5/21/2025
12:18 AM

Learn Git.

Git full course for Beginners

(1)

free code camp.org

bash2.bat

```
@echo off
start "" "C:\Program Files\Git\git-bash.exe"
exit
```

C:\Windows

bash2.bat

add bash to path

```
git --version
cd learngit
mkdir gitone gittwo gitthree
```

or shift + Right Click → Open Git Bash here

these folders are not tracked they have to be initialized to be tracked.

author: Hitesh Choudhary.

Check status.

```
git status
```

fatal: not a git repository. (directory not being tracked)

```
cd gitone
ls -la   } shows all with hiddenfiles
git status   to check tracking status.
```

Youtub.com/@HiteshCodeLab

```
git init   } Run once per project
git status → on branch master
```

gitone (This is being tracked)

- git → a hidden folder to keep tracking of files and sub folders.

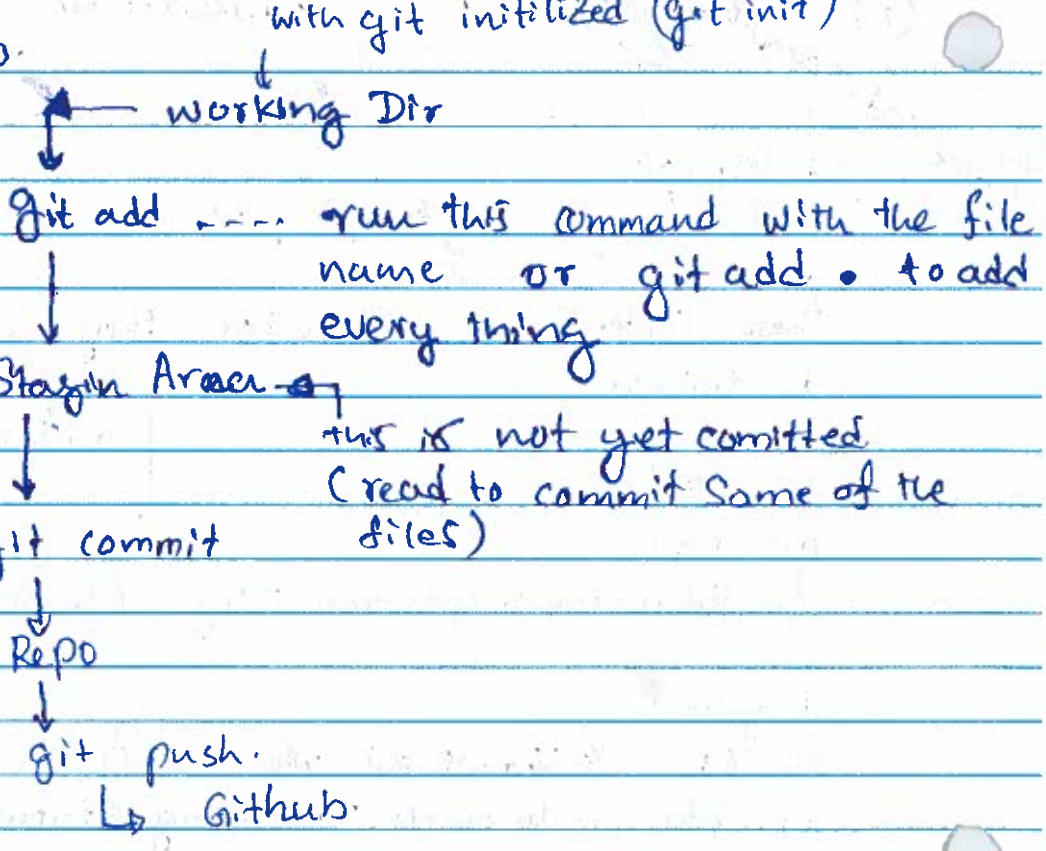
```
cd .git
ls
```

HEAD hook refs config ... } No manual changes should be made to .git folder

* Commit statements are like the check points
  Write → Add → Commit.

