git - the simple guide

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

Tweet

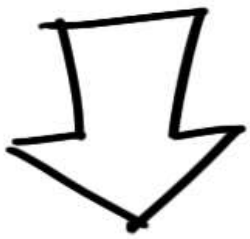
by Roger Dudler

credits to @tfnico, @fhd and Namics

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

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

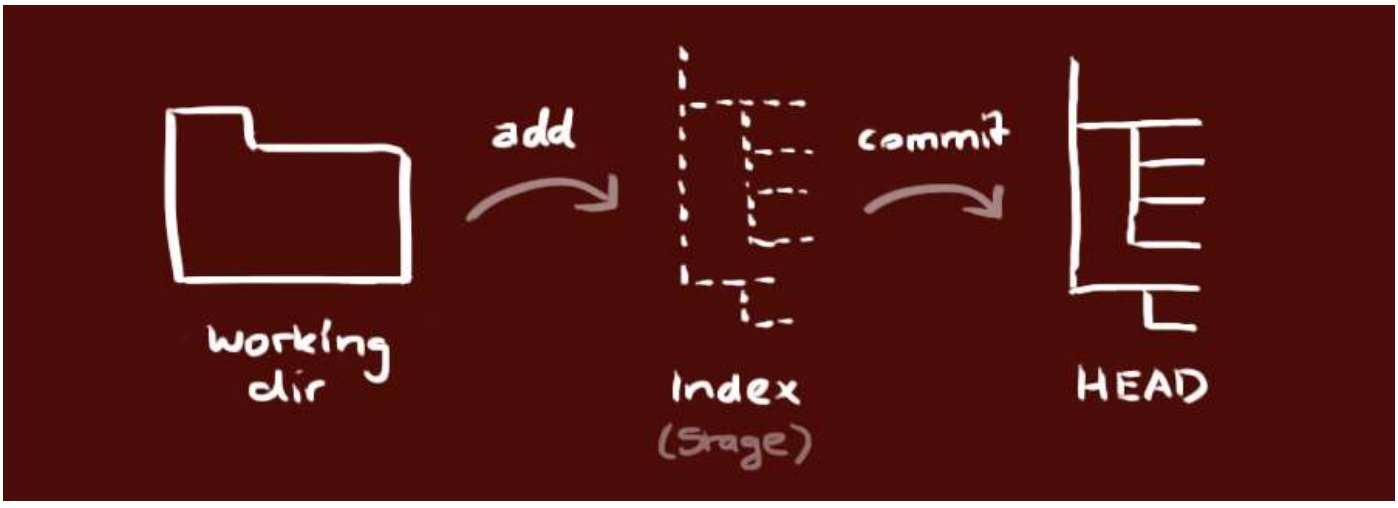
```
git clone /path/to/repository
```

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.

