git - the simple guide

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

Tweet

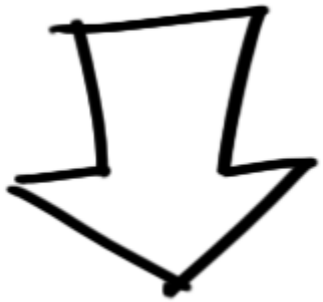
by Roger Dudler

credits to @tfnico, @fhd and Namics

uide in deutsch, español, français, indonesian, italiano, nederlands, polski, português, русский, tü

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

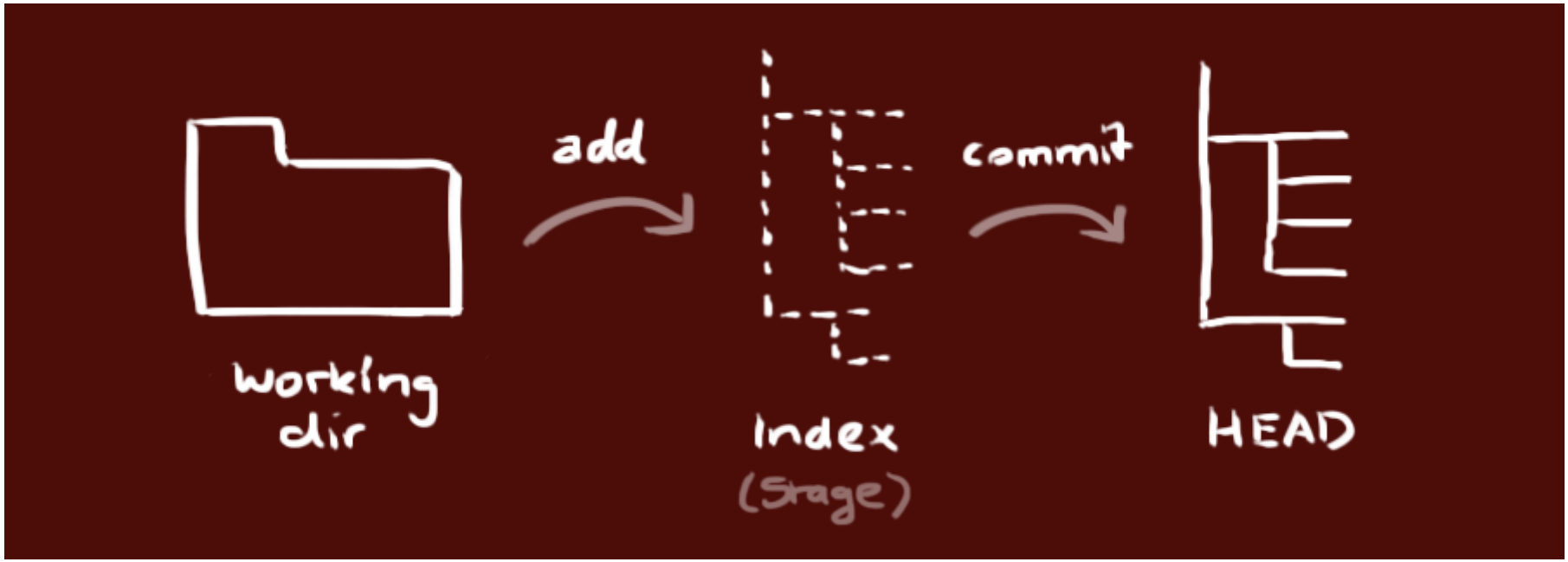
```
git clone /path/to/repository
```

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.