(2) 1:54 am
5/21/2025

Work Flow



To visualize on VSCode add the following extension
GitLens

touch test1.txt test2.txt

git status.
untracked files:
test1.txt. } these are not tracked this when
test2.txt } we are not yet done or we don't want them to be tracked

To track we need to add them.
git add . ← add all files (not recommended)

git add test1.txt ...test2 ... (multiple files can be added)
git status
new file: test1.txt (being tracked)
test2.txt (in red not tracked)

(3) 2:01 am
5/21/2025

(3)

when a file is added we are in the staging area.
Unstaging if needed.

git rm --cached <file>

To save a check point git commit., commit needs a message

git commit -m "add file one"
↑ message

adding file 2
git add test2.txt.
git status
git commit → opens Vim → add comment →
ESC :wq w: write q: quit. → ↳ Press enter.

Not its commited with another check point

git log

commit 996... (hash)
Author : Sandun <with...@gmail> } info was grabbed by a config file for name, email...
Date :.
message

git log --online
↑ gives a single line log

⇒ Atomic commit
One task at a time (one commit for one bug if you are fixing bugs)

(4) 2:36 Am
5/21/2025

commit message.
present tense - order command to code base
ex: code base add file test1.txt.

Set user name;

git config --global user.name "Mona Lisa"

git config --global user.name' Check whether name is Correct.

Setting the email.

git config --global user.email "vitharana1996@gmail.com

Changing default editor to VS Code;

git config --global core.editor "code --wait"
(vs code should be in Path)

Now it opens vs code and waits for the user to enter the message & close the window

git log | git log --oneline

⇒ .gitignore
this is a special file and it has to be created
touch .gitignore.

This is used to avoid sensitive files. such as the API keys from being included in the repo

(5) 12:41 am
22/05/2025

Sensitive data may be in a file called:
• env

```
MONGODBURI = test_url_Path
```

git status:
- env
- gitignore
- vscode/

when we have multiple files we would find it easy to do git add . ← add all
to remove certain files and folders from being tracked

```
• gitignore
• env
node_modules/
• vscode/
```
} folders to be tracked

git status now does not show .env

git add .
git status.
git commit → add the message and close vscode
git log

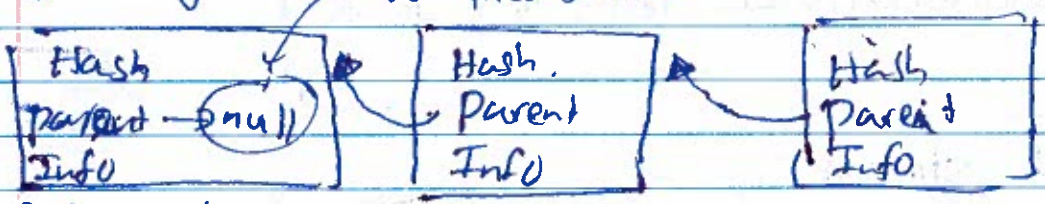
To generate common .gitignore file search online and select a relavant project (toptal.com)

Config settings are stored on a file
cd ~
cat .gitConfig. → name =
email =
editor = code --wait

⑥ 1:01
3/22/2025

How commits work

git log --oneline



first commit

Every commit is dependent upon the previous commit except the first commit.

cd .git/
.hooks/ ← allows to control pre-push / pre-rebase ....