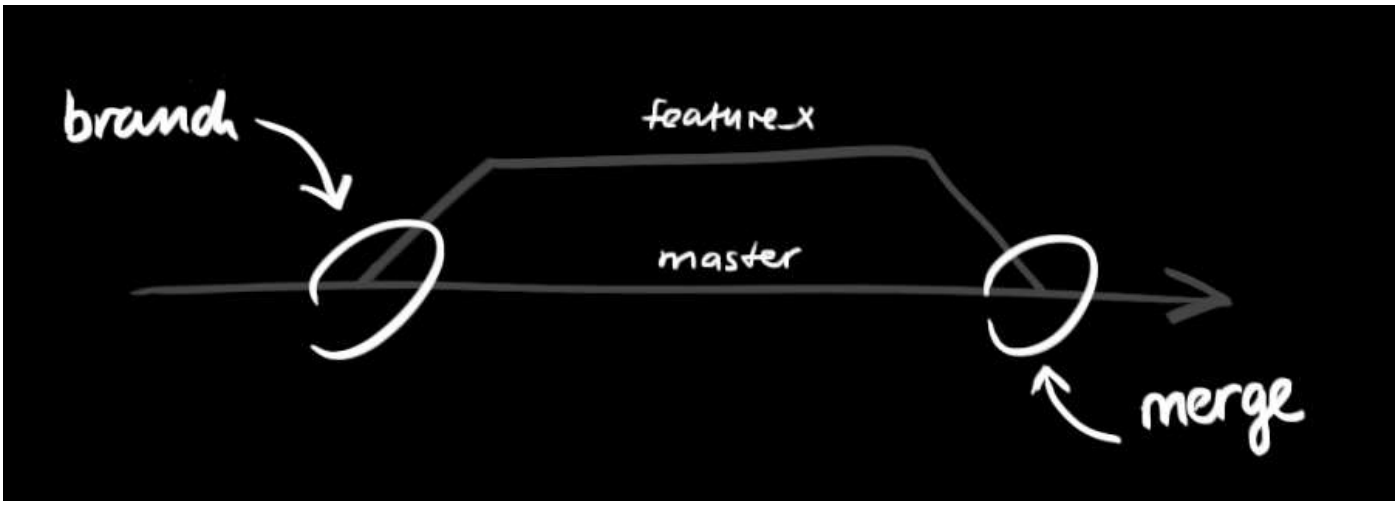
If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

```
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

comments

109 Comments

Login ▾

G

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

♡ 80 Share

Best Newest Oldest

Alan Knipmeyer
2 months ago

funny and useful, i'm going to be recommending this one for sure

0 0 Reply • Share ›

M

Marie
2 months ago

Thank you! The best guide ever!

0 0 Reply • Share ›

H

Николай Каретников
8 months ago edited

Thank you! Great info! Consider the simpliest example. I have a working directory, committed it the the head. Head now points to that commit. Next, I remove a file from the working directory, realize that it was a mistake. How to fix the error using git?

0 0 Reply • Share ›

M

Michael Alexander
8 months ago

Brilliant. I'm scratching my head in a few places, but overall well done! "Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away"--Antoine de Saint-Exupéry.

1 0 Reply • Share ›

L

Leonardo Salgado Aguayo
a year ago

I started years ago with this, and always send this guide to new devs looking for a place to start understanding git. So I have to say Thanks, in my name and all my friend who started here. **We love U.**

0 0 Reply • Share ›

F

Flummi
a year ago

Great reading! Thank you!

0 0 Reply • Share ›

R

Raisa Kulakovska
a year ago

Thanks a lot, the best article I've ever read about git.

1 0 Reply • Share ›

Ignatius Reilly
2 years ago

Useless. Only half-explains what is happening to new people.

2 8 Reply • Share ›

Jeff K Ignatius Reilly
2 years ago

Lol, you're in the wrong place buddy.

0 0 Reply • Share ›

adolf garlic Jeff K
a year ago

No, they're not, the explanations are not clear enough. e.g. 'index acts as a staging area'. Why is this required/needed? How is it used? Why can't I just commit directly to the main repository? etc
It's like a "guide for people who already know how it works and just want a bit of a reminder".

0 0 Reply • Share ›

AlejandroZ adolf garlic
a year ago edited

And that's why it's called the simple guide.
Were you expecting to read the full guide? Then it would've been named "the full/hard/complex guide".

Although some brief words with examples explaining why or when you would use those commands are welcome.

0 0 Reply • Share ›

Cesar
2 years ago edited

Nope. Git still seems as a good example of how to make something simple needlesly complicated. It's ironic that everyone is using it to build things that should be very simple to use as software should be. But your guide is great though!

7 1 Reply • Share ›

R3 Cesar
a year ago

I don't think you understand the need of version control systems.

0 1 Reply • Share ›

Cesar R3
a year ago

And I dont' think you really understand English... guess learning Git doesn't leave time for that.

I've been using version controls systems for 2 decades. The Git version control system is simply a very confusing powerful software.

3 0 Reply • Share ›

Andrii Kuplevakhskyi → Cesar

a year ago

Can you give an example of what you think a simple version control system is?

0 0 Reply • Share ›

C

Cesar → Andrii Kuplevakhskyi

a year ago edited

TFS Version Contorl and others...
Doesn't need more than a few hours to master (in the worst case) and is enough for most small to medium projects.

Meanwhile, with git, we had to bring someone to give a 10 hour extended git tutorial and that's after working with it for weeks and reading many tutorials. There must be a better solution!

Also, with git, I still haven't found an easy way to view a file's history (which is half of its role) that doesn't show IRRELEVANT commits ("blame" is not file history), the way TFS VC shows file history for example in a very clear and quick way, with no need for more third party tools.

Yes, you get branches which are very cool, but you also get the branch hell.

Why did I move ? because I was kind of forced to do it.

1 0 Reply • Share ›

H

Николай Каретников → Cesar

8 months ago

git was created by geeks that didn't want anyone to understand what they do.
Glad there are other, more simple tools

0 0 Reply • Share ›

MatthieuScarset → Cesar

a year ago

Install VSCode and you can see a file's history
Add GitLens extension for more advanced filtering options.

