**VERSION CONTROL SYSTEM :(Note git does not track empty folder)**



git config --global user.name “<username>” => to set user name shown in commit log

git config --global user.email “<useremail>” => to set user email shown in commit log

\*The --global flag is not mandatory, if removed the changes will be limited to the current repository

git config --list => to get all the configuration data

1. Create a local Git repository
2. GUI

VCS(TopBar) -> Enable Version Control Integration -> Git

1. CLI

Open git bash -> Navigate to the directory (pwd , cd , ls , ls -a)

Command:

git init

1. Adding things to staging area

Use a .gitignore file to specify the files that you don’t want to keep track of

CLI Commands

git add . => to add all the files and folders in current directory

git add index.html error.html => to add specified files

To unstage files

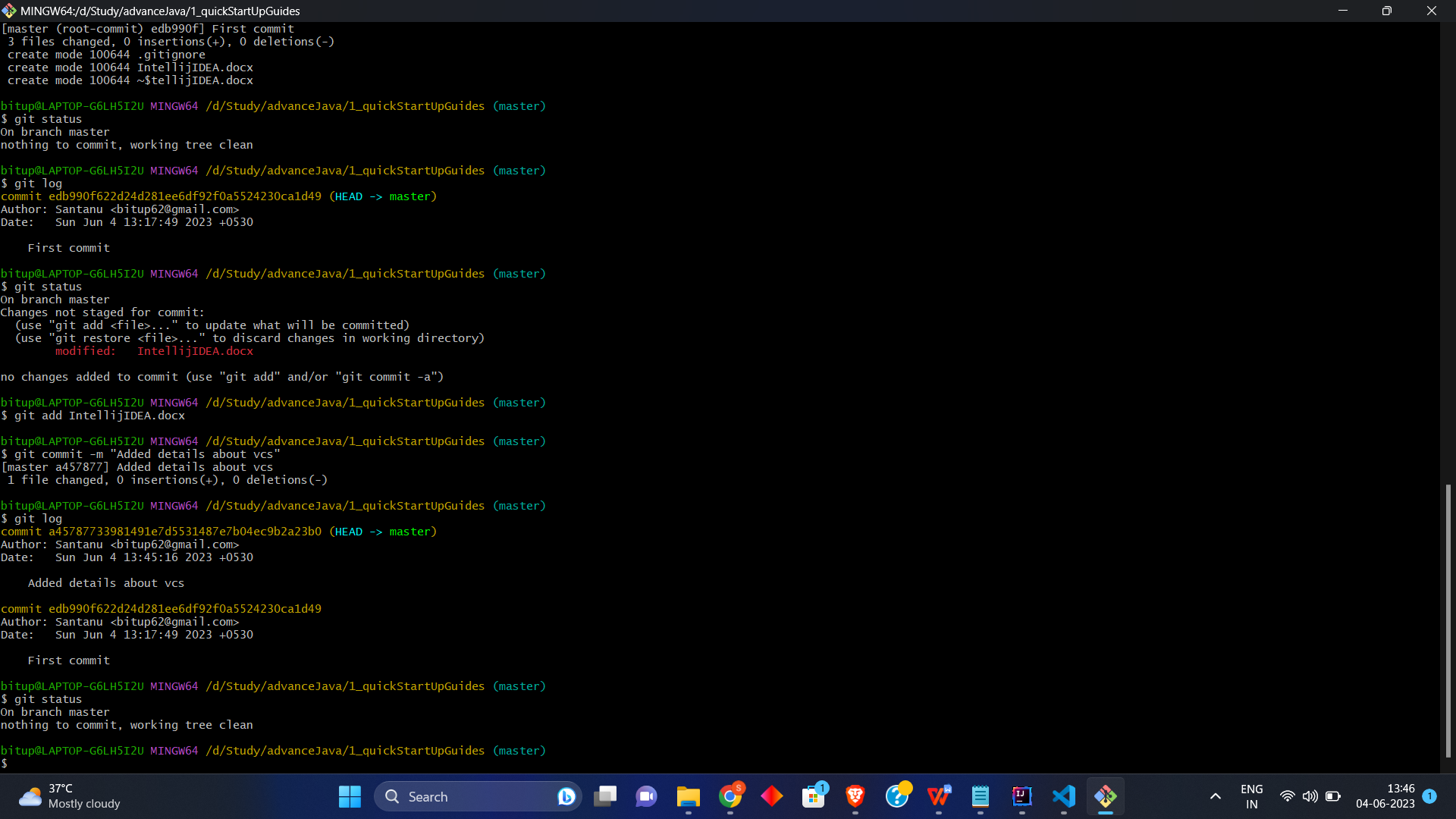
git reset HEAD <file1> <file2> =>to unstage the files but maintain the modifications made on them

