

DA-IICT

GIT AND GITHUB

ALL YOU NEED TO KNOW ABOUT



Some basic info about git and github

Repository -> Folder place where all your folders and codes are kept

Github: A website that hosts your online repository

Git: Version control system

Git Commands

- 1) Clone -> Bring a repository that is hosted somewhere else like GitHub into your folder on your local machine
- 2) add -> Track your files and changes in it
- 3) Commit -> Save your files in git
- 4) Push -> upload Git commits to remote repo like Github



Creating a new repo on github

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



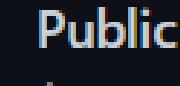
Bhavya418

Repository name *

/

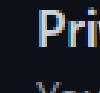
Great repository names are short and memorable. Need inspiration? How about [turbo-doodle](#)?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

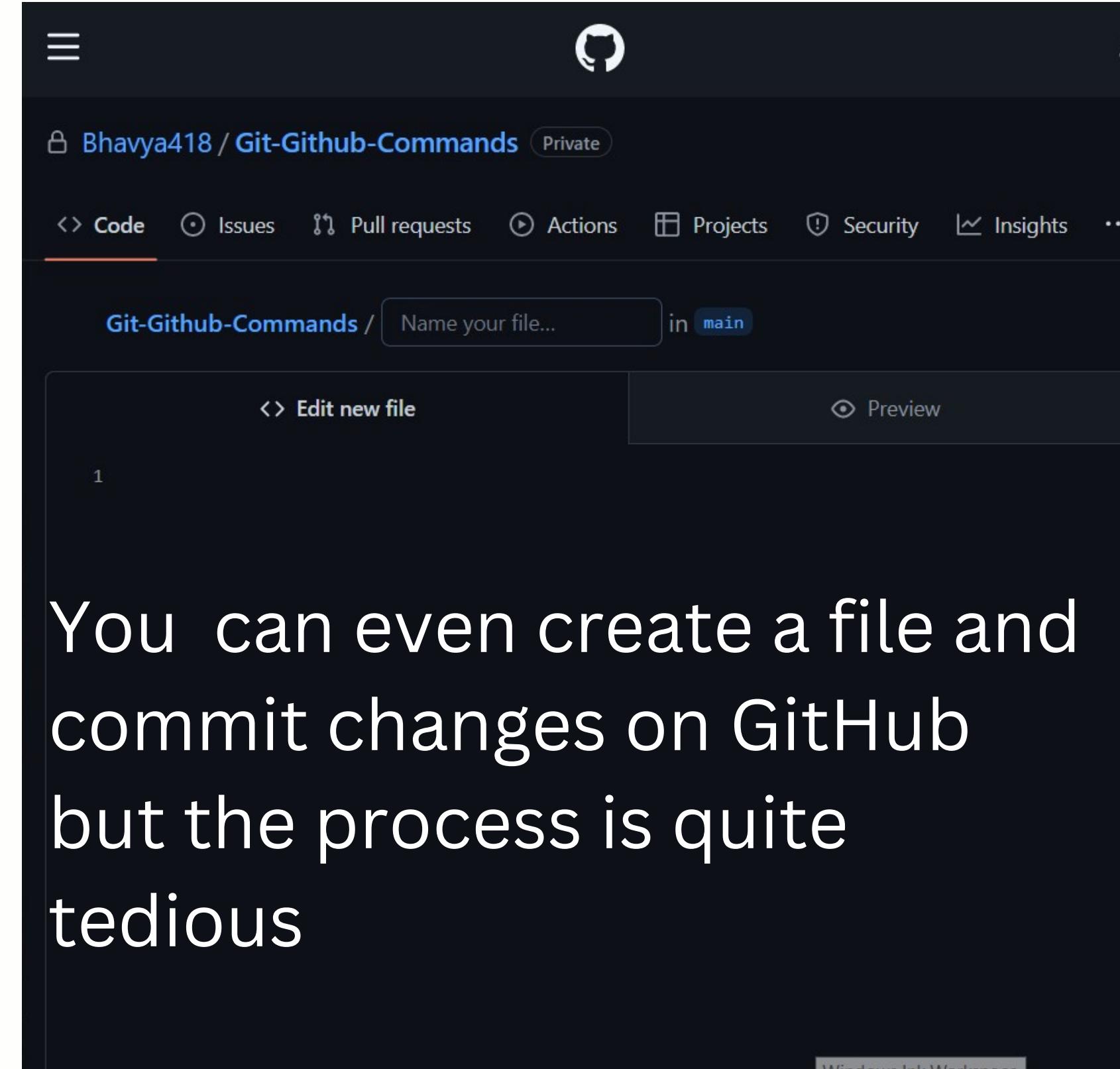
Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Creating the file in github



