

git - the simple guide

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

Tweet

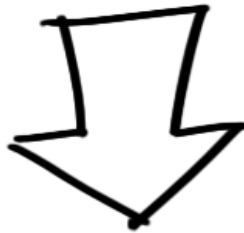
by Roger Dudler

credits to @tfnico, @fhd and Namics

de in deutsch, español, français, indonesian, italiano, nederlands, polski, português, русский,

မြန်မာ, 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

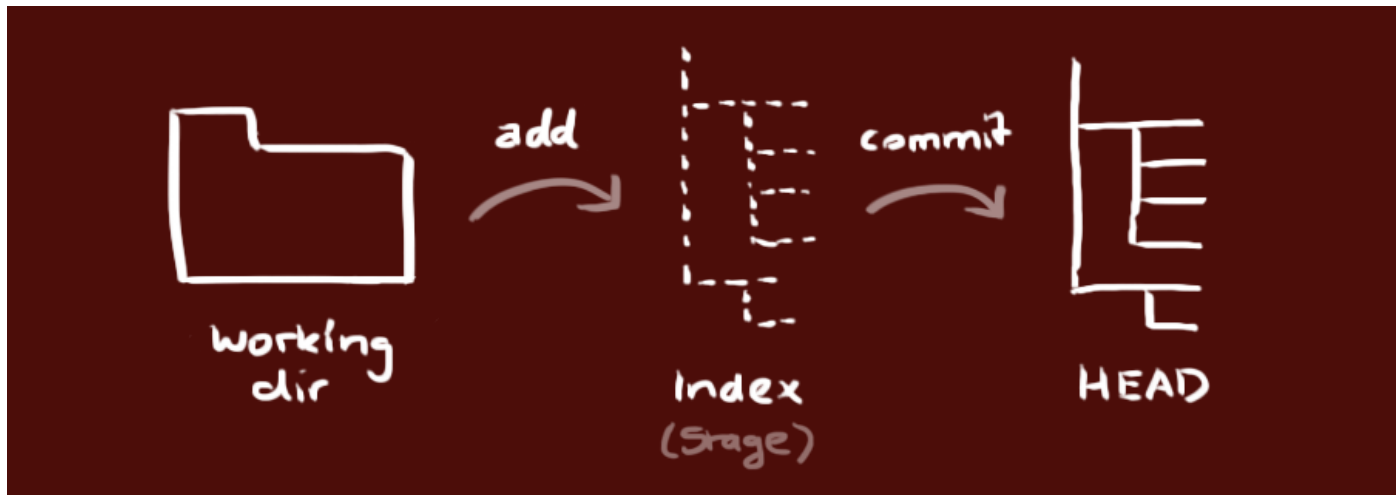
```
git clone /path/to/repository
```

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.

