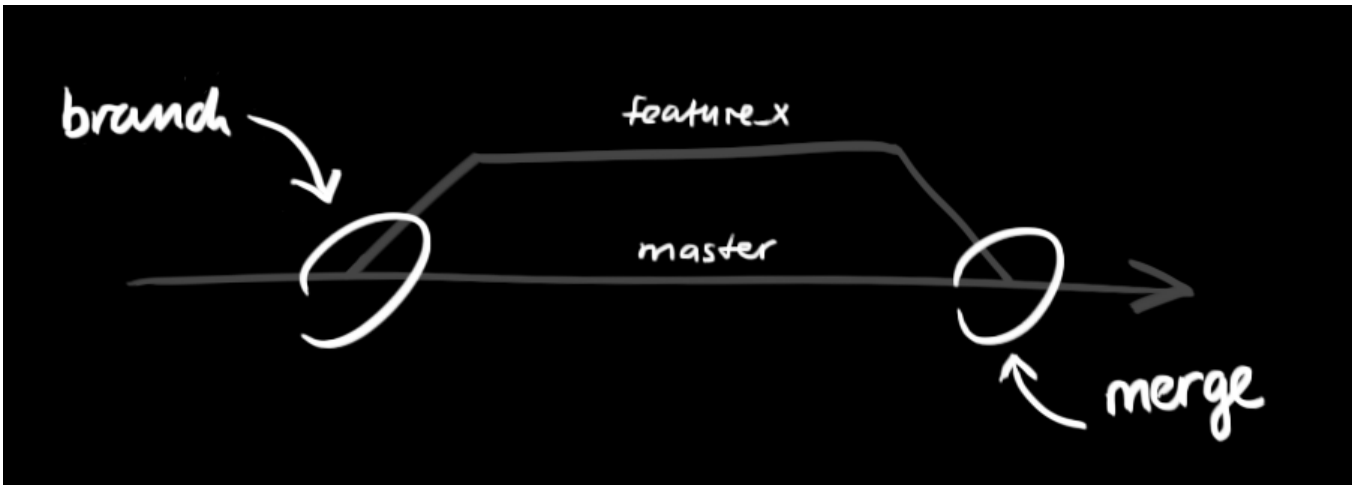
If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

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

Now you are able to push your changes to the selected remote server

branching

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.



create a new branch named "feature_x" and switch to it using

```
git checkout -b feature_x
```

switch back to master

```
git checkout master
```

and delete the branch again

```
git branch -d feature_x
```

a branch is *not available to others* unless you push the branch to your

remote repository

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

update & merge

to update your local repository to the newest commit, execute

```
git pull
```

in your working directory to *fetch* and *merge* remote changes.

to merge another branch into your active branch (e.g. master), use

```
git merge <branch>
```

in both cases git tries to auto-merge changes. Unfortunately, this is not always possible and results in *conflicts*. You are responsible to merge those *conflicts* manually by editing the files shown by git. After changing, you need to mark them as merged with

```
git add <filename>
```

before merging changes, you can also preview them by using

```
git diff <source_branch> <target_branch>
```

tagging

it's recommended to create tags for software releases. this is a known concept, which also exists in SVN. You can create a new tag named *1.0.0* by executing

```
git tag 1.0.0 1b2e1d63ff
```

the *1b2e1d63ff* stands for the first 10 characters of the commit id you want to reference with your tag. You can get the commit id by looking at the...

log

in its simplest form, you can study repository history using.. `git log` You can add a lot of parameters to make the log look like what you want. To see only the commits of a certain author:

```
git log --author=bob
```

To see a very compressed log where each commit is one line:

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

Or maybe you want to see an ASCII art tree of all the branches,

decorated with the names of tags and branches:

```
git log --graph --oneline --decorate --all
```

See only which files have changed:

```
git log --name-status
```

These are just a few of the possible parameters you can use. For more, see `git log --help`

replace local changes

In case you did something wrong, which for sure never happens ;), you can replace local changes using the command

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

this replaces the changes in your working tree with the last content in HEAD. Changes already added to the index, as well as new files, will be kept.

