git commands

1. Cd

Is used to change from present directory to desired directory.

Syntax: - cd desired path

Eg:1 cd F:

Changes to F drive.

2. cd d:/projects/mini

Changes to mini directory.

Note: to go home directory use cd ~ command.

Cd .. for one step back

2)mkdir

Is used to create a folder.

Syntax: - mkdir foldername.

Eg: mkdir projects

Creates projects folder in the current path.

3)ls

List the all files and folders in the current location. (expect hidden)

ls -al list all the files and folders including hidden files.

4)clear

Clear the screen

5) config

usage: git config [<options>]

Config file location

--global use global config file

--system use system config file

--local use repository config file

--worktree use per-worktree config file

-f, --file <file> use given config file

--blob <blob-id> read config from given blob object

Action

--get get value: name [value-pattern]

--get-all get all values: key [value-pattern]

--get-regexp get values for regexp: name-regex [value-pattern]

--get-urlmatch get value specific for the URL: section[.var] URL

--replace-all replace all matching variables: name value [value-pattern]

--add add a new variable: name value

--unset remove a variable: name [value-pattern]

--unset-all remove all matches: name [value-pattern]

--rename-section rename section: old-name new-name

--remove-section remove a section: name

-l, --list list all

--fixed-value use string equality when comparing values to 'value-pattern'

-e, --edit open an editor

--get-color find the color configured: slot [default]

--get-colorbool find the color setting: slot [stdout-is-tty]

Type

-t, --type <> value is given this type

--bool value is "true" or "false"

--int value is decimal number

--bool-or-int value is --bool or --int

--bool-or-str value is --bool or string

--path value is a path (file or directory name)

--expiry-date value is an expiry date

Other

-z, --null terminate values with NUL byte

--name-only show variable names only

--includes respect include directives on lookup

--show-origin show origin of config (file, standard input, blob, command line)

--show-scope show scope of config (worktree, local, global, system, command)

--default <value> with --get, use default value when missing entry

To open an editor just type the editor name.

Eg: notepad++ (opens notepad++)

Configure notepad++ as default editor in Gitbash

Creating Alias:

Step1: Cd ~

Step2: Notepad++ .bash\_profile

Opens notepad++ with file name .bash\_profile.enter the fallowing code in the file.

alias npp =” notepad++.exe –multiinst –nosession”

step3:

git config --global core. editor "notepad++.exe -multiInst -nosession"

commands used for repository

\*\*process of uploading local project to github:

step1: ->open GitBash.

->change directory to your project location (eg: cd C:/Users/LENOVO/eclipse-workspace/inetbanking\_v2)

step2: ->use command

syntax: git init

(creates empty repository [.git hidden file in your pwd (present working directory)]

Syntax: git init <projectname>

(create empty repository in <projectname> folder. i.e., in <projectname> folder .git file created..<projectname>is in your pwd)

git init <your repository name>

step3: ->use command

git add –A or git add .

(add all the files to staging area)

step4: ->use command

git status

(view all the files which are going to be staged to the first commit.)

step5: -> use command

git commit -m "message" (add to .git folder)

(git commit uses '-m' as a flag for a message to set the commits with the content where the full description is included, and a message is written in an imperative sentence up to 50 characters long and defining "what was changed", and "why was the change made".)

step6:->use command

syntax: git remote add <short\_name> <url>

short\_name: in general, we give “origin”

url: path oy the remote repository

(Git remote command acts like a border, and If you need to connect with the outside world, you have to use the Git remote command. This command will connect your local repository to the remote.)

eg: git remote add origin "https://github.com/shaikshavaliDudekula/inetbanking.git"

step7:->use command

syntax:  git push -u <short\_name> <your\_branch\_name>

git push -u origin master

(is used to push localcontent to github)

step8: enter github credentials

Getting project from github to local:

1)rm –rf <filename>

Delete the file.

Eg: rm –rf projects (delete project and files init)

Or

git rm <your\_file\_name>

Sometimes you need to delete files from your codebase, and in that case, you can use the Git rm command.

It can delete tracked files from the index and the working directory.

