**GIT COMMAND**

**git remote add origin [and the web address for your github]** ---> it used to configure connection to the project on github web;

**git push origin master** ---> push local branch with modification to github project site

**git commi**t ---> salveaza schimbarile

**git branch <name**> --->cream o ramura noua

**Git checkou**t **<name>** ---> ne muta pe ramura nou creata.

**git checkout -b [yourbranchname**] ---> creaza o ramura noua si ne muta in ea

**Git checkout bugFix; git merge master** ---> uneste ramura bug Fix cu ramura principala master

**git checkout HEAD~4** --->specify a number of commits back with ~

**git branch -f master HEAD~3** --->moves (by force) the master branch to three parents behind HEAD;

**git reset --->** reverts changes by moving a branch reference backwards in time to an older commit. In this sense you can think of it as "rewriting history;" git reset will move a branch backwards as if the commit had never been made in the first place.

Ex. git reset HEAD~1

**git revert** ---> While resetting works great for local branches on your own machine, its method of "rewriting history" doesn't work for remote branches that others are using.

Ex. git revert HEAD

**git cherry-pick --->** it takes on the following form:

**git cherry-pick <Commit1> <Commit2> <...>**

It's a very straightforward way of saying that you would like to copy a series of commits below your current location (HEAD). I personally love cherry-pick because there is very little magic involved and it's easy to understand.

Ex. git cherry-pick C2 C4