**GIT AND GITHUB COMMANDS**

**Note:** **How we can change The Text Editor in Git Bash.**

First of All **download and Install** your favorite Text Editor, Then go to **C drive** >> check in **Program Files (x86) or Program Files** and find the **Text editor folder**, open it and **copy the Path** of the text editor >> go to **my computer properties** >> **Advanced system settings** >> **Environment variables** >> **Find Path** >> **Edit the Path variable** >> put **semicolon** at the end of the path and after semicolon **Paste the Text Editor Path** >> **Save everything**.

git init This command will initialize the git repository

git clone (rep\_link) This will clone the repository already created in your GitHub account

ls –l .git This will list all the files inside the .git directory

ls –la This will list all the files inside your working tree

diff This command will show the difference between files but not the explanation

diff –u This command will show the differences between files the explanation

patch This command will create output from the diff –u files

git add file\_name This command will add the untracked files to the staging area and it will pass the files that we want as a parameter, this file will immediately move your untracked files to the staging area and will change the state of the file from modified stage to the staged stage and make ready the files to get committed

git add \* This command will track all the files in a directory.

git diff –p This will show us the changes that has been made to the file and will allow us to decide whether the file need to be committed or not.

git diff –staged With this command we can see the changes that has been made but not committed

Definition: Staging Area: A file maintained by the Git that contains all of the information about what files and changes are going to go into your next commit. The staging area is also known as Index.

git status This command will show the current status of the working tree and the pending changes

git commit This command will commit all the files to the git directory. We can add –m to add message for our understanding

git commit –graph –oneline This command will show commit history in one line, this is a very specialized way of getting and showing commit history

Info: Git Project Any Git Project consists of three sections, the git directory, the working tree and the staging area, The git directory contains the history of all the files and changes, The working tree contains the current stage of the project including any changes that we have made, and the staging area contains the changes that have been marked to be included in the next commit.

git config –l This command will list all the files of your configuration file. It includes a lots of information’s but specially look to the user name and user email lines. The user name and user email will appear in public commit logs when you used a shared repository

chmod +x filename This command will execute the given file

git log The command will list all the commit messages of the project or directory

git log –p This command will show the all the comments with more details and it will show what changes has been made to the file. The –p flag is short from patch which will show more details of what changes has been made to files.

git log –stat This command will show different states of the files like how many changes has been made to the file and how many lines are added or removed from a file.

git show d60bd7d830ef23634f9d170d0df770da8cdc395a This command will show the comments with more details and it require comment id as a parameter as given in the example. You must copy the id from git log command and copy the id and give it as parameter to the git show command.

It is equivalent to the diff –u command, it will show the added line with green and + sign and removed line with – sign.

git diff This command will show the changes that had been made to the files. It will show what files has been added and what files had been removed from a file.

Note: Skipping staging we can Skip the staging step and directly commit files by using ‘git commit –a –m “Your Message”’ command

git rm filename This command is used to remove files from the directory, it will first stop the git to tacked the files and then remove it from the directory.

git mv oldfilename newfilename This command is used for renaming and moving files in a directory. Eg git mv myscripts2.py myscripts3.py

NOTE: gitignore The .gitignore enable us to ignore the files so that it won’t get add noise to the result. Inside this file we specify the roles which tell git which files to ignore in the current repo. To create .gitignore file we use the following command echo .DS\_STORE > .gitignore

To see that the .gitignore file is generated or not we use ls –la command

**Revert Changes**

**Reverting Unstaged Changes**

git checkout file\_name This command will revert changes made to a file, this command will restore the latest snapshot which could be either committed or before staged. it will revert changes and will get you the earlier version of your file.

git restore file\_name This command serve the same purpose as git checkout command, it will revert changes made to the file and will get you the old version of your file.

git checkout –p file\_name This command is used to checkout individual changes instead of the whole file by using the git checkout with –p flag. This will ask you change by change if you want to go back to the previous snapshot or not. These three commands are used to revert unstaged changes.

**Reverting staged Changes**

git reset HEAD file\_name This command will revert changes after staging. This command will used the HEAD, this represent the current snapshot of the file and will revert the changes after staging (git add \* ). Simply this will revert tracked files to untracked files 😀. The reset is the counterpart of add, with add we can stage the file and with reset we can unstage the file.

git reset –p HEAD file\_name This command will reset or remove a specific change you want to reset.

**Note :** Before staging to revert changes we use git checkout filename command, after staging we use git reset HEAD filename command and after commit we use git commit –amend command to overwrite or update the previous comment and we use git revert HEAD filename command to roll back to the previous version.

**Note:** The Git uses SHA1 algorithm as a commit ID which is 40 characters long.

**Rolling Back Changes:**

git revert HEAD file-name This command will create a new commit that is the inverse of the bad commit, We can reverting the changes by using the HEAD alias.