1. Commiting the files put into git repository

git commit -m “<Commit message>”

1. Checking out Repo details

git status => To check out the files in the staging area



git log => To see the commit history

git log -p -1 => To see the last commit

git log --stat => To see commit history along with it’s short summary

git log --pretty=short

git log --pretty=full

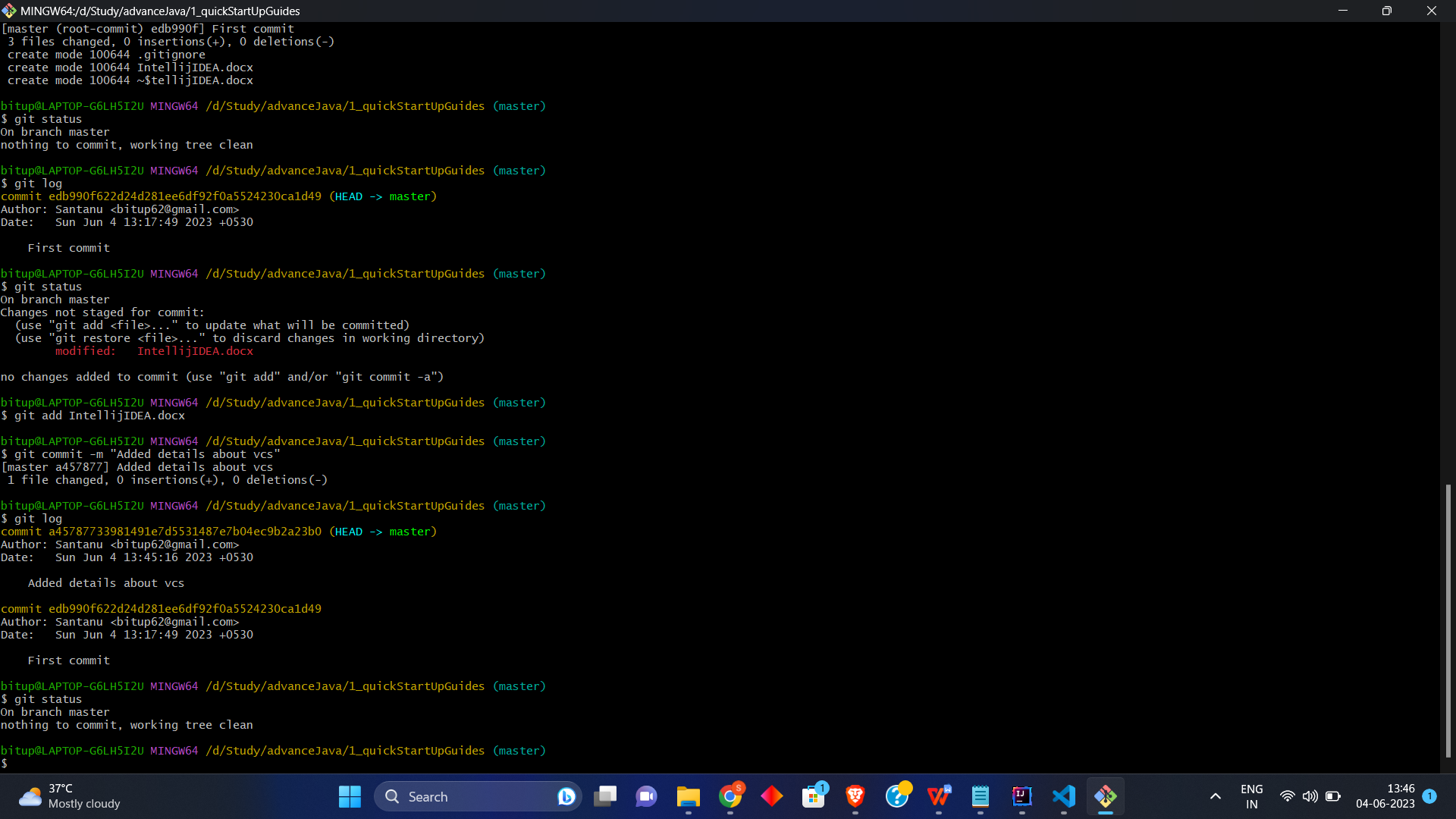
git log --since=2.months

git log --since=2.years

Website for tags: “https://git-scm.com/docs/pretty-formats”

git log --pretty=format:<str> => <str> = “%h -- %an”;

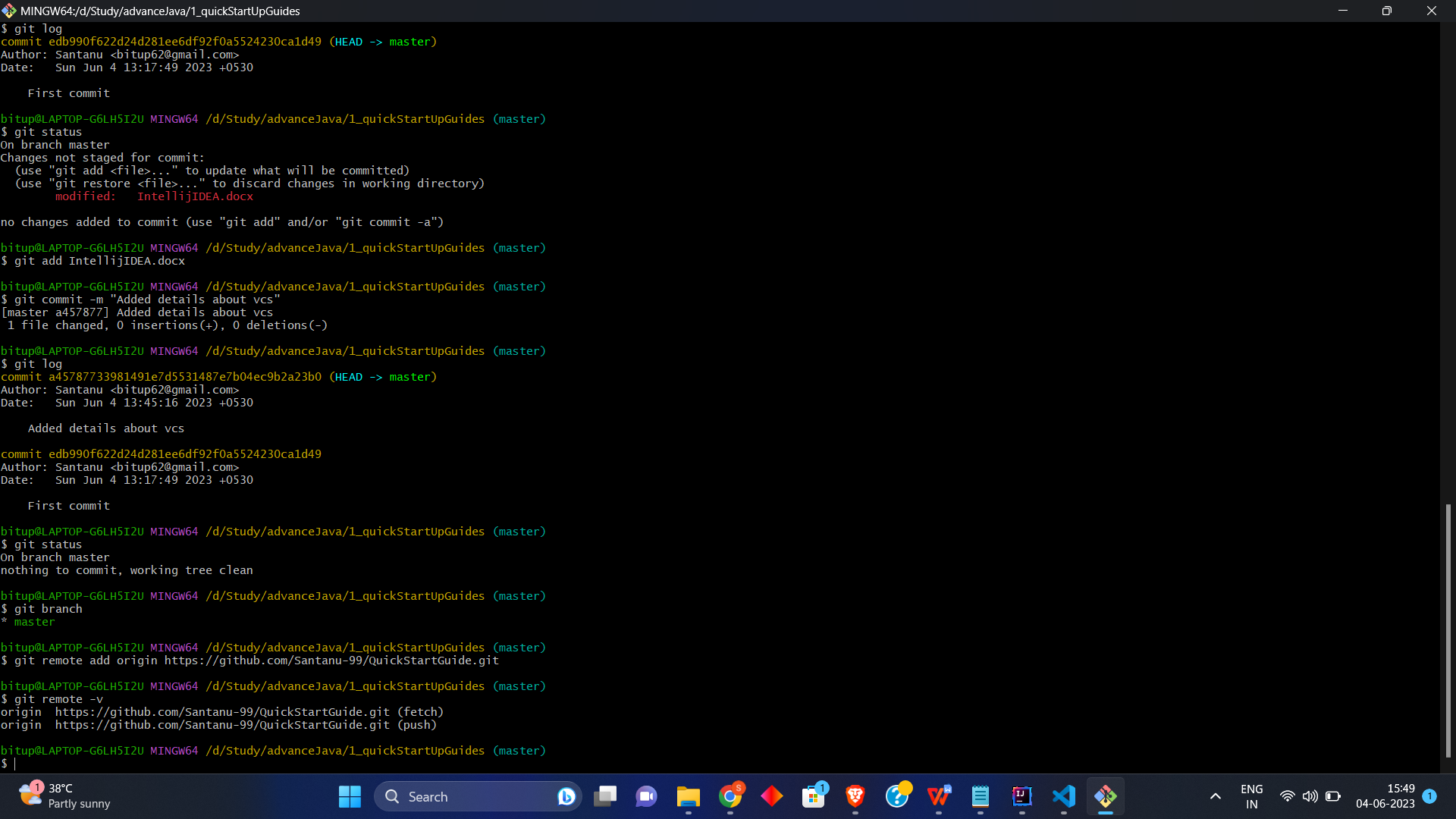
Ex:- git log --pretty=format:“%h -- %an”