You can even create a file and commit changes on GitHub but the process is quite tedious

Every commit in github has its own id through which it can be tracked

Install git from the link provided below

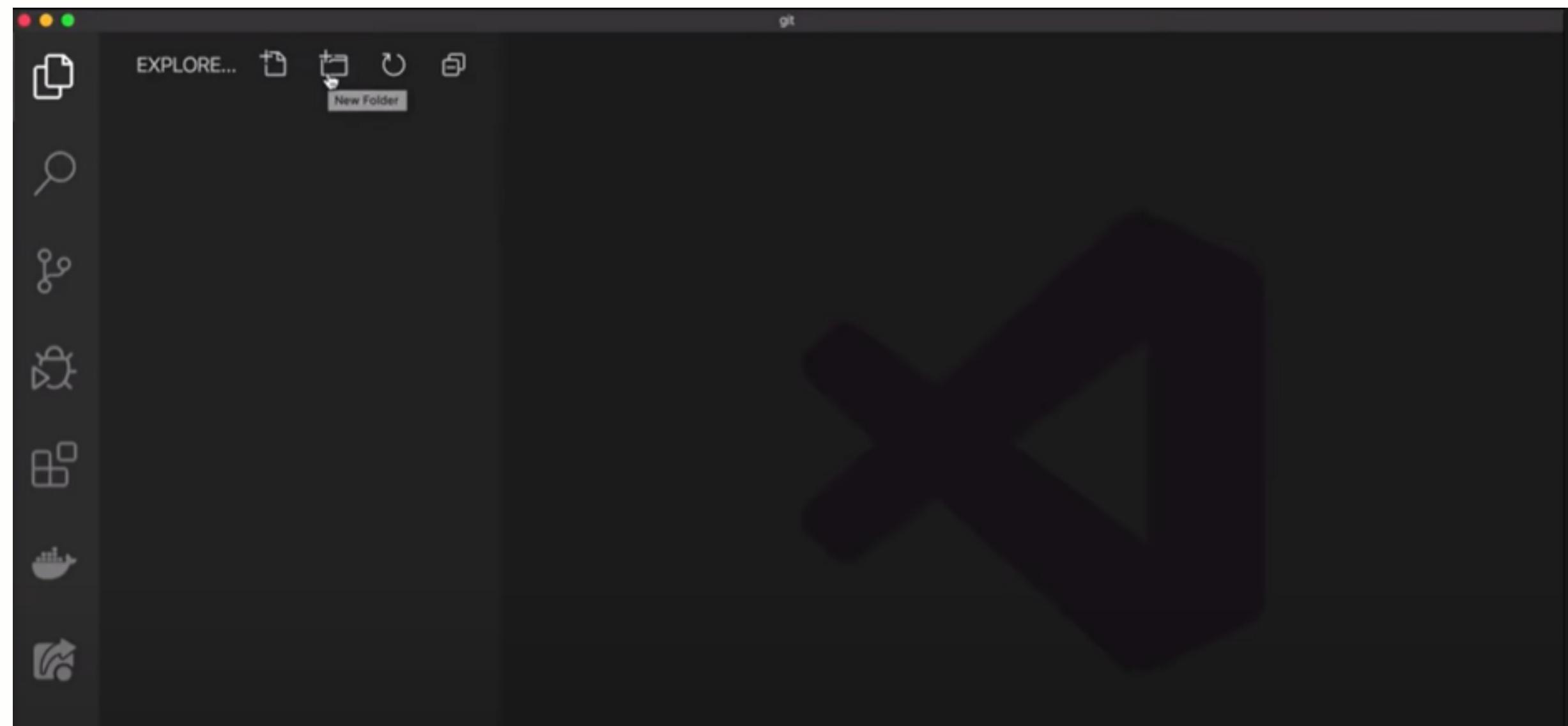
Install Git

[Install Git on Mac OS X](#) / [Install Git on Windows](#) /
[Install Git on Linux](#)

<https://www.atlassian.com/git/tutorials/install-git>

Install a text editor

Visual Studio code



1) git config --global user.name "xyz"

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

-> To create a git account and use this on github to login.

Use global to set the username for every repository you create

git --version to check the git version installed

2) mkdir -> create a new directory

3) cd -> to change the current working directory

4) git init -> to create a empty git repository in the specified folder

How to check if it is a git repository?

So it has created a hidden file .git inside your folder which can be seen by command ls -la

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ ls -la
total 8
drwxr-xr-x 1 shahb 197609 0 oct 1 14:01 .
drwxr-xr-x 1 shahb 197609 0 oct 1 14:09 ..
drwxr-xr-x 1 shahb 197609 0 oct 1 14:10 .git/
-rw-r--r-- 1 shahb 197609 625 oct 1 14:07 README.md
```

5) git status -> to check the status of the file if it is part of our repo or not

git status --short -> To see status in a shorter format

i) Tracked files ii) Untracked files iii) Modified files

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git status
On branch main
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md
nothing added to commit but untracked files present (use "git add" to track)
```

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git status
On branch main
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md
```

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git status
On branch main
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:  README.md
```

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git status --short
AM README.md
```

6)git add . or filename or --all -> To add the untracked file to the repo but yet are to committed to the repo(Just added)