Branches

mkdir git 2
git status
git init
git status → on branch master

touch index.html
git add index.html
git commit -m "add index file"

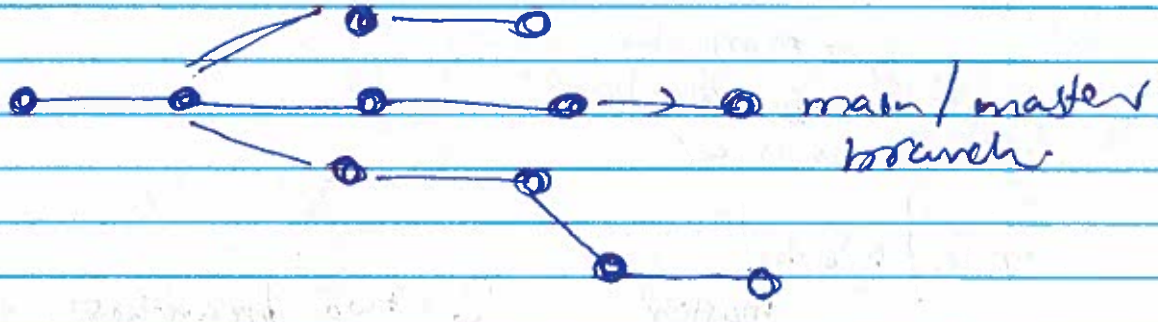
in vscode EXPLORER cannot see the .git folder
Ctrl+, → scroll the settings → remove
**/.git [X]
now we can see [ ] .git
index.html

2:08 am
5/22/2025
(7)

git branch

* master
pointed

Branches like an alternative timeline



index.html

? + tab → boiler plate code
<title> git learning </title>
<body>
looks good at a project.
</body>

git status.
modified : index.html.

now we need to add the file after
git add index.html --
git commit -m "update code for index file"
git branch
* master ← still on the master branch.

• git / HEAD → ref : refs / heads / master

2:45 am
5/22/2025 (8)

refs/heads/
master ← says a master.

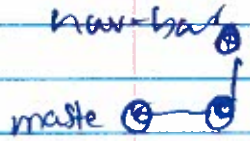
Creating branches: branch name.
git branch nav-bar

git branch
* master
nav-bar
Pointer on the master

refs/head/
master
nav-bar
} two branches.

git checkout nav-bar : changes the branch.
git branch
master
Pointer on → * nav-bar
nav-bar

HEAD
ref : refs/heads/nav-bar



lets now work on the nav-bar.
touch nav-bar.html

nav-bar.html.
<nav>
 home
aboutus
contact us </nav>

2:44 am
5/22/2025
(9)

git status
nav-bar.html untracked
git add nav-bar.html.
git commit -m "add nav-bar to code base".

now the navbar branch had moved forward than the main.

moving back to the master branch.

git checkout master

now the nav-bar.html file will be gone as it was not merged to the master branch

touch hero-section.html

```
<section>
    <h1> Hello how are you.</h1>
    <p> what is your name </p>
</section>
```

git add hero-section.html.
git commit -m "add hero section to the code base"

when we checkout the head will be at our latest point by default.

git branch.
* master

Position of the head can be seen by
git log --oneline

(10)

3:18 am
22/05/2025

git checkout <branch name> } almost the
git switch <branch name> } same.

create a branch and move to it

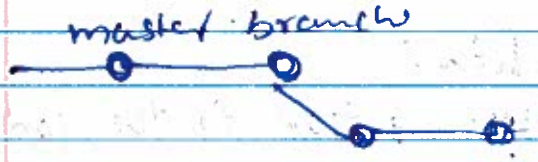
git switch -c branch_name
git checkout -b branch_name

commit before moving to a new branch.

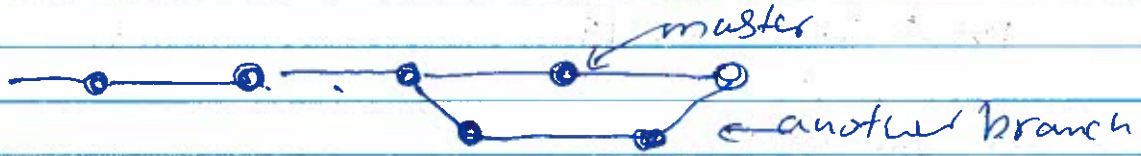
The current developments are in two different branches we need to bring every thing together