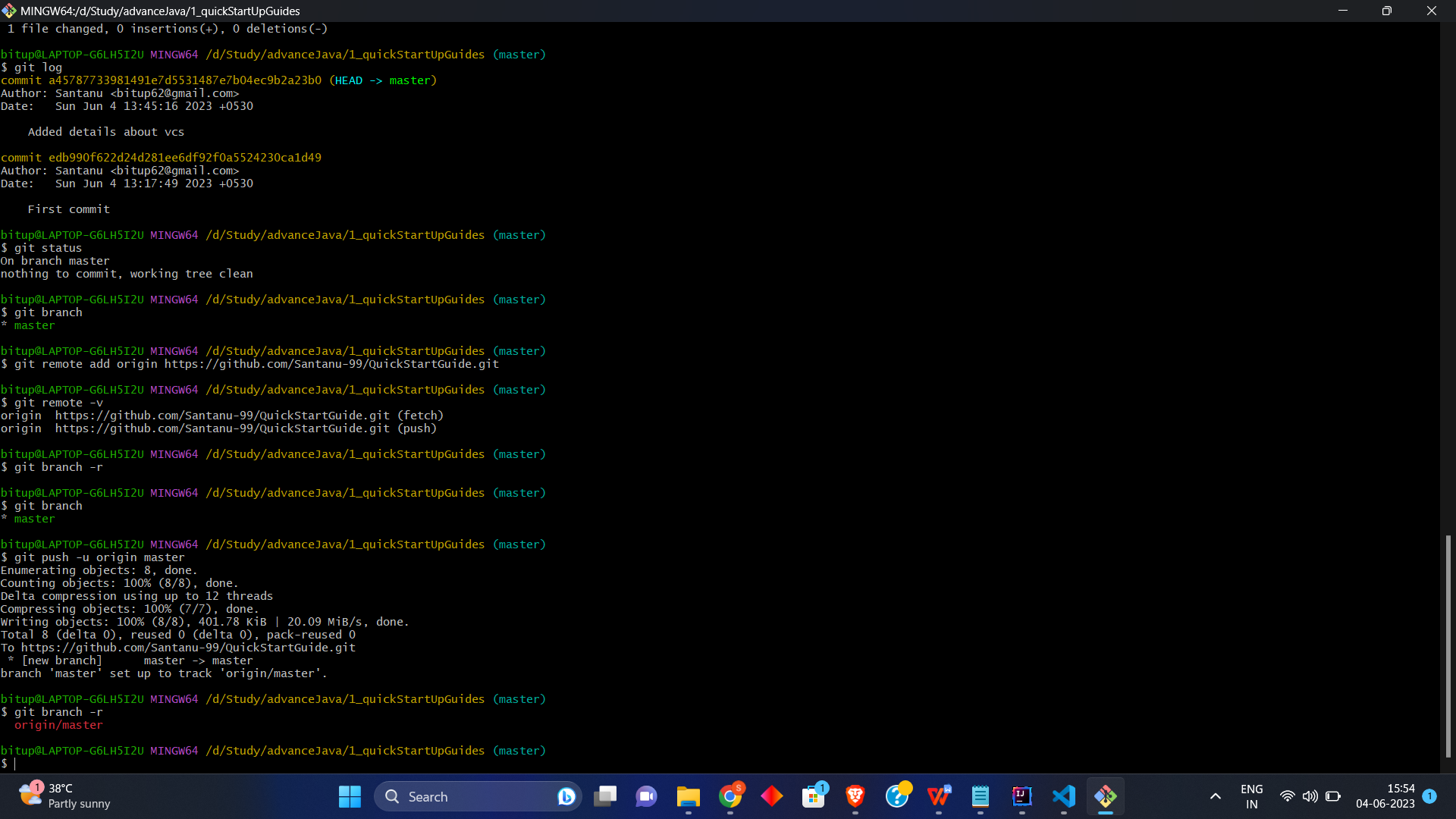
git remote -v =>To see all the remote repo

