Github- Zero To Hero

Note: repo=repository/directory/folder

1) **Add- to add the changes in the file to the staging area**

The `git add` command is used to add changes in a file to the staging area. This allows you to prepare the changes for a commit.

> **staging area- a stopping place or assembly point en route to a destination.**

2) **Commit- to make commitment of the changes to the file**

The `git commit` command is used to commit the changes made to the file. This creates a new commit in the repository, representing a snapshot of the changes.

When committing, it is important to provide a meaningful message using the `-m` flag. This message should describe the changes made in the commit.

Example: `git commit -m "Added new feature"`

Remember to always commit your changes before pushing them to the remote repository.

3) Push- to upload the changed files to the remote repo of git

Push- to upload the changed files to the remote repo of git

The `git push` command is used to upload the changed files to the remote repository of Git. This updates the remote repository with the latest changes made in the local repository.

Example: `git push origin main`

Here, `origin` refers to the name of the remote repository, and the `main` represents the branch where the changes will be pushed.

Remember to always commit your changes before pushing them to the remote repository.

Git commands— they use kinda similar commands to those of Linux

touch- to make a file

code - to edit the file. for example. html

----if remote repo is made from the web and then set to the local repo---

- git clone <link> -- to clone/download the repository to the local host

  git status -- check the status of git

- --make any changes if necessary--

==== If any changes are made then===

git add <filename> -- to add the file to the staging area .. staging area is the buffer zone between the local and remote repos before reflecting/updating the remote data

(alternate)--- git add. -- for adding all the latest modified files and then commit the changes

git commit -m "meaningful message" == here -m means message to be displayed

now push the code to the remote repo

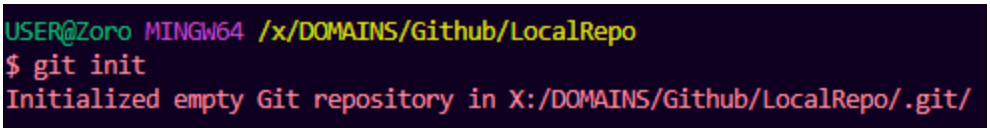
git push origin main--- origin = GitHub repo name

main = branch i.e sacred timeline - in the case of Marvel

----if u have a working directory/local repo in ur device and u want to push that to the remote repo then…. we need to initialize the Git to that directory---

git init — used to create a new git repo

To initialize a new Git repository in a directory, you can use the `git init` command. This command sets up a new Git repository, allowing you to start tracking changes in your project.



to check whether the Local repo is a Git repo or not… ls -a command

if there is a file .git present then it is Git repo

else not

```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo
$ ls -a
./  ../
```

then update the files using previous codes

git add . / git add <filename>

git commit -m “message”

now for the real part of uploading to the repository

git remote add origin <link of the GitHub repo> —- origin is the name of repository

The `git remote add origin` command is used to connect your local repository to a remote repository on GitHub. The "origin" in this command refers to the name of the remote repository. You need to replace `<link of the GitHub repo>` with the actual link of the GitHub repository you want to connect to.

For example:

`git remote add origin <https://github.com/username/repository.git` >

Remember to replace "username" with your GitHub username and "repository" with the name of your repository.

After running this command, your local repository will be linked to the remote repository, allowing you to push your local changes to the remote repository using the `git push` command.

to check the current git remote repo

git remote -v

```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (master)
$ git remote -v
origin  https://github.com/Deepesh-Sunuwar/LocalRepo.git (fetch)
origin  https://github.com/Deepesh-Sunuwar/LocalRepo.git (push)
```

The `git remote` command is used to view the current remote repositories associated with the local Git repository. It displays the names of the remote repositories along with their URLs.

To check the current remote repositories, you can use the following command:

```
git remote -v
```

This command will display the names and URLs of the remote repositories, like this:

```
origin  <https://github.com/username/repository.git> (fetch)
origin  <https://github.com/username/repository.git> (push)
```

The "origin" here refers to the name of the remote repository, and the URL represents the location of the remote repository on GitHub.

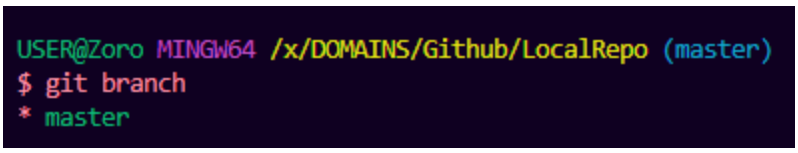
git branch — to check the current branch

The `git branch` command is used to list, create, or delete branches in Git. It allows you to see the existing branches in your repository.