⇒ Merging the branches (2)

fast forward merging. (main branch does not change) we work on a separate branch and merge back to the main branch (easy one):



Not fast forward merge.



here both have been modified

11

3:30

5/22/2025

merging:

I am bringing something into me, I should be on the main branch

lets bring nav-bar into the master branch

git checkout master
git branch

until now nav-bar was implemented in a different file which is good.

git merge nav-bar waits for the custom message enter and close the window, already has the default too!

git log --oneline.

git tries its best to resolve conflicts

deleting the branch:

git branch -d nav-bar

git branch
* master

git checkout -b 'footer'

(12)
3:58 am
22/05/2025.

adding a footer

touch footer.html

```
<footer>
    This is my footer
</footer>
```

git add footer.html
git commit -m "add footer section to the code base"

git checkout master

git merge footer

conflicting merges

to the body of the index.html lets add

```
<html lang="en">
<head>
    4012421 content="width=device-width,
initial-scale=1.0">
<title>git learning</title>
</head>
<body>
    looks good as a project
    footer added.
</body>
</html>
```

git status
    modified : index.html.

(37)
22/05/2025
10:27pm

$ git add index.html
$ git commit -m "add footer in index file"

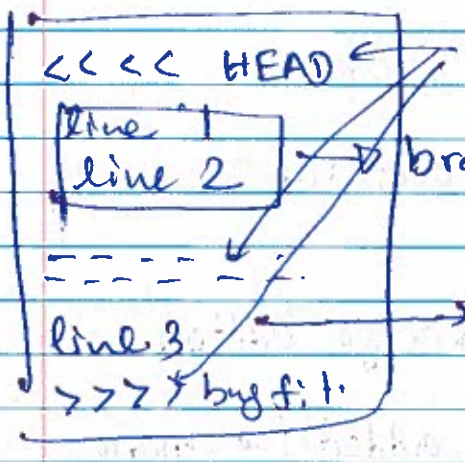
$ git checkout footer.

add to the body

<body>
:
footer code updated
</body>

$ git add index.html.
$ git commit -m "footer code updated to index file"

now both the main branch and the footer branch are changed which is a conflict



These has to be removed.

keep what ever you want, remove markers & save

git checkout master
git merge footer
conflict : merge conflict.
automatic merge failed.

(4)
10:53 pm
5/22/2025

```
<html lang = "en">
<head>
    <title> git learning </title>
</head>
<body>
    looks good as a project.
<<<< HEAD (current change)
    footer added.
=======
    footer was added sucessfully.
>>>>>>> footer
</body>
</html>
```

vscod has button, suggestion to click to accep incoming or accep both or current Change

<<< and things could be removed

```
<body>
    looks good as a project.

    footer was added sucessfully.
</body>
```

In VS code there is merge editor.

Nowe we have manually added the merge

$ git add index.html or git add .

(13) 1:09 pm 22/5/2025

Git Diff

Shows the difference between the same file at two different times

- a → file 1 & b → file 2 (same file over time)
- --- file 1 } indicates changes in file
- +++ file 2 }
- changes in lines & little preview of it.

--- represents file 1
+++ represents file 2

```
$ git status
on branch master

I am modifying the index.html.
<body>
    I would love to add new box here.
    look good as a project
    footer was added sucessfully
</body>
$ git status
    modified : index.html.

$ git add index.html.
$ git status.

$git diff --staged
```

(16)

11:26 pm.
22/5/2025

$ git diff --staged.

learning to read — file a & b both are the same file name

diff --git a/index.html b/index.html
before staging
--- a/index.html. → file a denoted by ---
+++ b/index.html → file b denoted by +++
after staging

<body>
- looks good as a project
-
-
+ I would love to add nav bar here.
+ looks good project.
</body>

for experiment change footer.html
<footer>
add some awesome footer here
</footer>

add footer.html to staging.
$ git add footer.html.
$ git diff --staged.

now both file differences are to be seen

$ git commit -m "change index and footer"
$ git log --oneline.

