1. // git –version
2. Git and GitHub are different things.

Configure Git

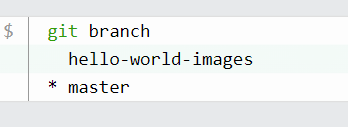
1. git config --global user.name "w3schools-test"
2. git config --global user.email [test@w3schools.com](mailto:test@w3schools.com)

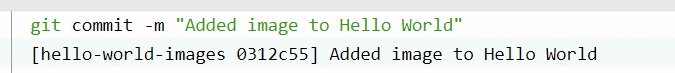
## Creating Git Folder

1. mkdir myproject
2. cd myproject
3. mkdir **make**s a **new directory**.
4. cd **changes** the **current working directory**.
5. initialize Git on that folder:
6. git init
7. ls
8. ls will **list** the files in the directory
9. Git status
10. 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

1. **taged** files are files that are ready to be **committed** to the repository you are working on.
2. git add index.html
3. git add –all or git add .
4. Git Commit
5. 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.
6. we commit, we should **always** include a **message**.
7. git commit -m "First release of Hello World!"
8. To view the history of commits for a repository, you can use the log command: git log
9. git help --all -  See all possible commands
10. In Git, a branch is a new/separate version of the main repository.
11. Branches allow you to work on different parts of a project without impacting the main branch.
12. When the work is complete, a branch can be merged with the main project.
13. create a new branch:
14. git branch hello-world-images
15. Now we created a new branch called "hello-world-images"
16. 
17. Graphical user interface, application, Word

    Description automatically generated
18. 
19. Graphical user interface, text, website

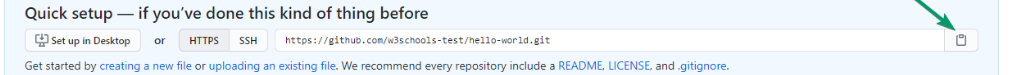
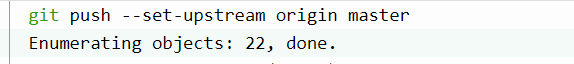
    Description automatically generated
20. Graphical user interface, text, application

    Description automatically generatedThis creates as well as swich to the new branch
21. First switch to main branch and then we can merge the current branch (master) with emergency-fix: Text

    Description automatically generated with low confidence
22. Graphical user interface, application, Word

    Description automatically generated

## Create a Repository on GitHub

1. create a new Repo:
2. Since we have already set up a local Git repo, we are going to push that to GitHub:
3. 
4. Copy the URL
5. Now paste it the following command:
6. 
7. Now we are going to push our master branch to the origin url
8. 
9. We can edit code in GitHub
10. 
11. Git pull to keep up to date with changes

pull is a combination of 2 different commands:

1. fetch
2. merge
3. fetch gets all the change history of a tracked branch/repo.
4. So, on your local Git, fetch updates to see what has changed on GitHub:
5. Graphical user interface, application

   Description automatically generated
6. Graphical user interface, application

   Description automatically generated
7. we can also verify by showing the differences between our local master and origin/main:
8. A screenshot of a computer

   Description automatically generated with medium confidence

merge combines the current branch, with a specified branch.

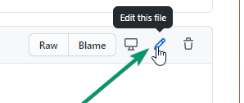
We have confirmed that the updates are as expected, and we can merge our current branch (master) with origin/master:

1. A picture containing text

   Description automatically generated
2. 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.
3. Text

   Description automatically generated with low confidence
4. Git push to gitHub
5. A picture containing text

   Description automatically generated
6. Graphical user interface, text, application

   Description automatically generated
7. Go to GitHub, and confirm that the repository has a new commit:
8. Create new branch on GitHub
9. 
10. Graphical user interface, text, application

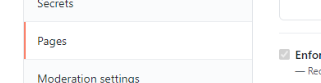
    Description automatically generated
11. Graphical user interface, text, application, chat or text message

    Description automatically generated
12. Pull branch from github
13. Graphical user interface, text, application

    Description automatically generated
14. Now our main branch is up todate
15. Do a quick status check:
16. And confirm which branches we have, and where we are working at the moment:
17. 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:
18. 
19. Shape, logo

    Description automatically generated
20. Push a branch to GitHub
21. git push origin (Branch name)
22. 
23. In GitHub, we can now see the changes and merge them into the master branch if we approve it.
24. If the changes look good, you can go forward, creating a pull request:
25. Github flow

The GitHub flow works like this:

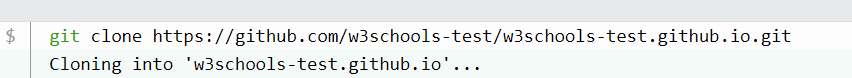
1. Create a new Branch
2. Make changes and add Commits
3. Open a Pull Request
4. Review
5. Deploy
6. Merge
7. Github pages
8. , , ,Graphical user interface, text, application

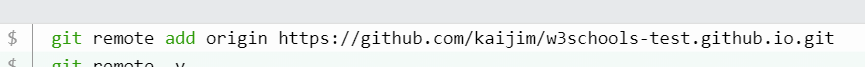
   Description automatically generated

## Git Contribute

1. [GitHub Fork](https://www.w3schools.com/git/git_remote_fork.asp?remote=github),[Git Clone from GitHub](https://www.w3schools.com/git/git_clone.asp?remote=github),[GitHub Send Pull Request](https://www.w3schools.com/git/git_remote_send_pull_request.asp?remote=github)
2. [GitHub Fork](https://www.w3schools.com/git/git_remote_fork.asp?remote=github)
3. 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.
4. fork is not a command in Git, but something offered in GitHub
5. Graphical user interface, application

   Description automatically generated
6. Now we have our own fork, but only on GitHub. We also want a clone on our local Git to keep working on it.
7. Move back to the **original** repository, and click the green "Code" button to get the URL to clone:
8. Graphical user interface, text, application, chat or text message

   Description automatically generated
9. Open your Git bash and clone the repository
10. 
11. And check the log to confirm that we have the full repository data:
12. A picture containing graphical user interface

    Description automatically generated
13. 
14. Text

    Description automatically generated
15. Now we push them to our GitHub fork
16. Graphical user interface, application, website

    Description automatically generated
17. we can send a Pull Request to the original repository:
18. Graphical user interface, text, application, chat or text message

    Description automatically generated
19. Graphical user interface, text, application, chat or text message

    Description automatically generatedRemember to add an explanation for the administrators.
21. Git ignore
22. When sharing your code with others, there are often files or parts of your project, you do not want to share.
24. Git will not track files and folders specified in .gitignore. However, the .gitignore file itself **IS** tracked by Git.
25. create a .gitignore file
26. Text

    Description automatically generated with medium confidence
27. Now open the file using a text editor.

* Ignore any files with the .log extension
* Ignore everything in any directory named temp

1. Graphical user interface, text, application

   Description automatically generated
2. It is also possible to have additional .gitignore files in subdirectories. These only apply to files or folders within that directory.