git remote add <remote repo alias> <remote repo URL> => To add a remote repo

git branch => To see all the branches in local repo

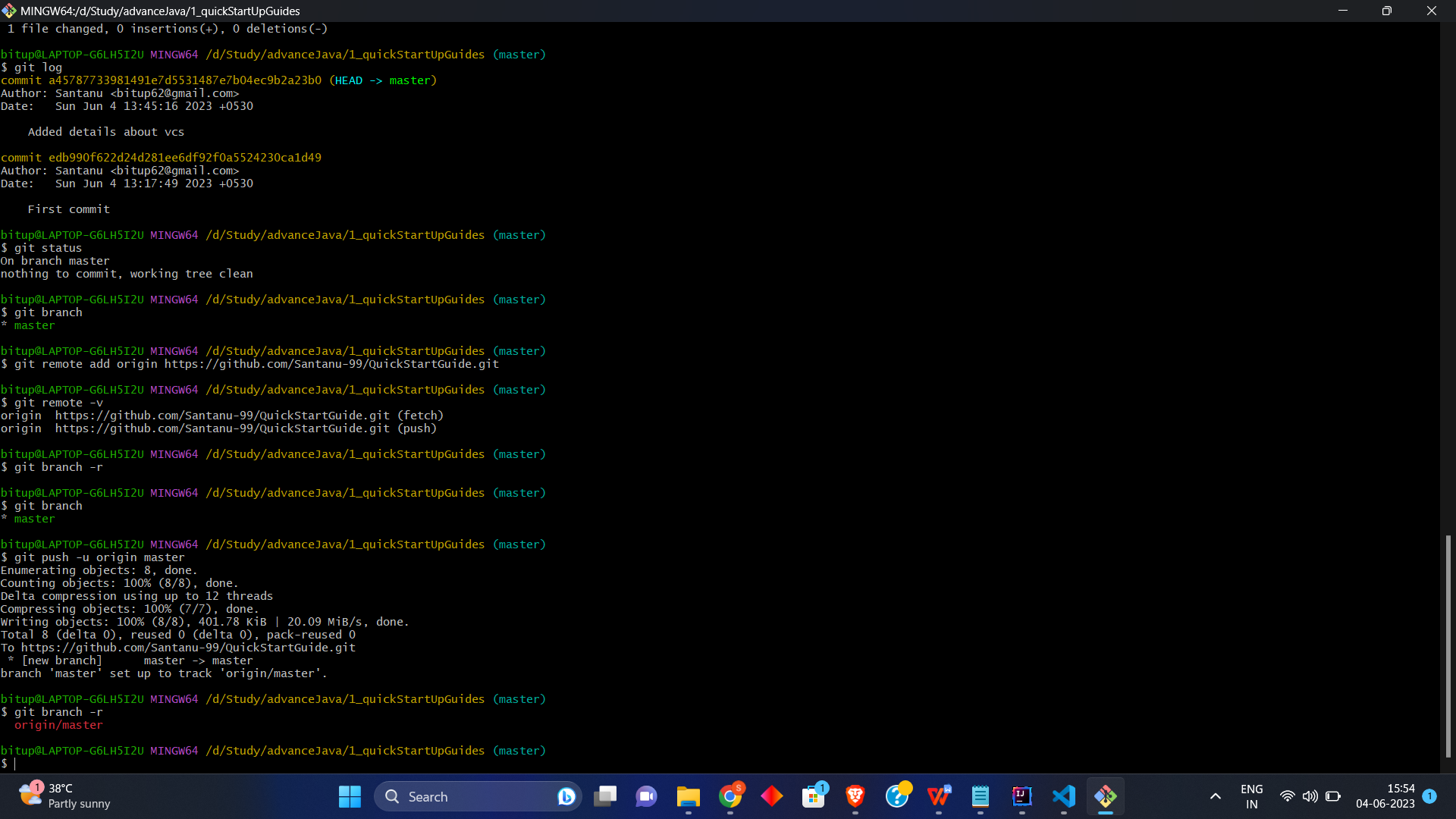


git branch -r => To sell all the branches in remote repo



1. Pushing commit to remote repo

git push -u <remote repo alias> <remote repo branch>



1. To delete a local git repository

rm -rf .git => removes the .git folder from the directory (essentially removing the repository , but other content of the folder does not get affected)