Identifying a commit The git log -10 command will show the latest 10 commits with their ID’s, you can use git show 3122d68e5617d7bc93d96ae4843e1aa99ef7b559 and hit enter. It will show the commit details. You can also provide the first 4-6 characters of the commit id to git show command, the git will display the complete details of the commit. Eg git log -10, git show 3122d6.

GIT BRANCHES:

git branch This command will list all the branches

git branch branch-name This command will create a new branch in the directory

git checkout branch-name This command is used to switch to another branch, we use git checkout branch-name command.

git branch –b new-branch-name this command will create a new branch and immediately switch to it. You can show the latest commits by using git log -2 you will see your latest branch and master branch

Note: When we switch branches the git also change the files of the branch.

git branch –d branch-name This command will delete the given branch

git branch –D branch-name This command will delete the branch that have not been merged and have changes.

git revert 3122d6 ( first few chars of commit id) This will revert (roll back) the commit id provided in the command.

**Merging**

git merge branch-name This command will merge the master branch with other branch which we provide in the command

**Issue: Resolving Merge conflicts** sometime we want to merge different branches but there are changes in files of both the branches, so it create merging conflicts. To solve that conflicts we follow different approach first of all if merge conflict happened, then use git status command to know more about the conflict, the open that file in text editor and eliminate the conflicts, save the file, staged the file, commit the file and try again to merge the merge conflict should be solved.

git merge –abort This command will thwart the merge

git clone repo-address This command will make a copy of the remote repository to our local machine.

Get push This command will push our changes to the remote repository

**Note about Git and GitHub credentials:** The git always wants users name and password whenever you try to clone or push changes to the remote repository, to avoid this, git offers two ways, the first way is to create SSH key pair and store it in your profile and the other way is credential helper. The credential helper is built-in to the git, we just need to enable it, to enable it we use **git config –global credential.helper wincred** command, after giving this command the git will ask you once more the user name and password, just provide it and it will be saved there.

git pull This command full all the contents and changes from the remote repositories made by other developers.

git remote –v This command is used to check the remote repository configurations. This will gives you the URL’s associated with the remote repository. There will be two URL’s, one will be used to fetch data from the remote repository and the other will be used to push data to that remote repository.

git remote show origin This command will give us more information about the remote repository.

git branch –r This command is used to show the remote branches that the git is using while working with remote repository.

$ git branch -r

origin/HEAD -> origin/master

origin/master

**Question:** if we want to make changes to the remote branch, what must we do?

**Answer**: First we pull the remote branch, then we merge it with the local branch, and then push it back to the origin(master).

git fetch This command is used to copies the commits done in the remote repository to the remote branches, so that we can see what other people has committed. It only fetches the updates from the remote repository and we need to run the merge command in order to merge it with our local repository.

git log origin/master This command will show the commits in the remote repository.

git merge origin/master This will merge the remote origin branch to the local master branch.

git pull git pull command fetch and merge the updates and changes into our local repository from the remote repository at once.

git log –p -1 using this command we can see the changes made to the files.

Note: If your friend have made changes to the remote repository and added some branches, you should pull the changes made to the file, but if your friends made new branches to the remote repository, then you must create those branches in your local repository.

git remote update This command will pull the changes from all the branches to your local git repository and you simply need to checkout or merge the contents as needed.

git log –graph –oneline –all The command will show the all the comments in one line and in linear

**Question:** What should you do with the <<<<<<<, =======, and >>>>>>> conflict markers when resolving a merge conflict?

**Answer**: Remove all of the conflict markers and only leave the code as it should be after the merge.

**Note:** if you want to push changes to the remote repository, and the remote repository already have new changes, then your push command will fail, and git will give you an error, that push has rejected, so your first pull all the new changes to the local repository and then try to push the changes to the remote repository. Sometime this lead to the merge conflict, resolve the merge conflict and try again.

git log –p origin/master This command will show the latest comments with lots of more information and changes made to the file.

git push –u origin branch-name This command will push the branch mentioned in the command to the remote repository. For example if we create some local branches and make changes to the branch files, then we can push the branches to the remote repository by using **git push –u origin branch-name(refactor)** command to push the mentioned branch to the remote repository.

git push –delete origin branch-name This command is used to remote or delete the remote branch in the remote repository

git branch –d branch-name This command will delete or remote the local branch for the local repository.

git rebase branch This command is used to rebase your branch commit history on top of the branch.

git rebase origin/master This command is used to full changes from the remote repository committed by your colleagues. If your colleagues made changes to the remote repository, and your also made some changes to the local branch, to get the changes made by your colleagues, we use git pull command to get the changes in our local branch with a three way merge but in this example we will try a different approach, we will use git fetch command to fetch all the changes from the remote repository to the local branch, but not merged that changes to the master branch, we will use the git rebase origin/master command to rebase our changes those made by our colleagues in order to keep history linear. Some time we got conflicts and we need to solve it, looking at the history log and opened it in the text editor. Fix the conflict save and exit the file. Now stage the file by using git add firstfile.py and continue the rebase by using git rebase –continue. In this example we use the fetch, rebase and push workflow to merge our changes to our collaborative changes while keeping our history linear.