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git commit -am "This a file for the information about git"
[main (root-commit) fff1b457] This a file for the information about git
 1 file changed, 21 insertions(+)
 create mode 100644 README.md

shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git status
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean
```

7) git commit -m "Placeholder" -m "Description"->
To commit the changes made to the repo and reflect it in local machine not at github

git commit -am "Placeholder" -m "Description"->
To commit and add the changes made to the file at once to a modified file

```
git commit -m "First release of Hello World!"
[master (root-commit) 221ec6e] First release of Hello World!
 3 files changed, 26 insertions(+)
 create mode 100644 README.md
 create mode 100644 bluestyle.css
 create mode 100644 index.html
```

8)git clone sshlink -> this will clone the repository into your folder

9) git log -> to view the commits history for the repository

```
shahb@Bhavya-Shah MINGW64 ~/OneDrive/Documents/Git/Git-Github-Commands (main)
$ git log
commit ff1b457e8e849adb172b4d4bc8bad61c3d91947b (HEAD -> main)
Author: Bhavya418 <shahb3703@gmail.com>
Date:   Sat Oct 1 14:48:07 2022 +0530

    This a file for the information about git
```

10) git help --all -> to get help for the command
If you find yourself stuck in the list view, SHIFT + G to jump the end of the list, then q to exit the view.

SSH Keys

SSH Keys are unique keys with which you can connect to github without any user name or personal access token at each visit.

```
ssh-keygen -t rsa -b 4096 -C "emailAddress"
```

```
→ ~ ssh-keygen -t rsa -b 4096 -C "email@example.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/gwen/.ssh/id_rsa): testkey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in testkey.
Your public key has been saved in testkey.pub.
The key fingerprint is:
SHA256:gHmbQn8S/hAhl2GMFzAap1moz2HYzD0wf6LkisCW9J0 email@example.com
The key's randomart image is:
+---[RSA 4096]---+
| ..*+*+
| +B.B+.
| *+=+.=
| o.B.=+.*
| o*ooo+0 S
| .+=. E =
| +. .
| o
+---[SHA256]---+
→ ~ █
```

