**GIT Commands:**

**Git clone:** Git clone is a command for downloading existing source code from a remote repository.

git clone <https://name-of-the-repository-link>

## Git branch: Branches are highly important in the git world. By using branches, several developers are able to work in parallel on the same project simultaneously. We can use the git branch command for creating, listing and deleting branches.

## Creating a new branch:

git branch <branch-name>

To push the new branch into the remote repository, you need to use the following command:

git push -u <remote> <branch-name>

Viewing branches:

git branch or git branch –list

Deleting a branches:

git branch -d <branch-name>

## Git checkout: To work in a branch, first you need to switch to it. We use ****git checkout**** mostly for switching from one branch to another. We can also use it for checking out files and commits.

git checkout <name-of-your-branch>

## There are some steps you need to follow for successfully switching between branches:

* The changes in your current branch must be committed or stashed before you switch
* The branch you want to check out should exist in your local

**There is also a shortcut command that allows you to create and switch to a branch at the same time:**

git checkout -b <name-of-your-branch>

## This command creates a new branch in your local (-b stands for branch) and checks the branch out to new right after it has been created.

## Git status: The Git status command gives us all the necessary information about the current branch.

git status

## Git add: When we create, modify or delete a file, these changes will happen in our local and won't be included in the next commit (unless we change the configurations). We need to use the git add command to include the changes of a file(s) into our next commit.

## ****To add a single file:****

git add <file>

## ****To add everything at once:****

git add -A

## ****Important: The git add command doesn't change the repository and the changes are not saved until we use git commit.****

## Git commit: Once we reach a certain point in development, we want to save our changes (maybe after a specific task or issue). Git commit is like setting a checkpoint in the development process which you can go back to later if needed. We also need to write a short message to explain what we have developed or changed in the source code.

git commit -m "commit message"

## ****Important: Git commit saves your changes only locally.****

## Git push: After committing your changes, the next thing you want to do is send your changes to the remote server. Git push uploads your commits to the remote repository.

git push <remote> <branch-name>

## ****Important: Git push only uploads changes that are committed.****

## Git pull: The ****git pull****command is used to get updates from the remote repo. This command is a combination of ****git fetch**** and ****git merge**** which means that, when we use git pull, it gets the updates from remote repository (git fetch) and immediately applies the latest changes in your local (git merge).

git pull <remote>

## Git revert: Sometimes we need to undo the changes that we've made. There are various ways to undo our changes locally or remotely (depends on what we need), but we must carefully use these commands to avoid unwanted deletions. A safer way that we can undo our commits is by using ****git revert****. To see our commit history, first we need to use ****git log -- oneline:****

## Git merge: When you've completed development in your branch and everything works fine, the final step is merging the branch with the parent branch (dev or master). This is done with the git merge command. Git merge basically integrates your feature branch with all of its commits back to the dev (or master) branch. It's important to remember that you first need to be on the specific branch that you want to merge with your feature branch.

## For example, when you want to merge your feature branch into the dev branch:

**First you should switch to the dev branch:**

git checkout dev

**Before merging, you should update your local dev branch:**

git fetch

**Finally, you can merge your feature branch into dev:**

git merge <branch-name>

**Git init:** This command makes your current directory a Git repository and creates a directory called .git.

mkdir MyProject

cd MyProject

git init

Initialized empty Git repository in /Users/YAZAR/MyProject/.git/