1. To Clone a project from github

GUI

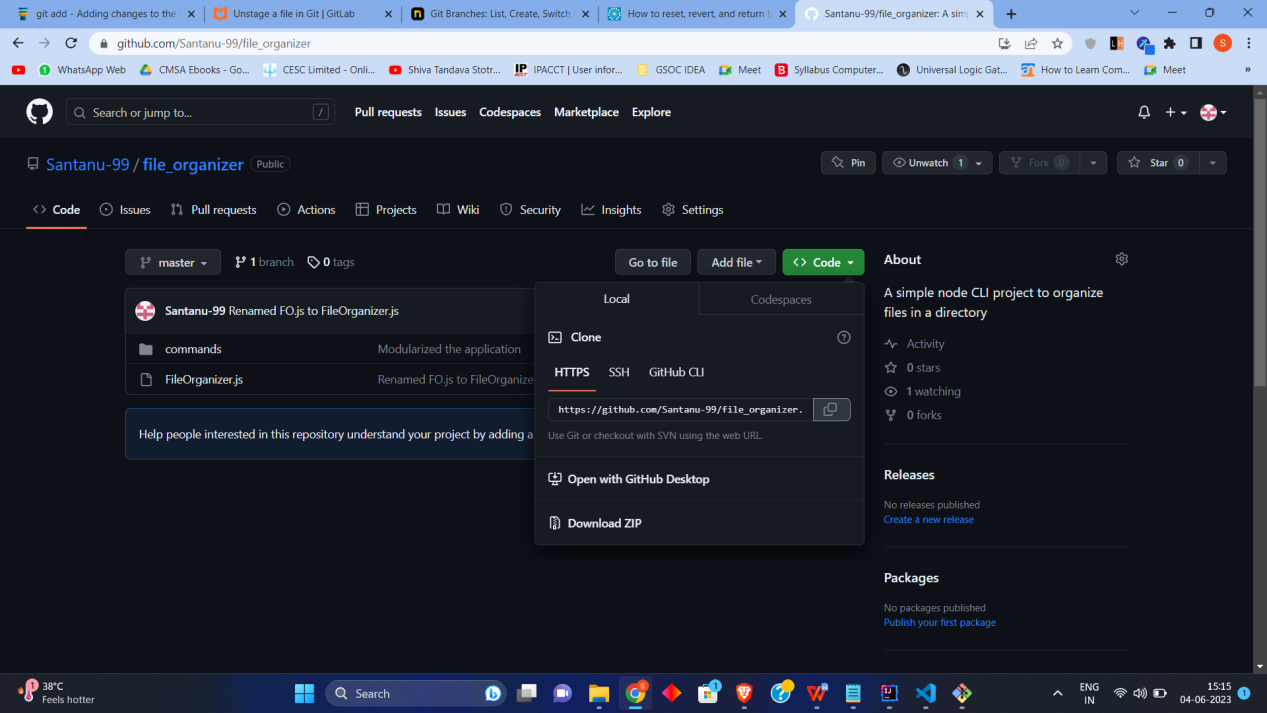
Open with VCS -> select git