If your team has money, you can use something like GitKraken.

Enjoy git again!

0 0 Reply • Share ›

H

Hans → Cesar

a year ago

Tortoise git can do a pretty good file history

0 0 Reply • Share ›

Amias

2 years ago

Best beginner guide I could find. I think I kind of understand how git works now, which is a lot better than before.

4 0 Reply • Share ›

Wallace Espindola

2 years ago

Excellent! Thank you for the great and concise knowledge shared!

3 0 Reply • Share ›

Josh Blaylock

2 years ago

Nice layout with concise explanations!

2 0 Reply • Share ›

www.pcconsultingasia.com

2 years ago

Thank you at last a great logical guide for us that only use GIT occasionally. Great

1 0 Reply • Share ›

wale adeniji

3 years ago

This is the simplest, yet explicit git walkthrough I've seen in recent times.

2 0 Reply • Share ›

James Carlson

3 years ago

More docs should be like this.

3 0 Reply • Share ›

S

Suryaa Jha

3 years ago

definitely this is a must guide for any begineer

1 0 Reply • Share ›

Siddharth Kothari

3 years ago

Superb work, thank you for this!

0 0 Reply • Share ›

Pay the Bill Motto

3 years ago

you nail it brother , thank you for this , that is what I'm looking for no deep shit .

2 0 Reply • Share ›

Jitendra Patil

3 years ago

no-nonsense guide to Git!

0 0 Reply • Share ›

D

Doug Kimzey

3 years ago

This is good work!

The greatest impediment to git is the syntax. The architecture or git is not too bad.
The commands used in git are ambiguous, vague, and convey no sense of source and direction. Since the documentation for one confusing git command is done in terms of other bewildering commands, the official git documentation is an obstacle. The actual git documentation provides examples of bad writing practices for seminars in Simplified Technical English. Command sets for tools like git should follow the guidelines of Simplified Technical English. These guidelines maximize clarity. If these guidelines had been followed, git would have a shallow learning curve.

0 0 Reply • Share ›

Jocimar Lopes

3 years ago

YES! thank you, thank you! This is THE git-cheat-sheet-no-deep-shit I was looking for. By the way, awesome design :))

1 0 Reply • Share ›

José Silva

3 years ago

Thanks a lot! Very useful, no deep shit :)

0 0 Reply • Share ›

S

Shah Bhuiyan

3 years ago

super dope and well designed git guide indeed!!!! love the graphic design and illustrations as well! **no deep shit!**

0 0 Reply • Share ›

AB

Aleksejs Birula

3 years ago

Thank you for "useful hints"

0 0 Reply • Share ›

TonyMath

3 years ago

Excellent tutorial - just what I needed.
I have a question. Suppose there's a file in the remote repository (myfile.txt), created online, so not on the local repository, and I just want to merge that one file only to local.
How do I do that? Can I do a "git fetch" on a single file?

0 0 Reply • Share ›

Ahmad Yasser ➚ TonyMath

3 years ago

you need to make "pull" first

0 0 Reply • Share ›

Ihwan ID

3 years ago

Nice sharing.

0 0 Reply • Share ›

Shimmy

3 years ago

Thanks for sharing

0 0 Reply • Share ›

gopinath_mb

3 years ago

What a simple tutorial...Great Work..Thanks alot :)

0 0 Reply • Share ›

paulwratt

3 years ago

yeah same, "best bloody git quick guide ever"

3 0 Reply • Share ›

G

Glafe

3 years ago

I always return to this guide whenever i want to teach someone git (or myself again lol)

3 0 Reply • Share ›

shaunkahler

3 years ago

I farted

1 9 Reply • Share ›

Greg Trevellick

3 years ago

Spot on - nil sh1t! :-)

1 0 Reply • Share ›

C

Cassie Huang

3 years ago

Finally I found a very useful git guide! Thanks for sharing!

1 0 Reply • Share ›

RRCampbell

3 years ago

You make lots of assumptions here about people's knowledge. I wouldn't call this something for a novice.

2 5 Reply • Share ›

Sithira Munasinghe

3 years ago

On point. No Deep shit ! <3

3 0 Reply • Share ›

Pablo Fs

3 years ago

Finally someone doing what's needed!! Thanks so much for this!

0 0 Reply • Share ›

V

Valeriy Ivanov

3 years ago

Thank you! Great guide!

Thank you! Great guide!

0 0 Reply • Share ›

ELuvixiLY

3 years ago

This is the best git guide I've ever read.
Thank you!

1 0 Reply • Share ›

S

Stylzsoserious

3 years ago

Bruh ... really appreciate this guide. keep up the good work.

0 0 Reply • Share ›

Load more comments