- 1 Start the ssh-agent in the background.

```
$ eval "$(ssh-agent -s)"
> Agent pid 59566
```

- 2 If you're using macOS Sierra 10.12.2 or later, you will need to modify your `~/.ssh/config` file to automatically load keys into the ssh-agent and store passphrases in your keychain.

```
Host *
AddKeysToAgent yes
UseKeychain yes
IdentityFile ~/.ssh/id_rsa
```

- 3 Add your SSH private key to the ssh-agent and store your passphrase in the keychain. If you created your key with a different name, or if you are adding an existing key that has a different name, replace `id_rsa` in the command with the name of your private key file.

```
$ ssh-add -K ~/.ssh/id_rsa
```

```
ls | grep testkey
cat testkey.pub
```

Copy this to github ssh and gph section

12) git push -u origin master -> To finally push the files to the github repo

If the repo was made on local machine using git init and not on github then you will be required to create a repo on github and link it

13)git remote add origin (link of repo from github)

14) git remote -v -> To check if it is linked to any repo or not and then you can use git push again.

Github Workflow

Write code



Commit Changes



Make a Pull Request



Local Git Workflow

Write code



Stage Changes



Commit Changes



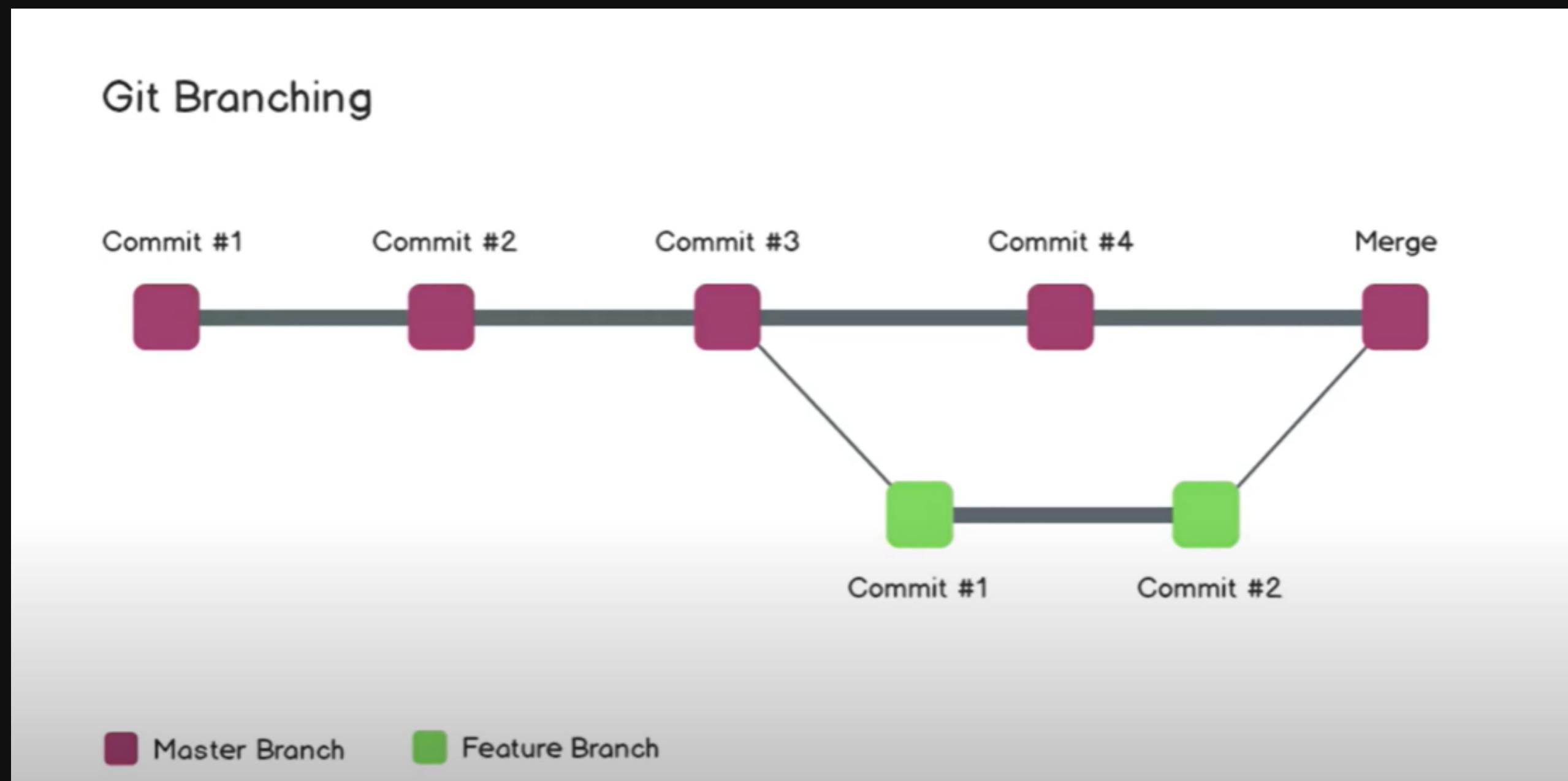
Push Changes



Make a Pull Request



Git Branch



It will be helpful when working in a team and different person are working on different features, so they can work independently without any hindrance to master branch and once reviewed can be merged with master branch using pr .

- 15) git branch -> to get list of number of branches
- 16) git checkout -b newbranch -> to create a new branch and even switch between branches

Push the new branch to the github and then create pull request for the main branch to merge it if it is correct.

- 17) git branch -d branchname -> to delete the branch
- Merge conflict when both branches are changes and have content at same line

So we merge master into our branch to stay updated with it

To copy someone else's repo into your repo -> Github's fork command is used

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

fork is not a command in Git, but something offered in GitHub and other **repositorio** hosts. Let's start by logging in to GitHub, and **fork** our repository:

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

w3schools-test / w3schools-test.github.io

Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags

Go to file Add file Code

Fork your own copy of w3schools-test/w3schools-test.github.io to your account