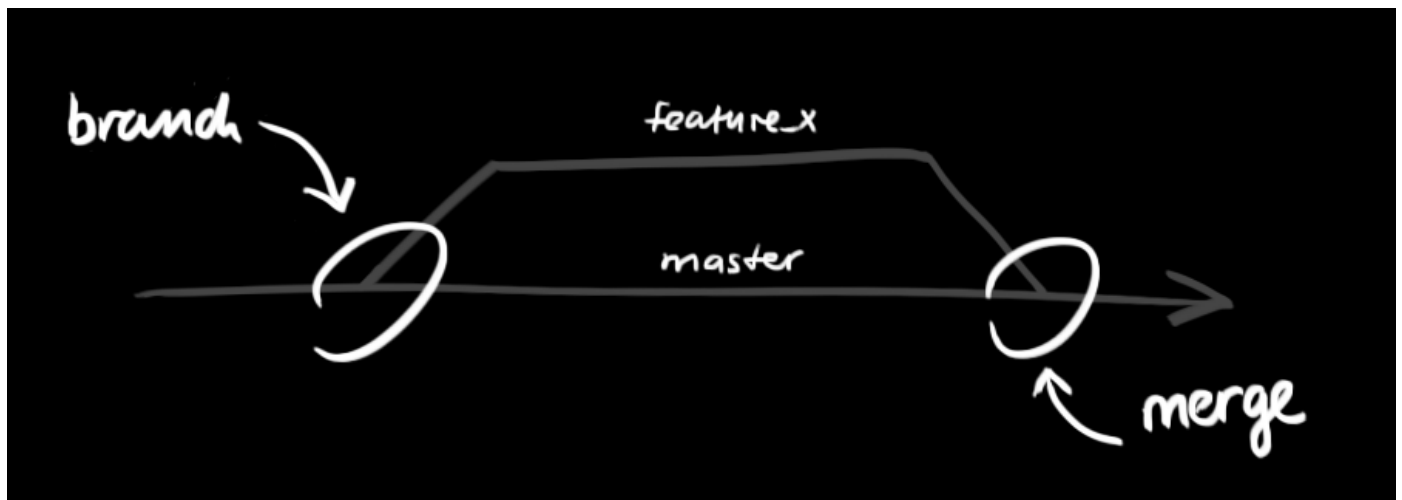
If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

```
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

969 Comments

git - the simple guide

 Login ▾

 Recommend 488

 Share

Sort by Newest ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

**bertenvdb** • 2 days ago

Over the years I tried so many git gui's and always returned to CLI, until I found GitKraken! Free for Win+Mac+Linux <https://www.gitkraken.com/>

^ | v • Reply • Share ›

**Deepak Kumar Singh** • 5 days ago

Simple and easy. No Deep shit!!!

1 ^ | v • Reply • Share ›

**Rodolfo Contreras** • 5 days ago

Best web ever!!!

^ | v • Reply • Share ›

**Nate Maher** • 11 days ago

Cool! Very helpful starting my new job!

^ | v • Reply • Share ›

**Ahmed Magdi** • 13 days ago

Awesome work :D

^ | v • Reply • Share ›

**Mohneesh Sreegirisetty** • 13 days ago

No Deep Shit! thank you for this awesome tutorial...

^ | v • Reply • Share ›

**Emily Kauffman** • 18 days ago

This is great!

^ | v • Reply • Share ›

**Vikas Almal** • 23 days ago

Loved your website!!

^ | v • Reply • Share ›

**Sreekaanth Ganesan** • 24 days ago

Very informative!!

^ | v • Reply • Share ›

**ernestas** • a month ago

Nice one!

^ | v • Reply • Share ›

**Sushant Todkar** • a month ago

Great information to learn GIT

^ | v • Reply • Share ›

**Gopi** • a month ago

Nice information for the beginners. Thanks for the creation.

1 ^ | v • Reply • Share ›

**Rhodimos Okon** • a month ago

Wao this is great well done and thanks in a million

^ | v • Reply • Share ›

**Djordje Arsenovic** • a month ago

Well fucking done, thanks! :)

^ | v • Reply • Share ›

**Will Ashworth** • a month ago

Love this guide. Just shared it with everyone :)

^ | v • Reply • Share ›

**Abhineet Raj** • 2 months ago

Excellent work with the guide !!

^ | v • Reply • Share ›

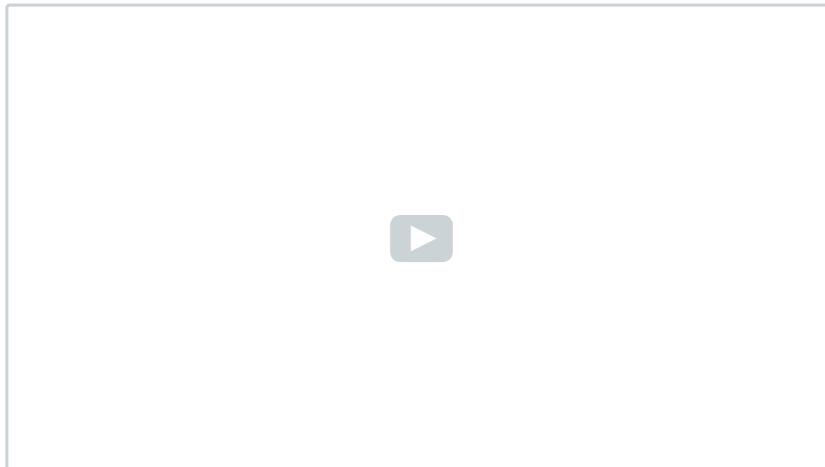
**Tri Nguyen** • 2 months ago

Aug 2017 and this is still useful. Thank you so much, man!

1 ^ | v • Reply • Share ›

**Igor** • 2 months ago

Uhhh, This is.. a good guide! :D



^ | v • Reply • Share ›

**wSafayat Jamil** • 2 months ago

Thanks man, Really appreciate it. (From Bangladesh)

^ | v • Reply • Share ›

**Keith Kibler** • 2 months ago

How about adding a "git stash" and "git stash pop" section? That way I only need a single web page for 98% of the git stuff I do? Please?

Excellent resource! Thanks!

^ | v • Reply • Share ›

**Rob** • 2 months ago

Very usefull !

^ | v • Reply • Share ›

**GOKUL Kathiresn** • 2 months ago

A most needed explanation in the best way

^ | v • Reply • Share ›

**victor Caballero** • 2 months ago

simple and excellent !

1 ^ | v • Reply • Share ›

**Joshua Eirman** • 2 months ago

Real nice artistic incorporation!

1 ^ | v • Reply • Share ›

**Joshua Eirman** • 2 months ago



Is this an appropriate place to ask why there is a remote master and a local master branch? I can't find a place to ask, please help.

Thanks,
Josh

^ | v • Reply • Share ›



Joshua Eirman → Joshua Eirman • 2 months ago

deleted

^ | v • Reply • Share ›



Brian Lee → Joshua Eirman • 2 months ago

Git is a distributed version control system. This means each repo is self-contained and operate independently. The remote itself is a repo and your local copy is also it's own repo each tracking changes independently. After you update and create commits on master in your local repo you can use `push` to apply those same commits to a remote repo (usually named origin because that's where it was originally cloned from) and apply them on the remote repo's copy of master.

