git - the simple guide

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

Tweet

download the cheat sheet now. it's free!

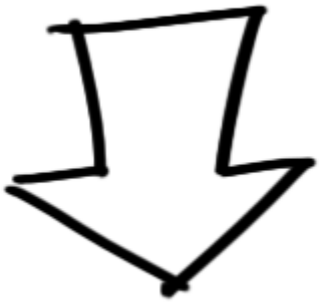
by Roger Dudler

credits to @tfnico, @fhd and Namics

in deutsch, español, français, indonesian, italiano, nederlands, polski, português, русски မြန်မာ, 日本語, 中文, 한국어 Vietnamese

please report issues on github

want a simple but powerful git client for your mac?



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.

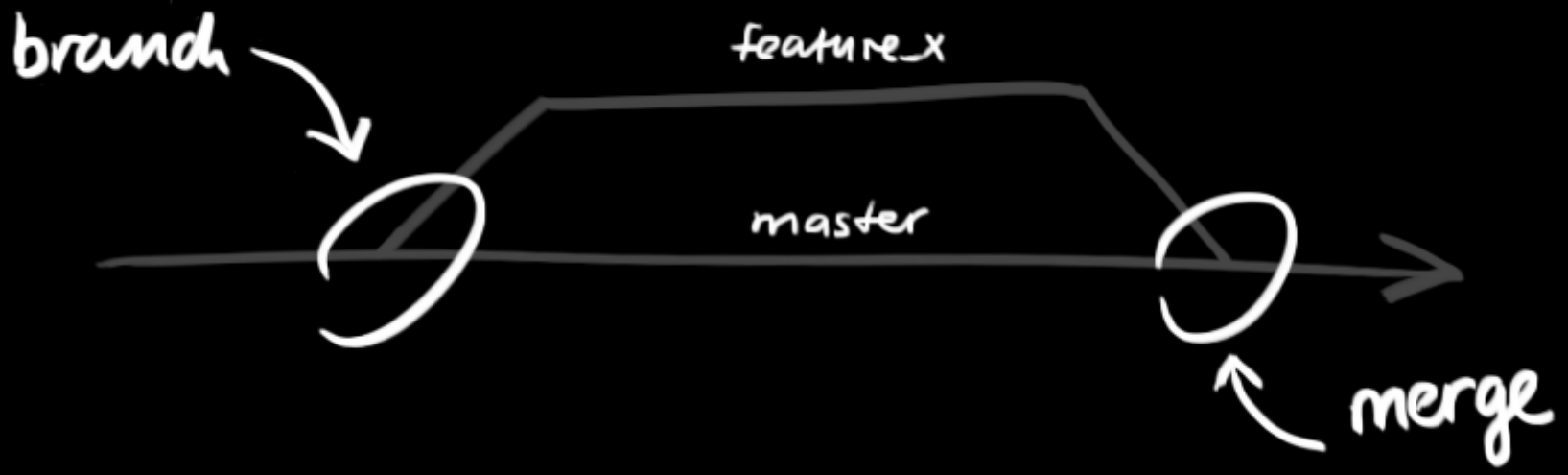
If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

```
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

837 Comments git - the simple guide 1 Login

Recommend 391 Share Sort by Newest

Join the discussion…

Bright Tsai • a day ago

Thanks.

Reply • Share ›

Asim Ali • 2 days ago

Thanks. Very well explained

Reply • Share ›

Jamiel Sharief • 2 days ago

Thanks for this guide, it is awesome.

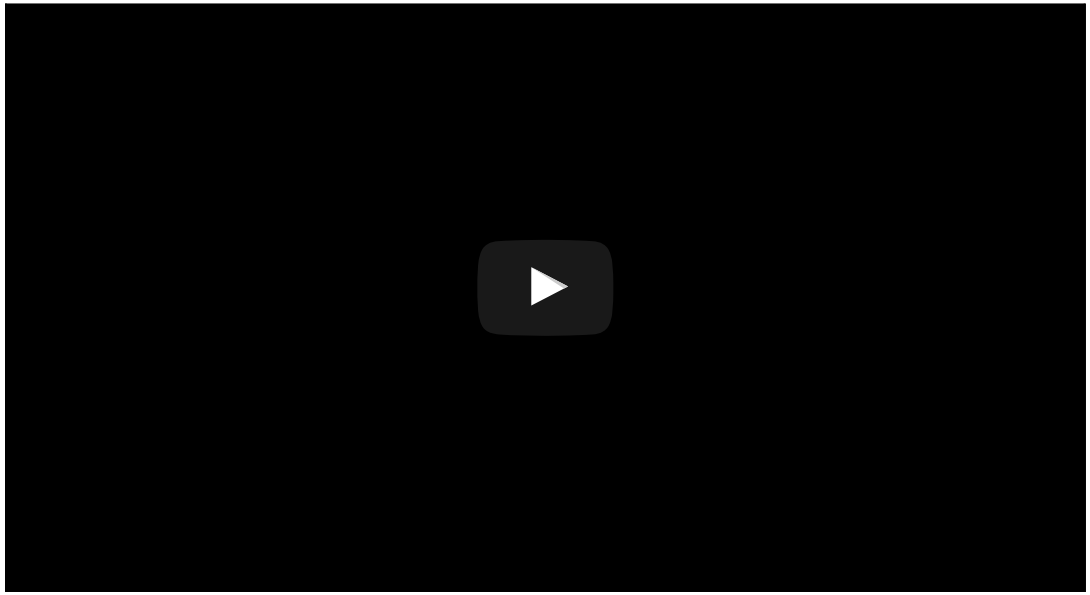
Reply • Share ›

fridayana baabullah • 3 days ago

This is great article. I love how the Roger explain in simple way to understand especially for beginners.

I create video version of this article, hope you enjoy as this :)

https://www.youtube.com/watch?...



Reply • Share ›

Ren • 5 days ago

Best f**&*(tutorial period!! thank you for this:)

Reply • Share ›

Allan K Preston • 5 days ago

You covered everything, except the one thing that most beginners want git for. To get a source tree from a github repository so it can be compiled.

1 Reply • Share ›

jay glass • 8 days ago

Great Post, thank you!

Reply • Share ›

Devendra Acharya • 12 days ago

Very useful information, also organized in a very awesome manner.

1 Reply • Share ›

sasi kumar • 15 days ago

what i understood is .. it is very similar to SVN ... with few differences. am i correct ?

Reply • Share ›

Marco Paganini ➚ sasi kumar • 14 days ago

Hmm, not really. :) git is a version control system, just like SVN, but *far* more sophisticated. The guide here is a good start and will allow you to do the basics (actually, most of what you need). Git offers way more. I'd suggest reading the Git Boot (https://git-scm.com/book/en/v2... to fully understand git.

Reply • Share ›

MoniqueRene • 16 days ago

Thank you!! The 5 hour GIT training wasn't doing it . .

Reply • Share ›

Ahsas Sharma • 16 days ago

Thanks a lot. This is a brilliantly designed no-nonsene guide to Git basics. Good job!

1 Reply • Share ›

DePersona • 22 days ago

Great piece for a jump start, for a more graphical illustration see http://marklodato.github.io/vi...

Reply • Share ›

baba blacksheep • a month ago

but u forgot to mention concept of head in branching

Reply • Share ›

Abdollah • a month ago

Thanks very much , great, simple, quick and very helpful guide!

Reply • Share ›

Ron Royston • a month ago

Looks great! Thanks Roger!

Reply • Share ›

Anupam Jain • a month ago

Super! Thanks for putting it down so neatly and cleanly. You also have a good eye for good design

Reply • Share ›

xgqfrms • a month ago

git for every times coding!

Reply • Share ›

Shyam Vashista • 2 months ago

I need to keep changes on local branch like connection strings, when I am taking latest from server (pull) it is not working asking first need to commit those changes. How can i keep local changes and take only committed code from server?

Reply • Share ›

abhigyan chatterjee ➚ Shyam Vashista • a month ago