About

w3schools-test Merge branch 'master' of https://github.com/w3schools-test/hello-world facaeae 18 days ago 10 commits

README.md Updated README.md with a line about focus 18 days ago

Git Work Flow

Create a New Branch

Branching is the key concept in Git. And it works around the rule that the master branch is ALWAYS deployable.

That means, if you want to try something new or experiment, you create a new branch! Branching gives you an environment where you can make changes without affecting the main branch.

When your new branch is ready, it can be reviewed, discussed, and merged with the main branch when ready.

When you make a new branch, you will (almost always) want to make it from the master branch.

Make Changes and Add Commits

After the new branch is created, it is time to get to work. Make changes by adding, editing and deleting files. Whenever you reach a small milestone, add the changes to your branch by commit.

Adding commits keeps track of your work. Each commit should have a message explaining what has changed and why. Each commit becomes a part of the history of the branch, and a point you can revert back to if you need to.

Open a Pull Request

Pull requests are a key part of GitHub. A Pull Request notifies people you have changes ready for them to consider or review.

You can ask others to review your changes or pull your contribution and merge it into their branch.

Review

When a Pull Request is made, it can be reviewed by whoever has the proper access to the branch. This is where good discussions and review of the changes happen.

Pull Requests are designed to allow people to work together easily and produce better results together!

If you receive feedback and continue to improve your changes, you can push your changes with new commits, making further reviews possible.

Deploy

When the pull request has been reviewed and everything looks good, it is time for the final testing. GitHub allows you to deploy from a branch for final testing in production before merging with the master branch.

If any issues arise, you can undo the changes by deploying the master branch into production again!

Merge

After exhaustive testing, you can merge the code into the master branch!

Pull Requests keep records of changes to your code, and if you commented and named changes well, you can go back and understand why changes and decisions were made.

Github pages

First create a repo with speifies name , clone the repo and add all the files into that repo, push the changes and you are done

<https://pages.github.com/>