# Git

>> git --version

# setting some global configuration variables (SET CONGIG VALUES)

>> git config --global user.name 'Basir Payenda'

>> git config --global user.email 'mr.basir@gmail.com'

>> git congi –list

# to get current user and user’s email

>> git config user.name

>> git config user.email

>> ls

>> ls -la # lists all the file in the directory

# help

>> git help config

>> git config --help

>> git add --help

# initilize a repository from existing code

>> git init

>> ls -la

>> [rm -r](https://www.quora.com/What-does-delete-folders-files-recursively-means)  # command deletes the folder recursively, even the empty folder.

>> rm -f  # command removes ‘Read only File’ without asking.

>> rm -rf .git # if you want to delete .git directory

# Staging files

>> git add –A

>> git add index.html

>> git add . # add all files in staging area

>> git reset calc.py # get calc.py out of staging area

>> git rm --cached fileName.ext # get file.txt out of staging area

>> git diff # shows changes that we made

# remove files from staging area !important

>> git reset

>> git reset calc.py

>> git rm --cached fileName.ext # get file.txt out of staging area

# commit

# commit all with a comment to describe what you have committed!

>> git commit -m 'Initial Commit'

>> git log # to see history of commits you made

>> git log --oneline # shows commit history in one line

# [Undoing things](https://www.youtube.com/watch?v=RIYrfkZjWmA&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR&index=7)

1. checkout: moving from one commit to another commit
2. revert: deleting a commit
3. reset: deleting a chain of commits

# 1. checkout

>> git checkout b254c66 # b254c66 is the code of the commit

# 2. revert

>> git revert b254c66 # deleting this commit

# 3. reset

# deletes a chain of commits, --hard will reflect those changes back to our file

>> git reset b254c66 --hard

## # git reset --hard, reverts back to the stage that it was but it is not going to # clean untracked files, so to clean those untracked files run:

>> git clean -df

# **d** git rid of any untrucked directory

# **f** get rid of any untracked files

# what if you run get reset --hard, and now you regret. So to recover it run

>> git reflog

and then choose your ideal commit and checkout to that

# [Working with branches](https://www.youtube.com/watch?v=QV0kVNvkMxc&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR&index=8)

We don’t want to miss with master branch because it is going to be our production code.

>> git branch feature-1 # add a new branch called feature-1

>> git branch –a # lists all branches

# switching to feature-1 branch from master branch

>> git checkout feature-1

# switching to master branch from feature-1 branch

>> git checkout master

# creating and switching to a branch simultaneously

>> git checkout –b feature-1

# delete a local branch

>> git branch -d feature-1

# push a local branch to remote branch

>> git push –u origin feature-1

# delete a remote branch

>> git push origin --delete feature-1

# merge feature-1 branch with master branch

>> git merge feature-1

# list all branches that are merged with master branch

>> git branch --merged

Git Pull

# before pushing your commits to remote repo, **first pull then push**.

# **Pull** is a Git command used to update the local version of a repository

# from a remote. We must pull first before push because there must be

# so many develpers pushing their codes and our local repo is now not in # sync with remote repo.

>> git pull origin master

# Working with GitHub

# how you push local repo to remote repo without any alias of repo url

>> git push <https://github.com/basirpayenda/delete-it.git> master

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# create an alias for above url so we don’t have to type it again & again

>> git remote add origin <https://github.com/basirpayenda/delete-it.git>

# remote add - means add a remote repository

# origin - is the alias for above url

# origin = <https://github.com/basirpayenda/delete-it.git>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# now that we have created an alias for that url we can type.

# notice that **origin** is an alias for above url

>> git push origin master

>> git clone <url>

# Collborating on Github

net ninja 10 and 11 videos

# [**Fixing Common Mistakes and Undoing Bad Commits**](https://www.youtube.com/watch?v=FdZecVxzJbk&list=PL-osiE80TeTuRUfjRe54Eea17-YfnOOAx&index=2)

**Oops… I spelled that last commit message wrong**

# in this commit we meant to commit 'Multiply feature added' but we made

# a mistake and wrote 'Divide feature added'now how to undo this

>> git commit -m 'Divide feature added'

# then we can ammend it like

>>> git commit --amend -m 'Multiply feature added'

**How to move a commit from master branch to another branch?**

if by mistake we added a commit in master branch however we meant to add it in another branch, use cherry-pick to move that commit to another branch.

# first checkout to your targeted branch and use cherry-pick with hash

# of that commit to move from master to that ideal branch

>> git cherry-pick **9d22701**

# keep in mind that cherry-pick does not delete it from master branch

# so in order to delete it from master branch use ‘reset’

## Oops… I forgot to add a file to that last commit

>> git add file.py

>> git commit --amend

# After writing commit message, a new window will pop up. just

# press Esc Button and then write :wq or :wq! and then Enter

**how to find difference between two commits?**

>> git diff **9d22701 5de52789**

## Oops… I made a spelling mistake in my branch name

we made a mistake renaming ‘feature-branch’ to ‘future-branch’. Now how to rename it.

git branch **-m** future-brunch feature-branch

If you've already pushed this branch, there are a couple of extra steps required. We need to delete the old branch from the remote and push up the new one:

git push origin --delete future-brunch

git push origin feature-branch

## Oops… I did it again

This command is for when everything has gone wrong. When you've copy-pasted one too many solutions from Stack Overflow and your repo is in a worse state than it was when you started. We've all been there.

git reflog shows you a list of all the things you've done. It then allows you to use Git's magical time-traveling skills to go back to any point in the past. I should note, this is a last resort thing and should not be used lightly. To get this list, type:

git reflog

Every step we took, every move we made, Git was watching us. Running that on our project gives us this:

3ff8691 (HEAD -> feature-branch) HEAD@{0}: Branch: renamed refs/heads/future-brunch to refs/heads/feature-branch

3ff8691 (HEAD -> feature-branch) HEAD@{2}: checkout: moving from master to future-brunch

2b7e508 (master) HEAD@{3}: reset: moving to HEAD~

3ff8691 (HEAD -> feature-branch) HEAD@{4}: commit: Adds the client logo

2b7e508 (master) HEAD@{5}: reset: moving to HEAD~1

dfa27a2 HEAD@{10}: commit (amend): Added contributing info to the site

700d0b5 HEAD@{11}: commit: Addded contributing info to the site

efba795 HEAD@{12}: commit (initial): Initial commit

Take note of the left-most column, as this is the index. If you want to go back to any point in the history, run the below command, replacing {index} with that reference, e.g. dfa27a2.

git reset HEAD@**{**index**}**