**Git\GitHub Introduction**

Git is a distributed version control system. Git is the tool for version control while GitHub is the web-based platform that leverages git to facilitate collaboration and management.

# **Keywords**

1. **Repository (Repo)**: A repository is a storage location where your project's files and their revision history are stored. It can be either local (on your computer) or remote (on a server).

- Create a repo for each project

2. **Branch**: A branch is a separate line of development within a repository. It allows you to work on different features or fixes independently from the main codebase. The main branch is often called master or main.

- Create a branch for each new feature or enhancement

3. **Local Repo**: A local repository refers to the Git repository that exists on your local machine. It's where you can make changes, commit to them, and keep track of the project history locally.

4. **Remote Repo**: A remote repository is a version of your project that is hosted on the internet or another network. It allows multiple users to collaborate by pushing their changes to and pulling changes from this central repository. Popular services for hosting remote repositories include GitHub, GitLab, and Bitbucket.

5. **Commit**: A commit is a snapshot or checkpoint in your local repository. It records changes made to the files and directories in the repository, along with a message describing what has changed. Each commit has a unique identifier (hash). // what is hash?

6. **Clone**: Cloning is the process of creating a local copy of a remote repository. When you clone a repository, you get all the project files, branches, and history from the remote repository to your local machine.

7. **Push**: Pushing is the act of uploading your local repository changes to a remote repository. It updates the remote repo with the commits made locally.

8. **Pull**: Pulling is the process of fetching changes from a remote repository to your local repository and merging them with your local branch. It ensures your local copy is up to date with the remote changes.

9. **Pull Request**: A pull request (or merge request) is a way to tell others about changes you've pushed to a branch in a remote repository. It's a request to merge your changes into another branch (usually the main branch). It allows code review and discussion before the changes are integrated.

# **Cloning a Repo**

A black screen with white text

Description automatically generated

Firstly get your Repo URL then use this command:  
**🡪 git clone <repo\_url>**

now we cloned the repo to our local machine we want to make changes then we want to add, reset or commit.

|  |
| --- |
| Here are the stages of working with Git , we use the following command to know which stage are we in :   * **git status** |

A diagram of a product

Description automatically generated

# **ADD**

we first make changes to the working directory like adding a file or another dir like the following:

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated

* we see that the git status here specify that we are up to date

after making changes at our working dir like in here we will notice the untracked files when checking status, so we need to get to the staging area using this command : **🡪 git add <file\dirname> \ \*** (to select all new added files) , also valid: **git add <file1> <file2>…** .

A screenshot of a computer

Description automatically generated A screenshot of a computer screen

Description automatically generated

* after running the “git add” command that what you will see:

A screen shot of a computer program

Description automatically generated

# **Restore \ Unstage**

here we want to restore or undo the addition of the files to the staging area, we run the following

Command : **🡪 git restore head <filename> or git restore --staged <file>**

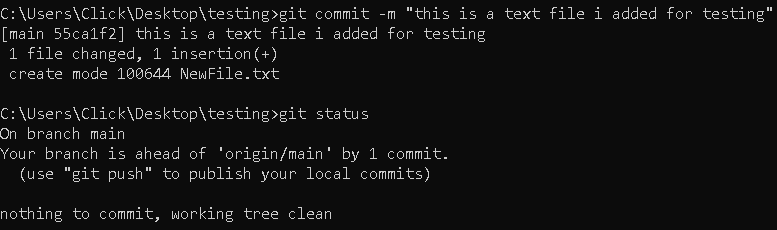
Both commands achieve the same result, but “git restore --staged” is part of the newer set of commands introduced to make Git’s functionality more intuitive.