If you instead want to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it like this

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

useful hints

built-in git GUI

```
gitk
```

use colorful git output

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

show log on just one line per commit

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

use interactive adding

```
git add -i
```

links & resources

graphical clients

GitX (L) (OSX, open source)
Tower (OSX)
Source Tree (OSX & Windows, free)
GitHub for Mac (OSX, free)

GitBox (OSX, App Store)

guides

Git Community Book
Pro Git
Think like a git
GitHub Help
A Visual Git Guide

get help

Git User Mailing List
#git on irc.freenode.net

comments

809 Comments git - the simple guide 1 Login

Recommend 372 Share Sort by Newest

Join the discussion…

faris • 3 days ago
Thanks! sometime i forget the syntax and just refreshing my memory by coming here..
1 • Reply • Share ›

ben • 3 days ago
fantastic guide! i've been looking for a guide like this!
• Reply • Share ›

Antonio Joseph Smith • 3 days ago
This is a lifesaver. I have been mixing up these steps all week. Thank you.!!!
• Reply • Share ›

Tadeus Araújo • 4 days ago
THANK YOU SO MUCH FOR PROVIDING THIS TUTORIAL!!!

As I'm starting to working with Git, this made me have a nice outlook of the commands and features.
• Reply • Share ›

KevinMcCready • 21 days ago

How do I find a version of a program then compile it and install it? eg sane-backends not yet released 1.0.26?

Reply • Share ›

Aye Mya Han • 22 days ago

Thank you and this article make me clear about git command and it's usage.

1 Reply • Share ›

Stork • 23 days ago

Thank you - didn't need the deep shit, didn't get the deep shit. I appreciate that you put this together.

1 Reply • Share ›

englishextra • a month ago

So many bots down there in the comments. What a shame!

Reply • Share ›

Pankaj Kolhe • a month ago

awsome man....Really amazing guideline

Reply • Share ›

Amit Chaudhary • a month ago

They say "simplicity is the ultimate sophistication". This article proves it.

4 Reply • Share ›

Sudipto Roy • a month ago

This really just Saved my Day! The bestest guide to GIT for beginners! I shared it with all in my team.

2 Reply • Share ›

philnc • a month ago

Thanks for this. I am mass-mailing it to all my developer colleagues who still don't "git" it.

Reply • Share ›

Anant Prajapati • a month ago

I like this tutorial and its very useful to learn command base git... thanks

Reply • Share ›

Sunny Kusawa • 2 months ago

This is what I looking for. Thanks

Reply • Share ›

Raj Singh • 2 months ago

This only offers quick shortcuts only. Better to understand basics first rather than using commands directly.

1 Reply • Share ›

> **borja** → Raj Singh • a month ago
>
> I agree with you. Of course, I am by no means saying that this article is not helpful, but it looks more like a list of shortcuts so newbies can easily remember the flow, more than a real guide. For example, it starts talking about commits like if newbies had to know already what they are... In other words, I think this resource will come handy after an introduction to Git.
>
> I will try to learn somewhere else but I am saving this to my Evernote so I can come back to it when the time is right.
>
> Thanxs for your work Olli
>
> Reply • Share ›

Olli Lappalainen ➔ Raj Singh • 2 months ago

Isn't it basics just to git init, add, commit, push and pull? That's the main thing you have to do. If you mean by basics "why it works or how it works". That's deep shit.

3 ^ | v • Reply • Share ›

Jean • 2 months ago

New (and lost) on git, having crawled a lot of docs, until I found your page. You saved my day !!! (John)

^ | v • Reply • Share ›

Mohammad Sharaf Ali • 2 months ago

After switching from svn to git it was a nightmare. Glad I came to this helpful guide and cleared all my concepts. Super great work man!

^ | v • Reply • Share ›

Miguel • 2 months ago

A small note for tagging ...
You can push an new tag with 'git push --tags' or 'git push origin <tagname>'

^ | v • Reply • Share ›

KaTIOWa • 2 months ago

Super clear!

^ | v • Reply • Share ›

Priyanga Smith • 2 months ago

Thanks a lot! Just great!

^ | v • Reply • Share ›

James Michels • 2 months ago

Thank you for this guide. I was looking for hours to find a way to fetch from a bare repo. Your git reset --hard origin/master saved me :)

^ | v • Reply • Share ›

Sanura • 2 months ago

Thank you! Short and sweet essentials. Love it

^ | v • Reply • Share ›

Rejeesh • 2 months ago

This is really cool. Love it. Simply simple !!!
One thing I learnt during the past few days as a novice was to specify multiple files in a single command, like git add <filenames>, here <filenames> have to be specified with a single white-space, e.g., git add file1 file2 file3

^ | v • Reply • Share ›

Max Lans • 2 months ago

KIS is the key, always.... Keep It Simple! Well done, my compliments...