You technically don't need to have a local branch named master and a branch named master on the remote. You could name them something different, but that would be confusing. The convention is to name branches the same across cloned repos.

1 ^ | v • Reply • Share ›



Joshua Eirman → Brian Lee • 2 months ago

Thank you Brian Lee, I am just finishing my immediate studies on GitHub and I had found this site.

^ | v • Reply • Share ›



GitCoder • 3 months ago

Excellent compilation. As a suggestion, please add TortoiseGit to the list of Git clients. It's one of the best, oldest, and doesn't ask you to "create an account" to use.

2 ^ | v • Reply • Share ›



Paras Gandhi • 3 months ago

great stuff

1 ^ | v • Reply • Share ›



Sovichea Cheth • 3 months ago

This is very helpful!

^ | v • Reply • Share ›



BlackYeti • 3 months ago

That was grate, thanks!

^ | v • Reply • Share ›



ybl → BlackYeti • 3 months ago

I too love to grate my commits to bite size!

^ | v • Reply • Share ›



Christopher John • 3 months ago

This is awesome

^ | v • Reply • Share ›



Mova Club • 3 months ago

a good starting point to learn git - many thanks

^ | v • Reply • Share ›

**Rami Bakry** • 3 months ago

helpful one ... thanks

^ | v • Reply • Share >

**Alucaard** • 3 months ago

I'm in deep shit

1 ^ | v • Reply • Share >

**Cees Timmerman** → Alucaard • 3 months ago

Why? Here's a sensible branching model that balances between the heavy GitFlow and light GitHub Flow, it's called GitLab Flow:

<https://about.gitlab.com/20...>

1 ^ | v • Reply • Share >

**Alucaard** → Cees Timmerman • 3 months ago

thanks that helps much

1 ^ | v • Reply • Share >

**Sue Maxon** • 3 months ago

Great explanation. Very simple.

1 ^ | v • Reply • Share >

**Empty** • 3 months ago

All you need is one day of reading commands and contribute once in awhile in GitHub so you will not forget it.

^ | v • Reply • Share >

**DOS User** • 3 months ago

Git is to go back to DOS days?

^ | v • Reply • Share >

**Cees Timmerman** → DOS User • 3 months ago

People who prefer typing to clicking prefer a terminal (command line) to a GUI like SourceTree or TortoiseGit.

^ | v • Reply • Share >

**Gabriel Pulido** • 4 months ago

this gives me a headache

^ | v • Reply • Share >

**Cees Timmerman** → Gabriel Pulido • 3 months ago

Why? It's just like CVS and SVN, except that people can use it without an internet connection or shared central repo.

^ | v • Reply • Share >

**Cees Timmerman** • 4 months ago

To update all of a repo's submodules:

```
repo ct$ git submodule update --init --recursive
```

To add one to an existing repo, e.g.:

```
repo ct$ git submodule add -f https://slsapp.com/git/comp...
```

^ | v • Reply • Share >

**R.k.** • 4 months ago

Awesome... thank you

^ | v • Reply • Share >

**Joseph Coombe** • 4 months ago

I can never remember these commands...

This is very helpful. Thank you!

^ | v • Reply • Share ›

**Tim** • 4 months ago

there's no mention ever using 'git status' :(but otherwise. VERY good intro to Git

1 ^ | v • Reply • Share ›

**azerazer** • 4 months ago

azeazeaze

^ | v • Reply • Share ›

[Load more comments](#)[✉ Subscribe](#) [D Add Disqus to your site](#)[Add Disqus](#)[Add](#) [🔒 Privacy](#)