Git is a program that saves all changes made to the file.

Installation of git

Git is a version control system.  
Git helps you keep track of code changes.  
Git is used to collaborating on code.

what it does:-

Manage projects with Repositories  
Clone a project to work on a local copy  
Control and track changes with Staging and Committing  
Branch and Merge to allow for work on different parts and versions of a project  
Pull the latest version of the project to a local copy  
Push local updates to the main project

Over 70% of developers use Git!  
Developers can work together from anywhere in the world.  
Developers can see the entire history of the project.  
Developers can revert to earlier versions of a project.

What is GitHub?  
Git is not the same as GitHub.  
GitHub makes tools that use Git.  
GitHub is the largest host of source code in the world and has been owned by Microsoft since 2018.

git --version

git config --global user.name/email " ....."

create a directory and initialize it with git by

git init - creates the repo

git status - this tells the status of all tracked and untracked files in the repo

Now Git is aware of the file but has not added it to our repository!  
Files in your Git repository folder can be in one of 2 states:  
Tracked - files that Git knows about and are added to the repository  
Untracked - files that are in your working directory but not added to the repository  
 When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them or add them to the staging environment.

Commits can only happen after files are in the staging area.

Git Staging Environment -One of the core functions of Git is the concepts of the Staging Environment and the Commit.

git add filename.ext --- this command helps to get the file into stagging environment , or git starts tracking them. Or to add all the files do git add –all or git add .

COMMIT

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point". It is a point in the project you can go back to if you find a bug, or want to make a change.

When we commit, we should **always** include a **message**.

Git commit –m “ …...” ----used for commiting ( m for message) or example

git commit -m "First release of Hello World!"

Git log ------ to check all the data of commits

All commits have unique hash code

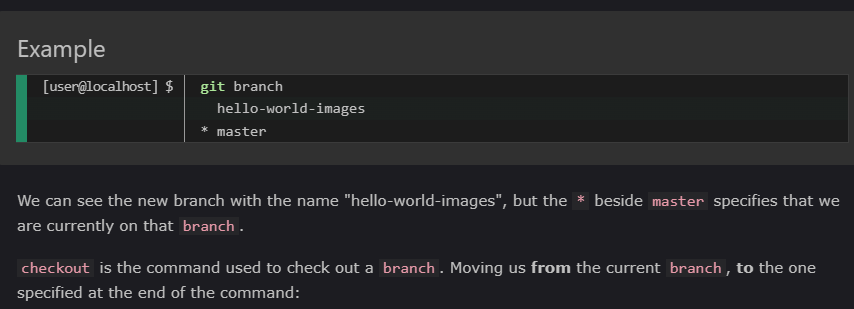
Clear to clear git bash

Git checkout #HASHCODE of the COMMIT u want to go ( going to past )

git checkout master --------- to come to present ( of repo)

* If you are having trouble remembering commands or options for commands, you can use Git help.
* git *command* -help - See all the available options for the specific command
* git help --all - See all possible commands

In Git, a branch is a new/separate version of the main repository.



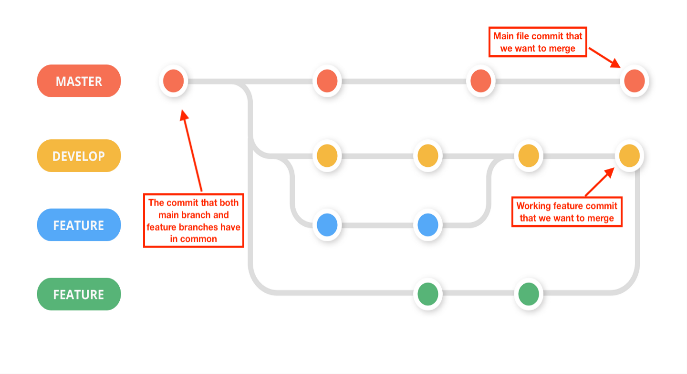
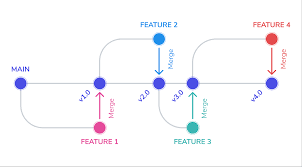
git branch ---- command shows all the branches and master branch in the repo

Git branch branchname -----> used to create a branch

git checkout branchname ----> to get into that branch

chechout command helps to navigate between branches

\* tells us on which branch are we presently.



So basically many branches can be made in master branch and many branches can be made with those branches also so that various features and manipulation can be done . Its like creating various time lines and changing all timelines .