To list all the branches in your repository, you can use the following command:

```
git branch
```

This will display a list of branches, with the currently active branch highlighted.



to rename a branch

git branch -M “new name”

```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (master)
$ git branch -M main

USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (main)
$ git branch
* main
```

now to push the code

git push origin main — one-time push

(alias) git push -u origin main— -u means upstream i.e if we want this repository to be used for a long time then it is tedious to type *git push origin main* for individual push so if we use -u then we just need to enter *git push* and it pushes all the changes

The `-u` flag in the `git push -u origin main` command stands for "upstream."

origin = remote repo and main means branch

When you use `-u`, it sets the upstream branch for the current branch. This means that in future pushes, you can simply use `git push` without specifying the remote repository and branch name. Git will automatically push to the upstream branch you specified.

So, when you run `git push -u origin main`, you are pushing your local changes to the `main` branch of the `origin` remote repository, and setting it as the upstream branch for the current branch.

After setting the upstream branch, you can use `git push` in the future to push your changes to the `main` branch without having to specify the remote repository and branch name again.

note: if u have created a repo with readme,md file

and if u try to push the content of local repo to the remote repo then conflict occurs as the local doesn't have the readme.md file so..

u need to pull the repo first

git pull

git branch —set-upstream-to=origin/main main

git pull —allow-unrelated-histories

now the readme file should be pulled to the local repo

now push the content to the remote repo

Branches in Git

to go to the next branch

git checkout <destination branch>

to make a new branch

git checkout -b <new branch name>

```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (main)
$ git checkout -b feature1
Switched to a new branch 'feature1'

USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (feature1)
$ git branch
* feature1
  main

USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (feature1)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

to delete branch

git branch -d <branch name>

note: to delete a particular branch, we need to first to go the next branch ie. say main branch

```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (main)
$ git branch -d feature2
Deleted branch feature2 (was 3741eed).
```

working with the branches as a separate version of the code

current branch : main

```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (main)
$ git branch
  feature1
* main
```

```html
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>From LocalRepo to GitRepo</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <p>This is pushing repo directly from Local Host to Remote repo</p>
  </body>
</html>
```

feature1 branch

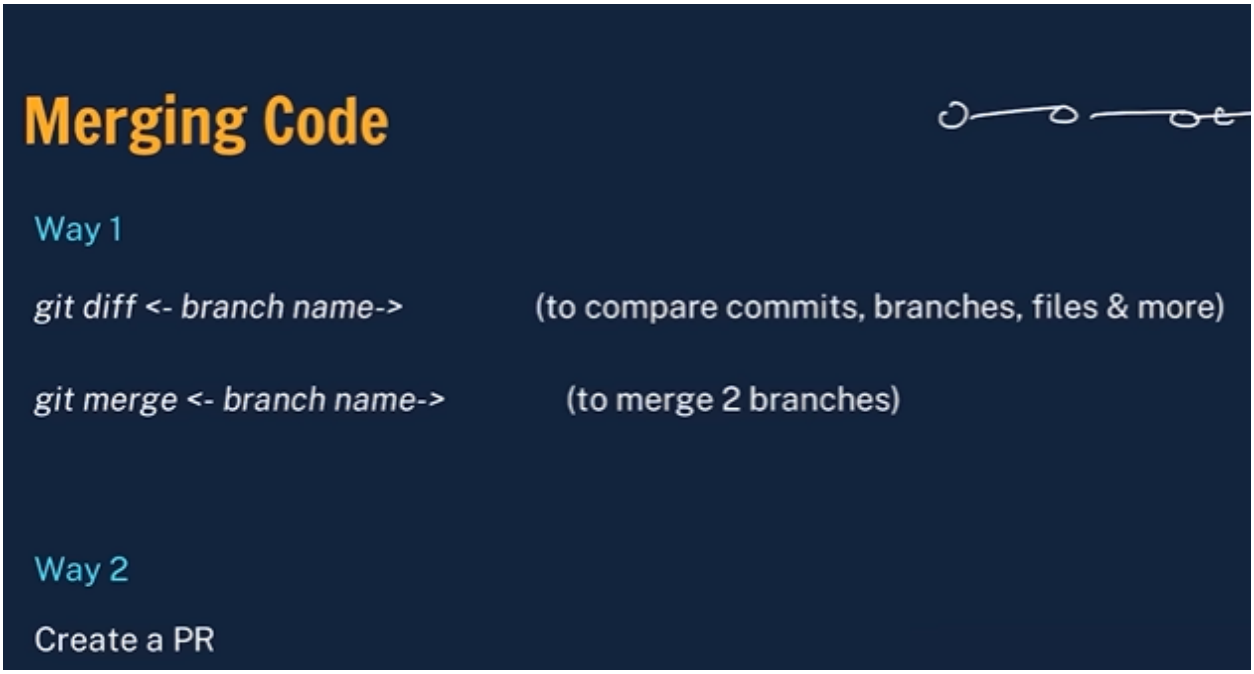
```
USER@Zoro MINGW64 /x/DOMAINS/Github/LocalRepo (feature1)
$ git branch
* feature1
  main
```