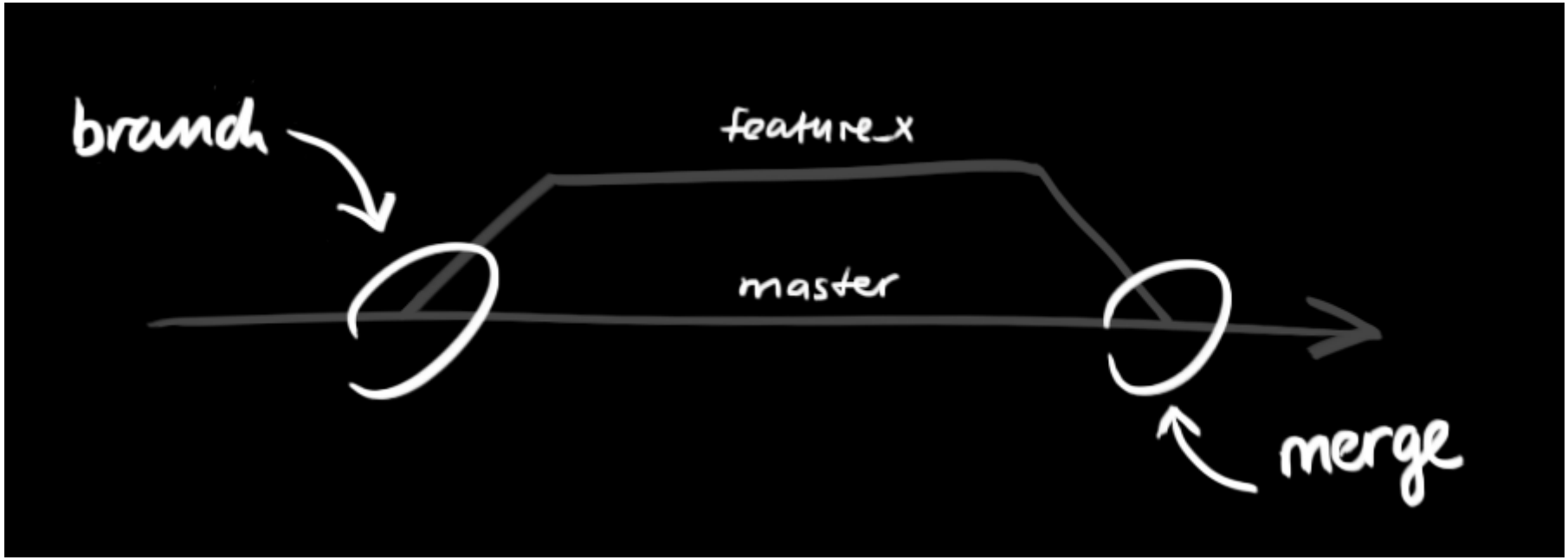
If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with

```
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

LOG IN WITH

OR SIGN UP WITH DISQUS (?)

Name

knows1 • 3 months ago

. بهذ نم درس الع فارك ش

Reply • Share ›

Marek Dyjak • 7 months ago

Szerwus, it's a me, Marek. I bardzo like the git thing I like doing frontend css style in it, very helps.
Thanks!

Reply • Share ›

Dimitri Smolov ➚ Marek Dyjak • a month ago

Szerwus the Marek, ju ar bardzo klewer front ent maderfaker. Do you like doing CSS dogi stajl? Before you type anything, brush your cycki carefully.

Reply • Share ›

Scott Taylor • 7 months ago

This helps with both the concepts and actual usage. what I really need now is a simple guide on how to add Git to an existing development environment. I am trying to use Bitbucket Server and Sourcetree to do this, with Git for Windows on the existing servers. I am close to a complete solution, but I can't get developers' changes back to the folders of existing projects, so developers can use our existing test environment to check their work. It's kind of like "The Five Blind Men and the Elephant". I search the internet, and find lots of bits and pieces, but no one, complete guide.

Reply • Share ›

Misaki • 9 months ago

This is "the most simpliest guide i ever seen" :) happy to read it, thanks!

Reply • Share ›

Chris Kenst • 9 months ago

As someone who knows / works with git regularly this is a nice reference guide. Can't speak to someone who is new and trying to use it but I'll recommend people check it out.

1 ^ | v • Reply • Share ›

Andrej Trusevic • 9 months ago

Nice Job :)

^ | v • Reply • Share ›

mommy • 9 months ago

great post!!11!22!!!!!11!!!

^ | v • Reply • Share ›

Eric Nguyen • 9 months ago

Very nice write-up. One thing I would add is how to delete a branch on server from CLI.

^ | v • Reply • Share ›

Tony Mannucci • 10 months ago

Problem: this treatment does not state how to interact with github, which I imagine would be a standard practice for many using git. I suppose github is a repository, but what is the username@host:/path/to/repository for my git repository?

^ | v • Reply • Share ›

scitronboy ➚ Tony Mannucci • 10 months ago • edited

GitHub is used for storing your local git repo online. After you create a github repository on github.com, you can download it to your computer with `git clone https://github.com/your-username/your-repo-name` Or else you can connect the remote repo to a local repo with `git remote add origin https://github.com/your-username/your-repo-name` You can find more info in my tutorial at https://scitronboy.github.i... or other github

tutorials which you can easily find by searching google.

Reply • Share ›

Tony Mannucci ➚ scitronboy • 9 months ago

Thanks!

Reply • Share ›

scitronboy • 10 months ago • edited

I wrote an article about git that has most of the stuff in this article. It's not as fun and easy to read, and it's much longer and maybe a bit complicated, but it lists a few more important ideas/etc. about some of the commands listed in this article:

WHY AND HOW TO USE VERSION CONTROL: A SIMPLE GIT AND GITHUB TUTORIAL

by the way, this is a fantastic cheatsheet/article about git!

Reply • Share ›

Tony Mannucci ➚ scitronboy • 10 months ago

This seems to discuss how to work with github! Thank you.

Reply • Share ›

scitronboy ➚ Tony Mannucci • 10 months ago

I'm glad it was helpful!

Reply • Share ›

venkateswara • 10 months ago

