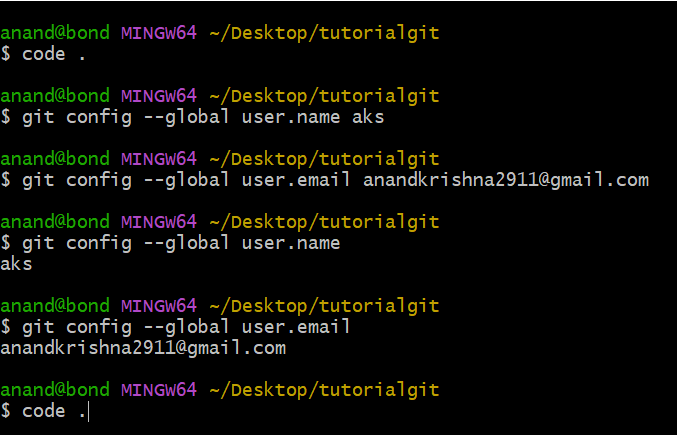
**GIT commands that are important to learn to use command line.**

**Make a folder go inside it and right click open git bash**

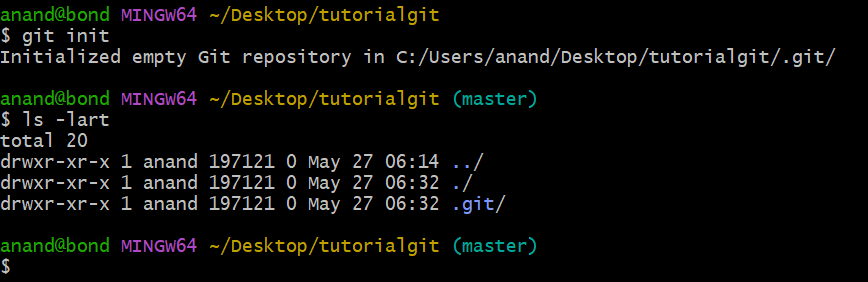
**git –version**

**git init 🡪to initialize the folder**

1. **git config --global user.name** **aks** – For configuring the name
2. **git config --global user.email** [**anandkrishna2911@gmail.com**](mailto:anandkrishna2911@gmail.com) – to set the email
3. **git config --global user.name** – For checking the changes have been affected
4. **git config** **--global user.email** - For checking the changes have been affected



1. **code .** – to open the VS Code
2. **git init** – initialize the git repository
3. **ls -lart** – To see the Hidden files



1. **git status** – To check the status of the files
2. **touch about.html**- create a blank about.html file (edit in vscode html:5🡪edit title “About Page”🡪edit body “This is About Page”)
3. **touch news.html** -create a blank news.html file (edit in vscode html:5🡪edit title “News Page”🡪edit body “This is News Page”)
4. **git status**
5. **git add about.html** – To add to the staging area
6. **git status**
7. **git add –A –>** To add all files in staging area
8. **git status**
9. **git commit –m “The First Commit” –** To commit along with message
10. **\*\*GoTo VS code and delete the contents of about.html and news.html and save**
11. **clear** – to clear the previous files
12. **git checkout about.html** – To match the file with last commit
13. **git checkout news.html** – To match the file with last commit
14. **\*\*GoTo VS code and delete the contents of about.html and news.html and save**
15. **git checkout -f** – All files get matched to the previous files
16. **git log** – to check the activity of the files, all the commits are shown and config files
17. **git log -p -1**– to see he last 1 commits on the machine and shows the changes we made
18. **press q to quit**
19. **touch contact.html**- create a blank contact.html file (edit in vscode html:5🡪edit title “Contact Form🡪edit body “This is contact Page”)
20. **git status**
21. **git add .**
22. **git status**
23. **GoTo VS code and edit and append the contents of news.html in body section <h1>This is New Feature</h1>**

**save**

1. **git status**
2. **git add –A**
3. **GoTo VS code and edit and append the contents of about.html in body section <h1>This is New Feature</h1>**
4. **GoTo VS code and edit and append the contents of contact.html in body section <h1>This is New Feature</h1>**

**save**

1. **git status**
2. **git add –A**
3. **git commit -m “Adding the new features to files”**
4. **GoTo VS code and edit and append the contents of about.html in body section <h1>This is New Feature</h1>**

**save**

1. **GoTo VS code and edit and append the contents of contact.html in body section <h1>This is New Feature</h1>**