(11)
11:54 pm
22/05/2025.

Compare from hash.

$ ~~diff~~ git diff 91b11f1 e8b5ba0.

or $ git diff 91b11f1..e8b5ba0

Compare branches.

$ git diff branch1 branch2.
..

$ git diff e8b5ba0.. 91b11f1

+++ will also be interchanged.

Git stash.

Sub tool.

→ create a repo, work & commit on main
→ switch to another branch and work.
→ Conflicting changes do not allow to switch branch, without commits

$ git status
$ git branch.
$ git switch -c bugfix.
$ git branch
* bugfix.

footer.html.

<footer>
add some awesome footer here.
~~from~~ to fix this bug
</footer>

(18) 12:52pm.
5/24/2025

$ git status
modified : footer.html
trying to swith to footer branch
$ git switch footer.
please commit your changes or stash them before switching branches → Aborting

* conflicting changes donot allow to switch branch, without commits

Commons { git stash (you can switch branch)
git stash pop (bring back those changes)

Optional

git stash apply (apply changes & keep them in stash)

$ git stash
saved working directory and index state...

$ git switch footer
(now allowed)

bug fix is temporily saved ...

may do
* editing here.
add trying to fix that bug

after stashing we can work on the footer.

git switch bugfix.

We need to bring back the changes:
$ git stash pop.

(14) 1:00 pm
5/24/2023

Stash allows to bring temporary changes from other branches but should be careful when doing so we are able to select the changes

$ git stash ..
$ git stash list.
stash@{0}:

} for temporary work only

To bring those changes.

git stash apply stash@{0} - change the number as wanted.

More commands

$ git checkout <Hash> (Deattach head) : new branch
$ git switch main (re attach head)
$ git Checkout HEAD~2 (look at 2 commit prior)
$ git restore filename (get back to last commit version)

git checkout.

$ git log --oneline
:
-9106940
:
$ git checkout 9106940.

(20)

Moving back.

$ git checkout master

or

$ git reflog (rarely used)

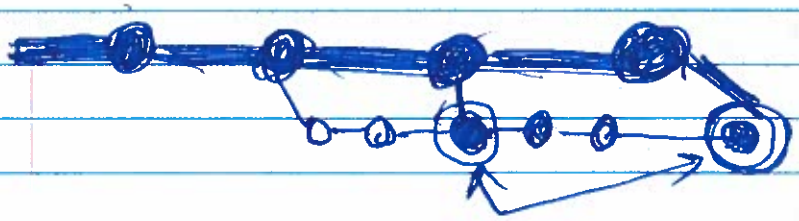
going back in commits

$ git check out HEAD~2 ← move back 2 commits
$ git check out master ← return to the latest commit.

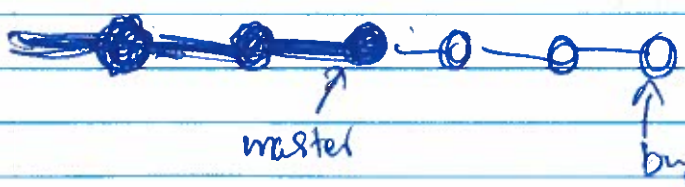
Git Rebase

$ git status

rebase → alternative way of merging..
→ clean up tool (clean up commits)



merge commits.



← Rebase.

When on master / main branch never run this command

- This is meant to be run from secondary branches

$ git branch. adds & commit. * add some to the body.

$ git commit -am "updated main website"

$ git branch
  * master

$ git checkout "bugfix"

→ modify navbar.html.

+ fixed bug .

git commit -am "updated main navbar"

git checkout master

Let's say fixed there are some changes in the master branch & we need to bring those updates to our bugfix branch.

