Definition of keywords/phrases in VC with Git and GitHub

Keywords	Description
Origin	Is one default remote repository, usually there is one.
HEAD	the last commit of the active/current branch
main	The default main branch,
Branch	An independent line of development
index	The proposed next commit—staged file
Repository/repo	Centralized storage with the revision history of all related & specified files.
Workspace	copies of the files in the local machine of the user
Tag	Represent a particular snapshot of a project at a given time.
Pull/Update	Update the local working copy with the latest changes
Commit/Check-In	Store a change in the central Version Control storage.
fetch	download objects and refs from another repository
pull	Fetch from and integrate with another repository or a local branch
Push	Used to send commits to the determined central repository.
Conflict	A situation where 2 developers try to commit changes in the same region of the same file.
Merge	Combining changes in different working copies to the same file in the repository.

Moving to the working directory

Keywords	Description
pwd	Check your present working directory(pwd)
cd ~/your folder dir	Your folder directory after pwd
cd drag your folder	If you are using mac you can drag the folder there
mkdir <foldername>	Create folder in the directory
ls	List the files and folders in the current folder
Ls -la	List the files and folders in the cd including hidden

Git installation check

git --version	Check git version if installed already
git	List git details if installed already

Configure Name & Email for commit and push to remote Git

git config --list	
git config --list --show-origin	

git config --global user.name "Your Name" #global indicates applicable across the system.
git config --global user.email email@example.com

.gitignore	Tell git which files/folders it SHOULD NOT track
rm -rf .git*	Remove git from project
git config --global alias.shortcut <command>	Set up git alias
git config --global alias.s "status"	**git s = git status**
git config --global alias.ch "checkout"	

git config --global alias.co "commit"	
git config --global alias.br "branch"	

Git initialization

git init	Create an empty Git repo or reinitialize an existing one.
git init c:\local_repo_name\repo_folder_name	
git status	Show status of working directory and staging area
git status –short or git status -s	Shows ??-Untracked, A-added to stage, M-Modified D-Deleted files
git clean	Remove the untracked files
git clean -n	
git clean -f	
git clean -dn	
git clean -df	

to exclude all.DS_Store files from your future repositories

```
echo .DS_Store >> ~/.gitignore_global
git config --global core.excludesfile ~/.gitignore_global
```

Stagging-making change ready for commit

git add <folder>/	Stage all files inside the folder(and subfolders)
git add <filename>	Stage individual file
git add .	Stages new and modified files, without deleting.
git add -a	Stage all files
git add -u	Stages modified and deleted, without new.
git add c:\local_repo_name\repo_folder_name/sub_folder_name	

Undoing change and staging

git reset <file/folder>	Release file from staging back to working
Git reset .	Reset new and modified file
git checkout -- <file/folde	Removing change made to file/folder
git checkout -- .	Remove change made to new and modified files
git rm	Remove tracked files from the staging area and (but not or) working directory
git rm file_name_1 file_name_2	git rm file_name_1 file_name_2
git rm --cached file_name_1	
git rm -r folder_name	

Committing-after staging is committing/creating point of save

git commit -m "message"	
git commit -a -m “message”	commit git without staging
git commit -m "message" –amend Update previous commit instead of creating new one	

Navigating between commits

git checkout <commit_hash|branch_name> View a previous commit	
git checkout <hash|branch> <file|folder> Restore the contents of files back to a prev commit	
git checkout <hash|branch> file Restore file	
git checkout <hash|branch> folder/ Restore all files in folder (& subfolders)	
git checkout <hash|branch> . Restore all files in project	