Now ,

When features are done , we can merge the existing branches back into main branch so that all files can co exist . A branch is always merged back to the parent branch . This parent branch and child branch analogy is cool . Various child branches can be merged back to a parent branch and parent branch can br merged to their parent branch and so on , until at last everything merges back to master branch in a repo .

git merge child-branch-name or git merge branchname simple to put. -----> With this the child branch will get merge with parent branch.

Git log for commit checking and git status to check status of files existing and tracking.

To hide files from getting into git environment or to ignore certain files from bein track or to hide them while working simple create :-

touch .gitignore ----> this will create a file called .gitignore . Then enter the name of file u want to ignore . Ex – file.txt . This will hide the file and will be ignored by the git.

git chechout –b branchname -----> using this code a new branch will be created , and u will directly checkout to that branch on the spot.

Remember to create new child branches when you checkout at the parent branches.

To create new branch from master directly and go into that do :

Git checkout master

Git checkout –b new

Once branches are merged and u don’t need those branches anymore since master has it all , so u can also delete the branches using -

git branch –d branch-name

MERGE CONFLICTS

Merge conflicts can be an intimidating experience. Luckily, Git offers powerful tools to help navigate and resolve conflicts. Git can handle most merges on its own with automatic merging features. A conflict arises when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other. Conflicts will most likely happen when working in a team environment.

There are many tools to help resolve merge conflicts. Git has plenty of command line tools we discussed here. For more detailed information on these tools visit stand-alone pages for git log, git reset, git status, git checkout, and git reset.

GITHUB

Creating repo – choose public or private .

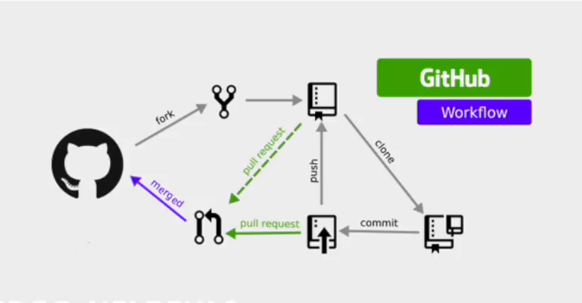
## Push Local Repository to GitHub

Create repo , then in quick setup . Copy the link or localhost url of github account and pase it in git bash , also git branch –M master and git push –u origin master (pushes changes to origin)

Using git remote –v u can check the origin from where push and fetch will happen .

First time an authentication is required . U can then see ur local repo on GitHub.

Now it is connected to GitHub. U can now make changes to ur code from ide and using command git push ----> u can send changes on the GitHub , then refresh page .



Now you can edit files in github and commit then accordingly.

Any time you start working on a project, you should get the most recent changes to your local copy.

With Git, you can do that with pull.

pull is a combination of 2 different commands:

* fetch
* merge

## Git Fetch

fetch gets all the change history of a tracked branch/repo.

So, on your local Git, fetch updates to see what has changed on GitHub:

git fetch origin

If the local copy and origin/master copy are different then

we can also verify by showing the differences between our local master and origin/master:

git diff origin/master

That looks precisely as expected! Now we can safely merge. To get the perfect updated copy we merge.

merge combines the current branch, with a specified branch.

git merge origin/master

There! Your local git is up to date!

## Git Pull

But what if you just want to update your local repository, without going through all those steps?

pull is a combination of fetch and merge. It is used to pull all changes from a remote repository into the branch you are working on.

Example

Make another change to the Readme.md file on GitHub.

Use pull to update our local Git:

git pull origin

Easy done. That is how you keep your local Git up to date from a remote repository.

[Push to GitHub](https://www.w3schools.com/git/git_push_to_remote.asp?remote=github)

Now if you did changes to local repo here and it is not updated on Github. You can push and update the files on Github there by.

git push origin

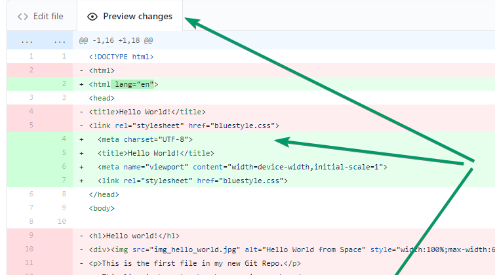
And confirm the changes on Github .

Branch on github

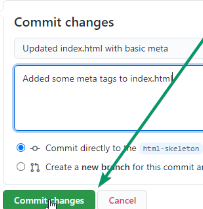
On GitHub, access your repository and click the "master" branch button.