git rebase –continue The command will continue the rebase process

ssh-keygen –t rsa –C “your id or email” This command will create a ssh key, just provide the path to store that key and enter. You can access the key by visiting that directory by using cd directory-path, list all items there, and you can see the ssh key by using cat ssh-key.pub and enter, copy the code, and go the your GitHub on web, go to settings and open ssh key options and create a ssh key, give it a name and paste the code there and save.

cat ssh-key-name.pub This command will show the ssh key. You can copy the key and paste it in your GitHub settings.

Rm –f ~/.ssh/ssh-file-name This command will delete the ssh files

rm –f ~/.ssh/id\_rsa\* This command will also delete the ssh key

Note: When you create your ssh key and save it in your GitHub settings, whenever you try to clone your repository using ssh key on your local machine, then it will stop you cloning the repository to your local machine, it is because your ssh key is not added to the ssh repository, so you need to add it to the ssh folder by using the following command ssh-add ~/.ssh/ssh-file-name

ssh-add ~/.ssh/ssh-file-name This command will let you to add your ssh file to the ssh folder on remote repository.

git gui& This command will open the git GUI.

gitk This command will open the commit history, with older and newer versions in GUI form. You can find more git gui clients and the most popular is sourcetree client.

git show –s –pretty=raw commit first 6 chars This command will show you that commit with a lots of more details, like commit, commit message, tree, author, parent commit and a lot more.

git log –pretty=online The command will create the list of latest comments in a very beautiful order of one line.

git checkout . The git check dot command will revert or restore changes in all the files of the current folder.

git checkout – filename The command will revert changes made in the file.

git clean –xdf The command git clean –xdf will remove all the unstaged or newly created files by force. Here the x means ignore all rules, d means directory, and f means apply changes by force.

git reset – file-name This command will revert the staged file back into unstaged file.

git reset HEAD^ This command will remove the latest commit and move its files to the unstaged area. We can write this command also like this git reset HEAD~1 to revert the latest commit back to unstaged area.

git reset –soft HEAD^1 This command will reset or revert files back in unstaged area (index) after committing without losing file changes.

git reset –mixed HEAD^ This command will reset or revert files to the main file system without losing the file contents.

git reset –hard HEAD^ This command will reset or revert files to the main file system with losing all the changed file contents after committing.

<https://archive.org/details/dirilis_s3ep3>

<https://archive.org/details/dirilis_s3ep4>

**Ertugrul Series**

**Command Line Essentials: Git Bash for Windows Basic Commands**

**Directory Listing**

pwd , ls, ls-l, ls –la, git ls-files

**Navigation**

cd folder-file name. cd .. file-name, which command-name, cd~, cd /c/windows/users/desktop/, cd ../../..

**Printing something**

echo “strings”, echo $path, echo “some string with $path”,

**Reading the contents of the files**

cat filename, less filename

**Create rename(move),delete files**

Create(touch filename), rename( mv firstname newname), remove (rm filename) note: touch filename command also change the timestamp of the file if exist

**Create, delete, recursive delete folders**

mkdir folder name, mkdir otherproject, rmdir otherproject, mkdir –p projects/nested-project/nested-project/nested-project, rmdir projects(will give error), rm -r –f (this will forcibly remove all the folders and sub folders)

**Clearing and exiting the screen and program**

clear, exit

**Outputting contents to files**

“some string” >> output.txt, “some other string” >> output.txt, the >> double greater than signs will add the contents to the end of the file

“some string” > output.txt The single > greater than sign will overwrite the contents with the output.txt files

Sending the Output of most command to a text file we use > or >> sign with ls –al Eg **ls –al > listing.txt** so the listing.txt has the OR less abc.txt > text.txt

This will transfer the contents of one file to another file

ls –al > filename.docx

less text.txt > output.txt

**How to write script and execute it in git bash**

**which bash** copy the path of the bash

**notepad++ script1.sh** The editor will open

**#!/usr/bin/bash**  then add shebang like

**echo “Hi every one this is my first script in Bash”**  then write your script

**save and Exit the file**

**chmod +x script1.sh** execute the file

**./script1.sh** run the file if already executed

**Customize Bash Environment**

We Can customize the bash environment for easiness yaani we can use the abbrivations for commands like for notepad++ we can use npp, for ls –al we can use ll.

To customize our environment we use **.bashrc** command

**Notepad++ .bashrc**

Put you credentials like this

**alias npp=’notepad++’**

**alias gts=’git status’**

**alias gtaa=’git add \*’**

**alias gta=’git add’**

**save and exit the notepad file**

**source .bashrc**

Save the file by pressing Ctrl-o, Enter key, and Ctrl-x.