**save**

1. **clear**
2. **git status**
3. **git commit -a -m “skipped staging area”** -If you wat to directly commit the file without making it to the staging area.
4. **git log -p -2**– to see he last 2 commits on the machine and shows the changes we made
5. **ls**
6. **touch delete.html -**to create a blank delete.html file (edit in vscode html:5🡪edit title “Delete File🡪edit body “This is Waste Page”) save
7. **git status**
8. **git add -A**
9. **git status**
10. **git rm –cached delete.html** – it will remove from staging area only not from hard disk it will become untracked
11. **clear**
12. **git status**
13. **git add –A**
14. **git commit -m “This is waste file”**
15. **git status**
16. **git log –p -1**
17. **git rm –cached delete.html**
18. **git commit -m “Deleted the waste file delete.html”**
19. **git rm –f delete.html** -If you want to delete a file from hard disk also
20. **git commit -a -m “Removed Delete.html”**
21. **git log -p -2** – If you want to see 2 commits
22. **git status -s** – shows the summarized status (modify the file then use this command )
23. **GoTo VS code and edit and append the contents of contact.html in body section <h1>This is New Feature</h1>**

**save**

1. **git status –s 🡪** **s** is for summary
2. **git add –A**
3. **git status -s** 🡪 shows the m in green
4. **touch .gitignore** -Files that you don’t want track
5. **touch mylog.log** -Example of log file write anything (process download upload etc)
6. **in .gitignore write mylogs.log to ignore all the file with this name(\*.c/\*.java) /foldername**
7. **git status -**it will not show that file added in the gitignore
8. **git add –A**
9. **Create folder mylogs and in that create a file mylog.log**
10. **git status**
11. **\*\*create a blank hello.html file (edit in vscode html:5🡪edit title “Hello File”🡪edit body “This is Hello Page”) save**
12. **git status**
13. **git add –A**
14. **git status**
15. **git commit –m “adding git ignore files”**
16. **add this in gitignore /mylogs.log -**if you want to add the same file in same folder
17. **git rm –cached logs/mylogs.log** -to remove from the stages area
18. **\*.log in gitignore – To add file with .log (.cpp, .exe, .c, .java)**
19. **/mylogs –To ignore the folder**
20. **Git commit -a -m “Ignoring the files in the ignore files”**
21. **git diff** – it compares the working tree with staging area(if both are same then no output)
22. **git diff –stagged** -compare the last commit with staging area
23. **git checkout -f** – matches the file with last commit
24. **ls** -we can see all the files as it is feature of UNIX.

**Creating a Branch by command line**

1. **git status** -Explain about the on branch master 
2. **git branch -**it shows all the branch in the git
3. **git branch update1** -New branch with name update1 is created
4. **git branch** – now two branches will be shown
5. **git checkout update1** -Switch to the branch update1

**Using the merge feature in the git command line**

1. *Make some changes in the code you have written in any of the file then use next command* **(we modified the about.html appended some more line in the body)**
2. **git add -A**
3. **git commit -m “Fixed the programs and added more features”**
4. **git status**
5. **git checkout master** – To move to the master Branch
6. **git checkout update1**
7. **add the comment in the file any comment**
8. **git commit -a -m “added some comments”**
9. **git status**
10. **git checkout master** -for merging the branch(update1) to master then we will switch to master
11. **git log**
12. **git merge update1** -this merge the update code to master
13. **git log -p -2** -We can see the logs of the same commit in update to the master now.
14. **git checkout -b update2** -It will switch and make the branch in this command

**Create a java file and add some printing code in it then commit**

1. **git commit -a -m “Added the node file in the code”**
2. **git status**
3. **when we switch to master the file of node will be lost now**
4. **git checkout master**
5. **in about.html write anything**
6. **git commit -a -m “modified the about page”**
7. **git status**
8. **git checkout update2**
9. **files again comes**

**Now talk about GitHub service (Microsoft service)**

**On command line**

1. **git checkout master**
2. **remote repository**
3. **paste the repository origin URL in the GitHub**
4. **git remote**
5. **git remote -v (push and fetch urls)**
6. **git push origin master ( as the private repo then it will show there is no repo)**
7. **create new folder on desktop on clone**
8. **copy the clone url**
9. **git clone url foldernanme (clone to the folder name )**

**Git Fetch**

**mkdir project01**

**cd project01**

**git init**

***add a remote URL to the local repo:***

**git remote add <short remote name> <remote URL>**

**git remote add origin <URL>**

***Confirm the remote added successfully:***

**git remote -v**

***Fetch a Remote Repository***

**git fetch origin**

***List all the fetched remote branches with:***

**git branch -r**

**git tag**

***Fetch A Specific Branch***

**git fetch origin test**

***To checkout the fetched content to a new branch, run:***

**git checkout -b test\_branch origin/test**

***To see all the remotes, run:***

**git remote**

***Fetch all the contents from all remotes with:***

**git fetch --all**

***Synchronize The Local Repository***

***Fetch the remote repository with:***

**git fetch <remote name>**

***Compare the local branch to the remote by listing the commit differences:***

**git log --oneline <local branch>..<remote name>/<remote branch name>**

**git log --oneline test\_branch..origin/test**

***Checkout the local branch where you want the changes merged:***

**git checkout test\_branch**

***Sync the local branch with the git merge command:***

**git merge origin/main**

**Git Stash**