Thanks a lot!

Reply • Share ›

Денис Гойда • 10 months ago

Thanks !

Reply • Share ›

Mansouri Habiba • 10 months ago

Thanks

Reply • Share ›

ExillustX • 10 months ago

Thanks!

Reply • Share ›

Daniel • 10 months ago

Hi great information. I have added https://github.com/Nyx0uf/q... and run it in git hub. How do i run the git so i can see the fonts? Very basic question i know.

Thanks

Danny

Reply • Share ›

commenter • 10 months ago

Great summary!, for another great interactive tutorial check out: https://learngitbranching.j...

Reply • Share ›

Dreamcatcher Sua Lipa 😎 💖 • a year ago

thank you!

Reply • Share ›

khoi • a year ago

awesome!

13 Reply • Share ›

Gsshankar Shankar • a year ago

Simply SUPER!!

Reply • Share ›

JoannaShaw • a year ago

JoannaShaw • a year ago

This is a fantastic starters guide to git, thanks a lot for taking the time :)

Reply • Share ›

eight • a year ago

In case you are not planning on taking advantage of the distributed nature of git, stansa (using svn) might be a good alternative: https://stansa.org

1 • Reply • Share ›

Techno Tard • a year ago • edited

Deer GAAWWDD, had I only found your site sooner! I've read and read and read only to find that your site had all I needed! I will like, subscribe or donate to any page you have!

Reply • Share ›

Michael Derosier • a year ago

Why cant technical guides always be this straightforward? The author should have a donate feature. I am extremely grateful - TY!

5 • Reply • Share ›

Rodolfo PEÑA • a year ago

How can I update Attom into specific commit

Reply • Share ›

Agustín • a year ago

Awesome! thank you very much!

Reply • Share ›

Sindhu Rajendran • a year ago

This is really The Clean and Simple Guide for GIT.
Thanks a bunch for this Great Job!

Reply • Share ›

Bahadir Balban • a year ago • edited

This is a great reference. Here is another one I wrote with some additional details on branching and merging yet not very long: https://getbuzz.io/c/gettin...

Reply • Share ›

Mark_Videon ➚ Bahadir Balban • a year ago

404

Reply • Share ›

Bahadir Balban ➚ Mark_Videon • a year ago

updated link: https://getbuzz.io/c/gettin...

also one on how to do a rebase: https://getbuzz.io/c/gettin...

1 Reply • Share ›

Zeb • a year ago

Super useful. Thanks!

Reply • Share ›

Kinjal • a year ago

This is very useful. Thank you for sharing

Reply • Share ›

Debbie Nelson • a year ago

Thank you for this...it's SO good!

Reply • Share ›

Oscar Crisanto • a year ago

This has cut my job in half.. thanks a lot...!

Reply • Share ›

Roger Hoem-Martinsen • a year ago

Great post! Thanks!

Reply • Share ›

Renzchler Oxiño • a year ago

Amazing!

Reply • Share ›

scitronboy • a year ago

It would be really great if you could make something just like this for some more advanced features of git!

Reply • Share ›

Vinicius Muniz de Melo • a year ago

A big thanks for this post, this help me a lot on the understanding of git and his tools.

1 Reply • Share ›

Trung Anh • a year ago

A big thumb up for this post. So short but also full of information

Reply • Share ›

Alex Rosenthal • a year ago

Soo. good. so so good.

1 Reply • Share ›

smart goose • a year ago

Thanks ... good consolidation and representation of important git cmds

Reply • Share ›

elescondite • a year ago

Superb. Well written and nice presentation makes it educational and a pleasure to read.

Reply • Share ›

marqusG • a year ago

Thank you for the superclear guide!!! Quick, easy and useful.:)