A screen shot of a computer

Description automatically generated

# **Commit**

now we want to commit changes done in the working dir that we staged to our local repo using the following command: **🡪 git commit -m “ <msg> ”**



# **Push**

now we want to push the changes from our local to the remote repo(server\GitHub..) using the following command : **🡪 git push <RemoteName> <BranchName>**

* get the remote name : **🡪 git remote**
* get the branch name : **🡪 git branch**

**A screen shot of a computer program

Description automatically generated**

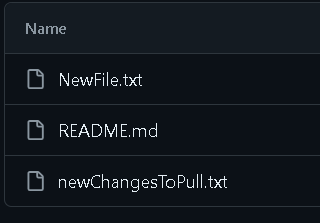
A screenshot of a computer

Description automatically generated

# PULL

let us say that a changes happened but not by your side , a teammate has done those changes, to keep your local repo updated as the remote you need to pull those changes:

* git pull <remotename> ( …… )

 A screenshot of a computer

Description automatically generated



* as we see the changes didn’t appear in our local repo ,so we need to run the pull command

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated



# Public Key for a repo!

# Git configurations!

# Create a repo for an existing local file

here we want upload an existing dir making a repo locally then pushing this repo to the remote

1. navigate to the dir you want to make the repo of
2. use the command: **🡪 git init**

**A black background with white text

Description automatically generated**

1. assume that there is a file made named “index.txt” now we add and commit

**A screenshot of a computer program

Description automatically generated**

1. now we need to go to GitHub and create a new repo

A screenshot of a computer

Description automatically generated



1. now we run this command:

**🡪 git remote add origin** [**git@github.com:AbdallahGasem/firstproj.git**](mailto:git@github.com:AbdallahGasem/firstproj.git) **, then**

**🡪 git push -u origin master ,** then you will be required to set the pass for the public key

* the -u means that you will pull any changes automatically then the push happens

# Command Aliasing

you can alias long commands using this command:

**🡪 config --global alias.<alias you want> <command> and if the command has spaces put it in a string quotes , “ <command> ” .**

**A computer screen with white text

Description automatically generated**

* the alias doesn’t remove the original command
* search more about aliases: [Git aliases](https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases)

# Branching

Get the branch name : **🡪 git branch**

Make a new branch:  **🡪 git branch <branch\_name>**

Go to a certain branch(switch): **🡪 git checkout <branch\_name>**

Delete a branch: **🡪 git branch -d <branch\_name>**

(\*using small “**-d”** it provides a warning before deletion if the branch hasn’t been fully merged. While when using **“-D”** it forces branch deletion, regardless of whether it has been merged or not. It’s useful when you’re sure you no longer need any commits from that branch.)

Make a new branch & switch to it directly: **🡪 git checkout -b** **<branch\_name>**

**Rename a branch: 🡪git branch -m <newname>**

After creating some changes in that branch you need to merge those changes to the main branch: (1st way)

1. **make changes**
2. **git add \***
3. **git commit -m “ ”**
4. navigate to main branch
5. **git merge <branch\_name>**
6. **you can delete that branch if you want**
7. **git push origin main**

A screenshot of a computer

Description automatically generated



**Hello is edited and hello2 is added**



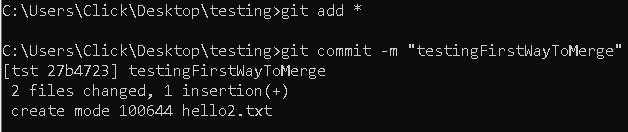
A screen shot of a computer program

Description automatically generated A screen shot of a computer

Description automatically generated

**2**

**1**

 A computer screen shot of a computer program

Description automatically generated

**4**

**3**

A screen shot of a computer program

Description automatically generated

**5**

**Second way begin at the branch you made this approach let you make a pull request before merging:**

1. **make changes**
2. **git add \***
3. **git commit -m “ ”**
4. **git push origin <branch\_name>**
5. **go to GitHub**
6. **generate the pull request**
7. **merge the pull request**

**A screenshot of a computer

Description automatically generated**

**hello3 was added**

A screenshot of a computer program

Description automatically generated



A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



1. pull changes to the local repo <after approval of the pull request>: **git pull origin main**
2. Deleting remote branches: **git push origin --delete <branchName>**
3. Delete the local branch: **git branch -d <branchName>**