Do you want to commit the code? If yes go ahead, if no then use git stash. This will put your uncommitted files in a temporary space and allow you to pull the code from repo. To use the stashed code again, use stash pop, this will restore all the files back to sanity. I aslo urge you to take a look at stash pop parameters in git documentation.

Reply • Share ›

Saqib Bilal ➚ Shyam Vashista • a month ago

I am a beginner at git myself but i think u need to use stash.
=>git stash

=>git pull
=>git stash apply or => git stash apply --index (also preserves old

commits)

=>git stash drop

Do make sure if this is the right thing to do... as i am a newbie myself

Reply • Share ›

hulesh chandra • 2 months ago

thanks a lot for such a wonderful and concise explaination

Reply • Share ›

Rishikesh Agrawani • 2 months ago

Nice explanation of Git.

1 Reply • Share ›

David Andersson • 2 months ago

This i a realy great and helpful guide. May I suggest you add a section for git rebase as well?

1 Reply • Share ›

Ravi Kumar Chintalapudi • 2 months ago

nice and helpful..keep it up..

Reply • Share ›

Rajat Kumar • 2 months ago

Really Helpfull thank you so much

Reply • Share ›

Ashvini Chauhan • 2 months ago

Thanks for guidence... Really amazing...

Reply • Share ›

Oren • 2 months ago

Great guide, good title!!!

Reply • Share ›

faris • 2 months ago

Thanks! sometime i forget the syntax and just refreshing my memory by coming here..

1 Reply • Share ›

ben • 2 months ago

fantastic guide! i've been looking for a guide like this!

Reply • Share ›

Antonio Joseph Smith • 2 months ago

This is a lifesaver. I have been mixing up these steps all week. Thank you.!!!

Reply • Share ›

Tadeus Araújo • 2 months 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 • 3 months 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 • 3 months ago

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

1 Reply • Share ›

Stork • 3 months 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 · 3 months ago

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

· Reply · Share ›

manikant ➚ englishextra · a month ago

same

· Reply · Share ›

Pankaj Kolhe · 3 months ago

awsome man....Really amazing guideline

· Reply · Share ›

Amit Chaudhary · 3 months ago

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

4 · Reply · Share ›

Sudipto Roy · 4 months 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 · 4 months ago

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

· Reply · Share ›

Anant Prajapati · 4 months ago

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

· Reply · Share ›

Sunny Kusawa · 4 months ago

This is what I looking for. Thanks

Reply • Share ›

Raj Singh • 4 months ago

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

1 Reply • Share ›

fuecisla ➚ Raj Singh • 3 months 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 • 4 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 Reply • Share ›

Jean • 4 months ago

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

Reply • Share ›

Mohammad Sharaf Ali • 4 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!

guide and cleared all my concepts. Super great work man!

Reply • Share ›

Miguel • 4 months ago

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

Reply • Share ›

KaTIOWa • 4 months ago

Super clear!

Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Add Disqus Add Privacy DISQUS