2)clone

To copy project from GitHub to local.

Syntax: git clone <repository path in github>

Eg:

git clone “<https://github.com/shaikshavaliDudekula/inetbanking.git>”

project created with the github repository name (inetbanking)

to rewrite our won project name instead of github repository name

syntax; git clone <repository path in github> <projectname>

eg:

git clone “<https://github.com/shaikshavaliDudekula/inetbanking.git>” myproject

project created with name myproject in local.

After modification uploading to github:

Step1: add the modified file

Syntax: git add <modified/new file> or git add \*

Eg: git add readme.txt

(here I am uploading readme.txt file which is modified or crated)

git add <filename> (it will add a single file to your staging area)

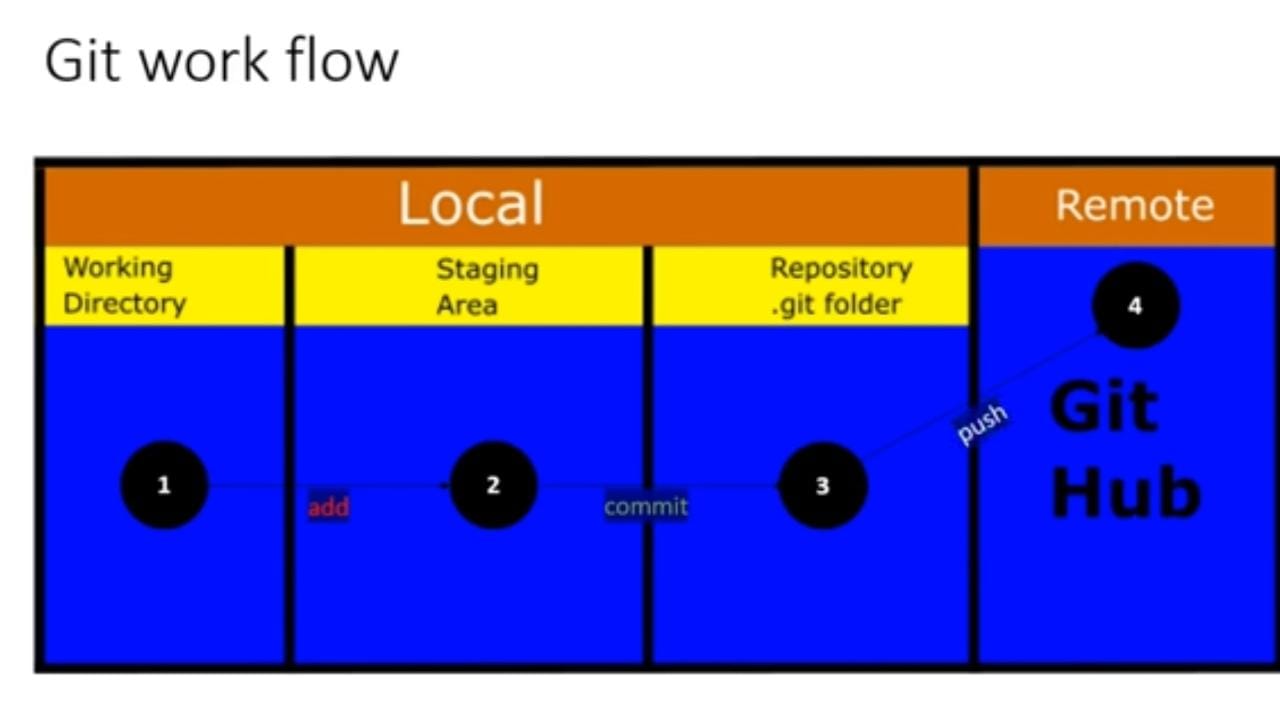
git add \* (this option will add all the modified and new files to the staging area)

Step2: commit

Eg: git commit –m “message to be posted”

Step3: git push –u origin master

(uploaded successfully to github after entering credentials)



Untracked and tracked files

1) to remove staged files use command

syntax: git rm –cached <filename>

2)to make short the status information use command

Syntax: git status –s

?? - untracked

A - staging Area

M - modified files

