GIT PRACTISE AND ITS NOTES PART2

* Download git, provide path of git bash with the main github account
* sudo yum install git (when you run **sudo yum install <package\_name>**, you're telling the system to use **yum** with administrative privileges to install the specified package. This ensures that the package manager can make the necessary changes to the system's package database and file system.)
* **git –version**
* git config user.email --> to check the email of the user or to configure the mail in .git file
* git config user.name --> to check the username
* git config user.email “vaibhawpandey0405@gmail.com” --> to login the particular mail in git
* git config user.name”vaibhawpandey” --> the user name configuration
* git config --global --unset user.name”Vaibhaw Pandey” --> to logout at global level
* git config --global --unset user.email”vaibhawpandey0405@gmail.com” --> logout mail at global level
* git init

# To make entry git on file where have to work

* git status

# To check the status

* git add <file1\_name.txt> <file2\_name.txt> ...

# To add file so it start tracking

* git add .

# To add files to staging area

* git add\*

# To add all files to staging area

* git commit -m "commit\_message"

# To commit changes

* git commit -m -a “legal message”

# By passing the file (only for once tracked file)

* git log

# To view the commit history

* git log –oneline

# To view the commit history in a more condensed format

GIT Diff Commad:

* Diff command is used in git to track the difference between the changes made on a file.
* Diff command takes two input and reflects the difference between them.
* To check the changes between the working area and the staged area:

Git diff <FILE NAME>

* To check the changes between the stage area and the repository area:

Git diff-staged <FILE NAME>

* To check the changes between the working area and the repository area Git diff head <FILE NAME>
* NOTE: higher stage file is denoted by a—and lower stage file is denoted by b++

REPOSITORY AREA

STAGGING AREA

WORKING AREA

* # To create a local copy of the repository
* git clone <repository\_url>
* to by pass
* # To add a remote repository
* git remote add origin "http\_url\_of\_repo"
* # To push changes to the remote repository
* git push origin master
* # To pull changes from the remote repository
* git pull origin master
* # To create a new branch
* git branch <branch\_name>
* # To delete a branch
* git branch -D <branch\_name>
* # To switch to a branch
* git checkout <branch\_name>
* # To view repository history
* # To stash changes
* git stash
* # To retrieve stashed changes
* git stash pop
* # To revert a commit
* git revert <commit\_id>
* # To check the difference between two versions of a file
* git diff <commit\_id\_x> <commit\_id\_y>
* # To show information about a Git object
* git show <commit\_id>
* # To add all changes (including deleted files) to the staging area
* git add -A
* # To unstage changes for a file
* git reset HEAD <file\_name>
* # To discard changes in the working directory for a file
* git checkout -- <file\_name>
* # To create a new branch and switch to it in one step
* git checkout -b <new\_branch\_name>
* # To rename a file in Git
* git mv <old\_file\_name> <new\_file\_name>
* # To delete a file from Git and the filesystem
* git rm <file\_name>
* # To show changes introduced by the last commit
* git show HEAD
* # To fetch changes from a remote repository without merging
* git fetch origin
* # To merge changes from a different branch into the current branch
* git merge <other\_branch\_name>
* # To create a tag for a specific commit
* git tag <tag\_name> <commit\_id>
* # To list all tags in the repository
* git tag
* # To switch to a specific commit
* git checkout <commit\_id>
* # To create a new branch at a specific commit
* git branch <new\_branch\_name> <commit\_id>
* ```