**Team Iconic**

**git add**

* The git add command adds new or changed files in your working directory to the Git staging area.
* git add is an important command - without it, no git commit would ever do anything.
* git add [filename] selects that file, and moves it to the staging area, marking it for inclusion in the next commit.
* git add allows you to shape history without changing how you work.

**Command:**

* git add

**Example:**

* git add <path> (Stage a specific directory or file)
* git add test.txt

**git rm -r [file-name.txt]**

* Remove files from the working tree and from the index
* Recursively removes folders. When a path to a directory is specified, the -r flag allows Git to remove that folder including all its contents.
* Allow recursive removal when a leading directory name is given.

**Command:**

* git rm -r [file-name.txt]

**Example:**

* git rm file1.txt
* git commit -m "remove file1.txt"

**GIT PUSH ORIGIN [Branch Name] :**

The command `git push -u origin [branch name]` is used to push your local branch to a remote repository and set up tracking between the local and remote branches. Here's a breakdown of the command:

* `git push`: This command is used to send local commits to a remote repository.
* `-u`: This flag is short for `--set-upstream` and is used to set up tracking between the local and remote branches. It establishes a link between the local branch and its counterpart on the remote repository.
* `origin`: This is the name of the remote repository where you want to push your changes. By convention, "origin" is often used as the default name for the main remote repository.
* `[branch name]`: Replace this with the name of the local branch you want to push to the remote repository.

**example:**

git push -u origin my-branch

In this example, the local branch named "my-branch" will be pushed to the remote repository called "origin". The `-u` flag sets up tracking between the local "my-branch" and its counterpart on the remote repository.

After executing this command, subsequent pushes to the same branch can be done using just `git push`, as the tracking relationship has been established.

**Git Log :**

* The git log command shows a list of all the commits made to a repository. You can see the hash of each Git commit detail information such as the commit has, author name, date, and commit.

**git pull:**

* This command is used to fetch the latest changes from the remote repository and merge them into the current branch. It is essentially a combination of git fetch and git merge.
* For example, to pull the latest changes from the remote repository, you would run git pull.

**Git Checkout -- [file-name.txt]":**

* The **"git checkout -- [file-name.txt]"** command is used in Git to discard changes made to a specific file and restore it to the state it was in the last committed version. It is commonly used when you want to undo modifications you made to a file and revert it back to the version in the repository.

**git branch -m [old branch name] [new branch name]:**

* This command is used to rename a branch in Git. By providing the old branch name followed by the new branch name, you can rename the branch.
* For example, to rename a branch from "old-branch" to "new-branch", you would run
* **git branch -m old-branch new-branch**

**git push origin --delete [branch name]:**

* This command is used to delete a remote branch from the Git repository. By specifying the branch name after --delete, you instruct Git to delete the branch with the given name from the remote repository.
* For example, to delete a branch named "feature-branch", you would run

**git push origin --delete feature-branch**.

**Git Stash**

**\*git stash** is a command in Git that allows you to temporarily save your local modifications in a "stash" without committing them. This is useful when you want to switch to a different branch or pull changes from a remote repository without committing your current work.

\*Let's say you're working on a branch called feature-branch and you have some changes that are not yet ready to be committed.

**\***To save your local modifications in a stash, you can run git stash

\*$ git stash

\*Git will create a new stash entry and save your changes. It will also revert your working directory to the last committed state, allowing you to switch branches or perform other operations.

\* $ git stash apply

\*If you have multiple stash entries, you can specify the stash you want to apply by using git stash apply stash@{n}. Here, n represents the index of the stash you want to apply.

\* Once you have applied the stash, you can choose to remove it using git stash drop:

\*$ git stash drop

\*This will remove the most recent stash entry. If you have multiple stash entries, you can specify the stash you want to drop by using git stash drop stash@{n}.