There you can create a new Branch. Type in a descriptive name, and click Create branch:

The branch should now be created and active. You can confirm which branch you are working on by looking at the branch button. Start working on an existing file in this branch. Click the "index.html" file and start editing: After you have finished editing the file, you can click the "Preview changes" tab to see the changes you made highlighted:

  
If you are happy with the change, add a comment that explains what you did, and click Commit changes.

You now have a new branch on GitHub, updated with some changes!



Now continue working on our new branch in our local Git.

Lets pull from our GitHub repository again so that our code is up-to-date:

Do a quick status check: g

where we are working at the moment: git branch

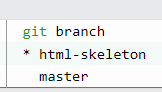
So, we do not have the new branch on our local Git. But we know it is available on GitHub. So we can use the -a option to see all local and remote branches:

git branch –a ---> This helps to display all local and remote branches. On the other hand branch –r is for remote branches only.

We see that the branch html-skeleton is available remotely, but not on our local git. So

git checkout html-skeleton

git pull

 yup its done. Check using git branch

That is how you pull a GitHub branch to your local Git.

Now

Let's try to create a new local branch, and push that to GitHub.

Start by creating a branch, like we did earlier:

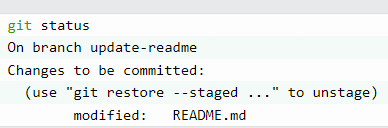
git checkout -b update-readme ( And we make some changes to the README.md file.)

git status

We see that README.md is modified but not added to the Staging Environment:

So , git add README.md

Check the status of the branch:

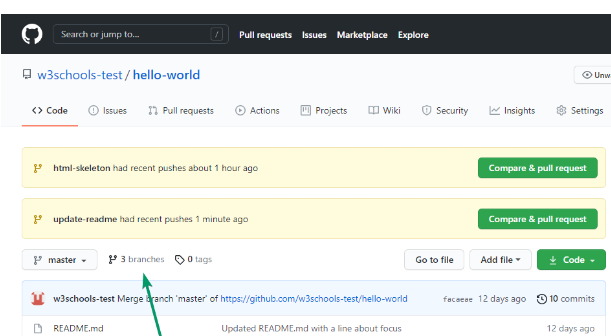


git commit -m "Updated readme for GitHub Branches"

Now push the branch from our local Git repository, to GitHub, where everyone can see the changes:

git push origin update-readme

Go to GitHub, and confirm that the repository has a new branch:



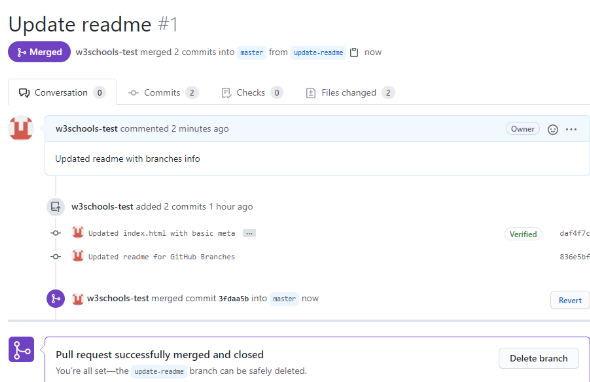
In GitHub, we can now see the changes and merge them into the master branch if we approve it.

If you click the "Compare & pull request", you can go through the changes made and new files added

If the changes look good, you can go forward, creating a pull request:

A pull request is how you propose changes. You can ask some to review your changes or pull your contribution and merge it into their branch.

Since this is your own repository, you can merge your pull request yourself:



To keep the repo from getting overly complicated, you can delete the now unused branch by clicking "Delete branch".

--------------------------------------------------------------------------------------------

U can edit any file in vim editor inbuild in such terminals. vim file.ext - click I to get into editor mode and press esc + ; + w + q to get out of vim in bash.

--------------------------------------------------------------------------------------------

A fork is a copy of a repository. This is useful when you want to contribute to someone else's project or start your own project based on theirs.

U can also contribute and collaborate with others

1. By forking for open source projects
2. By adding contributers to ur own repo. Settings -> manage access -> contribute -> invite them

Forking makes a copy of project on ur github. U basically gets a copy of that repo where u can make changes.

Now we have our own fork, but only on GitHub. We also want a clone on our local Git to keep working on it.

A clone is a full copy of a repository, including all logging and versions of files. Move back to the **original** repository, and click the green "Code" button to get the URL to clone:

Open your Git bash and clone the repository:

git clone <https://github.com/w3schools-test/w3schools-test.github.io.git>

Now we have a full copy of the original repository.

## Configuring Remotes – only required if we don’t clone our forked repository. If we clone the original file we only get read access therefore we then need to configure the clone with fork. So fork and clone.