^ | v • Reply • Share ›

Mark • 2 months ago

Thank you so much!

^ | v • Reply • Share ›

Kaan • 3 months ago

Very helpful!

^ | v • Reply • Share ›

RobinMC • 3 months ago

Thanks for making this!

^ | v • Reply • Share ›

George • 3 months ago

Very nice.

• Reply • Share ›

Ruediger Willenberg • 3 months ago

This is mostly great. I know it's the point to keep it simple, but I would strongly suggest to either substitute or augment the "git init" with "git --bare init". Github most likely does this internally when setting up a repository for you by web interface; however people who want to set up a shared repository for their company or research group most likely want a bare server repository to keep their (shared) projects.
Like other people coming from SVN etc., I went by all the "git for dummies" explanations like this one that talk about using "git init" and got cryptic errors pushing into my server repository for the first time. Even some of the online explanations for what's happening are not helpful for said dummies like me, took me a while to get it. Save us the headache by making clear that you need "git --bare init" for a server repository that you'd want to push to.

1 • Reply • Share ›

hunt fred • 3 months ago

I use mostly GUI tools. But your guide makes git super easy. Your title "...no deep shit" shows you understand the frustration of developers. You seem to listen and understand the need for such a great topic. I wish you publish a book one day.

1 • Reply • Share ›

David Gatti • 3 months ago

This is lovely, thank you for creating this, I can share your work with anyone now new to Git - amazing :)

• Reply • Share ›

Shell • 3 months ago

Great job!! Thank you!!

• Reply • Share ›

Yuhui Zheng • 3 months ago

Simply elegant!

• Reply • Share ›

Yondaime008 • 3 months ago

Really cool guide :)

• Reply • Share ›

Somasekara Rao • 3 months ago

Thank U; Simple and Enough to play with GIT

• Reply • Share ›

thirumalaikumar pappaiah • 3 months ago

This is really awesome !!!

• Reply • Share ›

Anu • 4 months ago

Simply awesome !!!!!!!!!!!!

Easy understanding !!!!!!!!!!!!!

• Reply • Share ›

Hafiz Harron Ejaz • 4 months ago

This i s really helpful :) Thank You very Much !

• Reply • Share ›

Jacopo Bontà • 4 months ago

Thanks! Very useful!

• Reply • Share ›

sicnarfngo • 4 months ago

amazing. just amazing. much wow. such awesome.

• Reply • Share ›

Roger Albino • 4 months ago

Muito obrigado. ;)

• Reply • Share ›

mills • 4 months ago

gitkrakenn found on http://www.becomeonewiththecod...

• Reply • Share ›

Buhraz Bolgo • 5 months ago

Any git clients that happen to run on Linux? maybe ... I know OSX and windows are the only two operating systems in the world, but just in case they are not, maybe a a couple of Linux gui clients, for KDE or GTK would have balanced the offering a little.

• Reply • Share ›

Igor Moura → Buhraz Bolgo • 5 months ago

I've used gitkraken and liked it so far.

1 • Reply • Share ›

Buhraz Bolgo → Igor Moura • 5 months ago

then i found GitEye from CollabNet.

Much lighter on cpu, and does seem to have all the functionality. Very professional looking and feeling plus you can click and open the files in your editor, something gitkraken was just not going to do, GitEye is running now on my desktop, and i can still type normally, something i couldn't say with gitkraken running.

• Reply • Share ›

Igor Moura → Buhraz Bolgo • 5 months ago

Interesting, I didn't have any problem CPU-related with GitKraken (although I have to admit that I use the terminal more far often than GitKraken itself).
I'm glad that you found a nice alternative though :)

• Reply • Share ›

Buhraz Bolgo → Igor Moura • 5 months ago

hoping this will spur a debugging session, the cpu load was coming from X not GitKraken directly, something GitKraken was making X do caused a significant CPU load

• Reply • Share ›

Buhraz Bolgo → Igor Moura • 5 months ago

Fount ot to be nice, liked the ability to do the push and pull the staging and nu-staging and the commit with comment .. nice graph. But i can't use it. It is simply way too heavy on the cpu, consumes 30% of a quad xeon, you have to be kidding me. it literally chewed up my system to the point where i had to wait for mouse clicks. gitkraken need more work for sure, so i went hunting...

• Reply • Share ›

Load more comments