```
alRepo > index.html > html > body > p
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>From LocalRepo to GitRepo</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <p>This is pushing repo directly from Local Host to Remote repo</p>
    <br />
    <p>this is a new feature modified from feature1 branch</p>
  </body>
</html>
```

there is separate data for each branch that acts differently.,. so the main branch host the fixed or core content of the code

while the feature1 branch holds the new updates that are to be reflected to the main branch or the actual software in a real scenario.

Merging in Git



Git Pull Request

A Git Pull Request (PR) is a feature in Git that allows developers to propose changes to a project's codebase. It is commonly used in collaborative development workflows, especially when working with remote repositories and multiple branches.

To create a Git Pull Request, follow these steps:

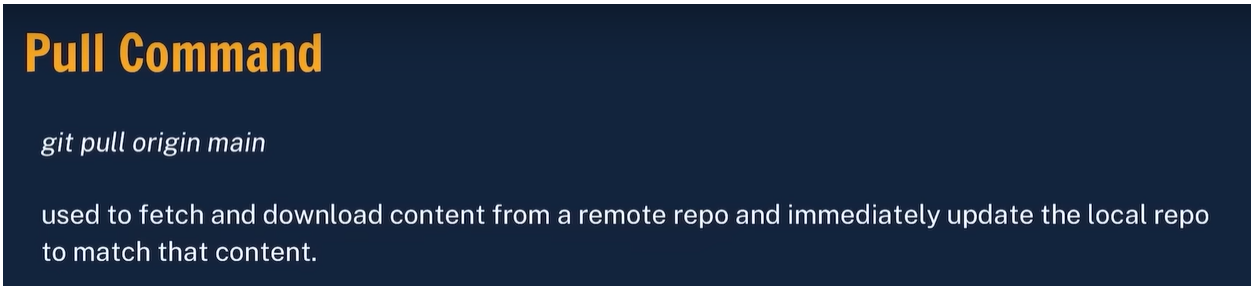
1. *Make sure you are on the branch that contains the changes you want to propose for merging.*
2. *Push your branch to the remote repository using the* `git push` *command. For example,* `git push origin branch-name`*.*
3. *Go to the repository's page on the Git hosting platform (such as GitHub or GitLab).*
4. *Find the option to create a new Pull Request.*