$ git branch
  * bugfix.

$ git merge master
  * need to save the changes on the Vim or the VS code

now lets have more work done on the bug fix branch.

$ git status.

on any branch we can see the log

$ git switch bugfix
$ git log --oneline

and most of the commits are for merging

$ git rebase master

this adds the bugfix branch infront of the master branch

Any conflicts need to be fixed manually

$ git checkout bugfix,
- modifies index.html.
- new pricing card added

git rebase master
for conflicts resolve the manually or Prog
, vscode

follow all instruction on the previous rebase attempt.

$ git add . / index.html.
$ git rebase --continue.

Github.

http://docs.github.com/en/get-started

search for ssh

we need an ssh key to communicate with github via the terminal

* Generating a new SSH key and adding it to the ssh-agent
* Adding a new ssh key to your github account

move to a director.

$ cd ...
$ cd gitthree.
$ ls - - -
$ git status → Fatal : not a git rep.
$ git init.
$ code .
add boiler plate html code.
add → ! → press enter

add to body.

best to change the master ← ⊕ the main
git branch -M main

set up the ssh keys.

To check whether there is a remote repo set up on the system.

$ git remote -V (remote) if there is one

(24)

9:22 pm
5/25/2025

vik

on git-bash

ssh-keygen -t ed25519 -C "vitharana1996@gmail.com"

enter a file in which to save.
Press Enter for default save path

enter passphrase if needed.
vik.
verify passphare vik

- In Powershell (run as admin:)

→ $ Get-Service -Name ssh-agent | Set-Service -Startup Type Manual

$ Start-Service ssh-agent

⇓

Then normal terminal (replace with user name)

ssh-add c:/Users/Sandun/.ssh/id_ed25519

GitHub Settings.
↓
SSH and GPG keys
↓
New SSH key → Key type (✓Authentication key)
copy the ssh key from
C:/Users/Sandun/.ssh/id_ed25519.pub
in VScod and copy the key.

Title : My Lap
✓ authentication key
✓ Paste the ssh key copied -

$ git init
$ git status

In git hub create a repo

Create a new repository., add ~~create~~ descriptions

/nitharana/ learn-git

• Public

↓ after creating the repo we will see commands

echo "# learn-git" >> README.md
Creates a readme file

$ git init (initialize local folder)
$ git add . / index.html ....
$ git commit -m "added the file"
$ git branch -M main (Rename existing branch as main)

Before executing this make sure to be in the correct branch ⇒ $ git checkout master
So that master renamed main

$ git branch
* main

Code to push & streamline the flow.
$ git remote add origin https://github.com/nitharana/learn-gitsit
get from the repo

(26) commands:

$ git clone <url> . get a repo to your system.

$ git config --global user.name "Saudun Sampath Vitharana"

$ git config --global user.email "vitharana1996@gmail.com"

Setup the SSH key:

$ git remote -v ⇒ to check whether you have a remote repo setup [empty if None].

$ git remote add <name> <url>

ex:

$ git remote add origin https://github.com/vitharana/learn-git.git

(default name)

$ git push -u origin main (this can be changed or required by you...)

Push into the Origin from my main

if you need to rename.

git remote rename (origin) <old name> <new name>

or remove

git remote remove origin <name>

git push -u origin main.

git remote -v → origin https ... (fetch) (push)

Verify the remote repo link

(27) To send the code.

git push origin main
(git push origin bugfix.)

git remote -v
origin https:// (fetch)
origin https:// (push)
} mostly this is done once

} git push
fatal: The current branch main has no upstream branches

$ git push --set-upstream origin main
$ git push -u origin main
↓
When main is active now we can directly push.

* When we are done with the local repo we push the code on git hub (remote repo)

$ Pull / fetch ⇒ bring code from the connected repo

$ git pull = git fetch + git merge

$ git fetch : get the info don't put in my work

$ git pull origin main & changes will be merged to main

28