Log-to see history of commit and branches

git log	View the commit history of a branch
git log -p -3	Shows details of changes
git log --all	Show all commits (not just current branch)
git log --all --graph	Show branching visually in the command line
git log --oneline	
git log --oneline --all --graph	

git log --oneline --all --graph --decorate	
git log --pretty=format:"%cn committed %h on %cd"	Customise git log format
Filter git log output	
git log --after="yyyy-mm-dd" --before="yyyy-mm-dd"	
git log --author="author1\\|author2"	

git log --grep="commit_message_key_word"	
git log -S"file_content_key_word"	
git log branch_name_1 branch_name_2	
git diff	Inspect changes in a repository
git diff file_name	
git diff commit_id_1 commit_id_2	
git diff branch_name_1 branch_name_2	
git blame Review a file’s modification history, often used together with git log	
git blame file_name	
git blame -L 1,10 file_name	
git reset	Move both current HEAD pointer and branch ref pointe
git reset --soft commit_id	
git reset --mixed commit_id	
git reset --hard HEAD~1	remove the latest local commit.
git reset --hard HEAD~3	Remove the latest three local commits
git push origin HEAD --force	Delete remote commit-push local change
git reset --hard commit_id/hash	
git reset commit_id file_name	
git reset --head	https://hackernoon.com/how-to-delete-commits-from-remote-in-git
Git revert	Undo changes to a commit history
git revert HEAD	
git revert commit_id	
git revert -n HEAD	
git stash	Save and hide committed and uncommitted changes
git stash save "stash_message"	
git stash -u	
git stash -a	
git stash -p	
git stash list	
git stash show	
git stash pop	
git clean -n -d -x	
git tag	Snapshot specific points in a repository history
git tag	
git tag -a tag_name -m "tag message"	
git tag -a tag_name commit_id	

GitHub

git remote	Manage remote repositories’ information stored locally
“git remote rename origin new_name“	To rename default remote repo name to your choice
git remote add <remote_name> <url> Link local repo to a remote repo and git it a name	
git remote	List all remote repositories that are linked
git remote -v	List all remote repositories (but with more detail)
git remote remove <remote_name> Removes a link to a remote repository	
git remote remove origin	
git remote rm repository_name	
git config --global credential.username <username> Configure your GitHub username so you can get access to your Github repository	
git push <remote_name> <branch>	Upload local repos content to a remote repository
git push origin main	
git push -u origin local_branch_name	
git branch	Shows a list of available branches
git push <remote_name> <branch> --set-upstream Sets up a shortcut for this branch and remote repository	

git push origin main --set-upstream Next time you are on the main branch and you run git push, it will automatically push the main branch to origin

git push <remote_name> <branch> -f	Force push to remote repo -it will overwrite what is in the remote repo
git fetch origin master	Download content from remote repository, but doesn't force the merge

Cloning

git clone <url>	Use http url create a copy of remote repo on your local machine
git clone <url> <folder_name>	Clone repo and git it a name
git fetch	Update all remote tracking branches
git pull <remote_name> <branch> git pull origin main	Update local repo with latest update from remote repo

git pull origin main --set-upstream set-up upstream shortcut so that the next time you are main branch, just run git pull

Branching- *manage branches of a repo, e.g. create, list, rename and delete*

git branch	List out existing branch
git branch <branch_name>	Creates a new branch
git branch feature1	Create a new branch named feature1
git branch -a	
git branch -m renamed_branch_name	
git branch -d existing_branch_name	
git checkout <branch_name>	Switch to a different branch and start working on
git checkout <branch_name>	Switch to a different branch and start working on it
Git checkout out feature1	Example for branch named feature1
Git checkout -b [branchName]	Switch from HEAD to new [branchName] and switch to it
git checkout -b emergency_fix	Emergency branch
git branch -D <branch_name>	Delate a branch
git branch -D feature1	Delete the feature1 branch
git rebase	Move one branch to the tip of another branch
git rebase base_branch_name	*do not rebase shared branchs
git reflog	View the HEAD change history of all local branches
git reflog	
git reflog --all -3	

Merge

git merge [branchName]	Merge current branch [Head] with [branchName]
git merge [branchName] -m "Message"	Include merge comment/message
git merge [localBranch1] [localbranch2]	Join two branches together