5. *Select the branch with your changes as the "compare" branch and the target branch where you want to merge the changes.*
6. *Provide a descriptive title and description for the Pull Request, explaining the purpose and scope of the changes.*

7. *Review the changes and make any necessary adjustments.*
8. *Submit the Pull Request.*

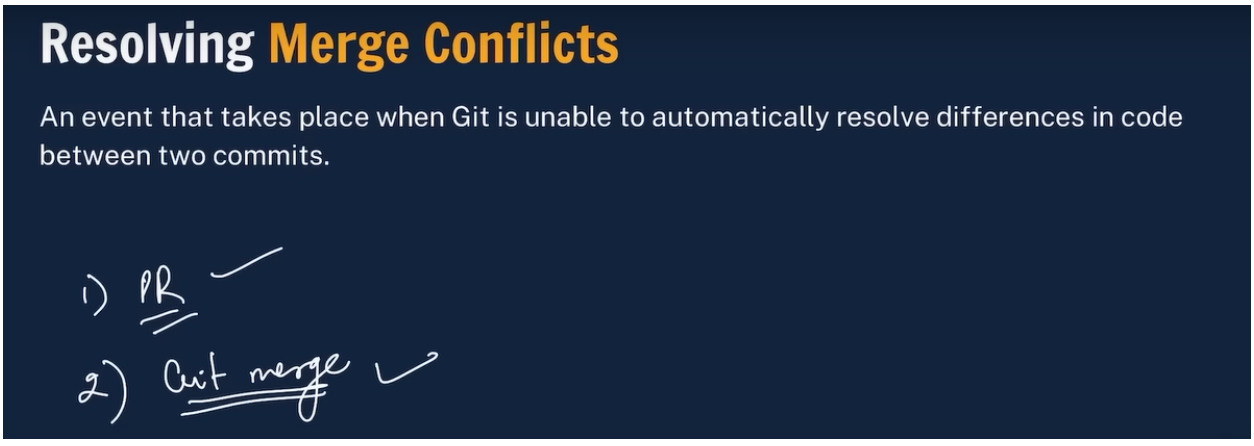
The Pull Request will then be reviewed by other developers, who can leave comments, suggest changes, and approve or reject the proposed changes. Once the Pull Request is approved and all discussions are resolved, the changes can be merged into the target branch.

Using Pull Requests helps facilitate collaboration, code review, and maintain a clean and organized codebase.

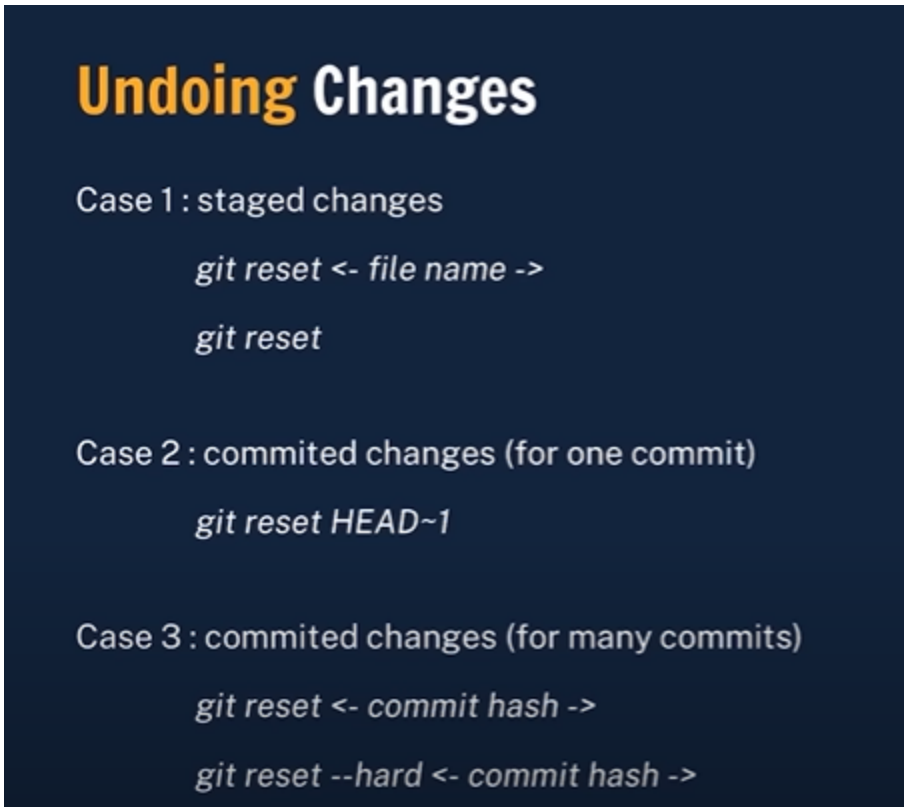
to pull the updated features to the local repository



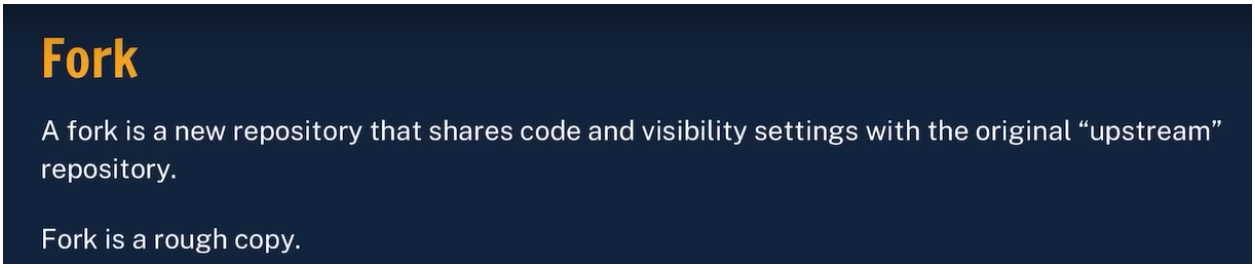
merging the conficts



undo the changes


Undoing Changes
Case 1 : staged changes
git reset <- file name ->
git reset
Case 2 : commited changes (for one commit)
git reset HEAD~1
Case 3 : commited changes (for many commits)
git reset <- commit hash ->
git reset --hard <- commit hash ->

fork


Fork
A fork is a new repository that shares code and visibility settings with the original “upstream” repository.
Fork is a rough copy.