-> set the directory where you want to save the repo clone AND set the remote repo URL

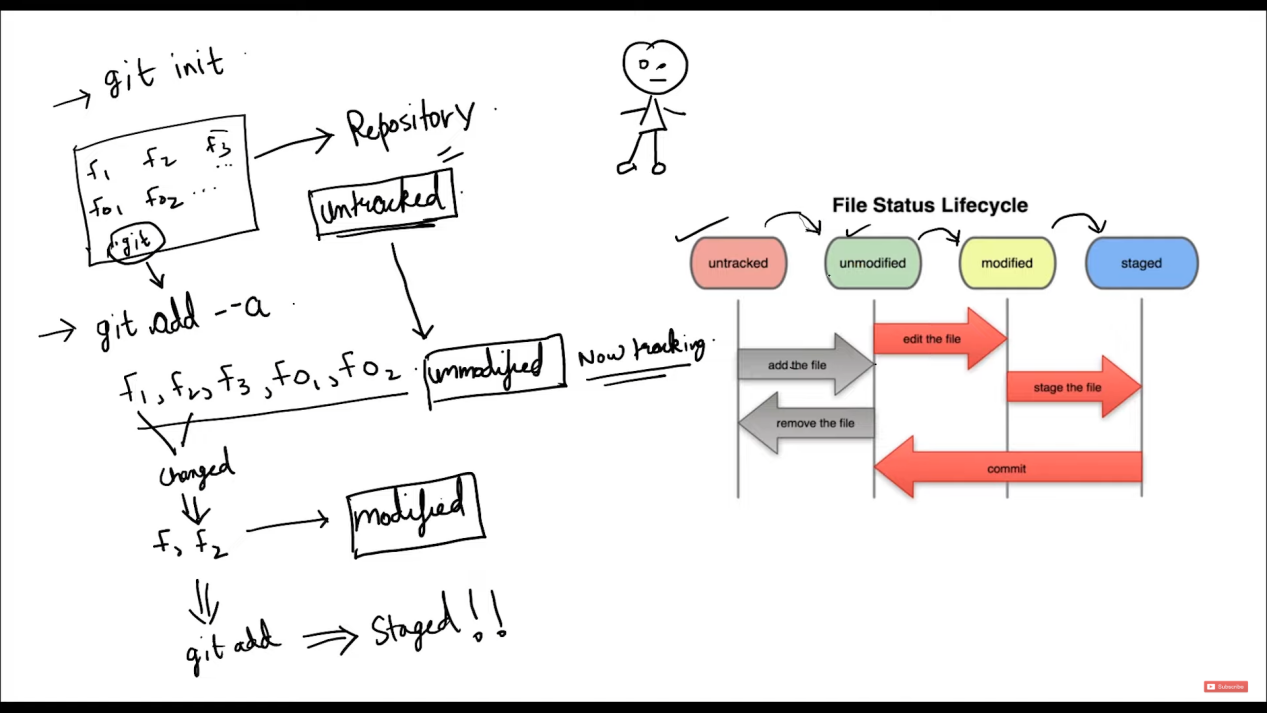
OR

git clone <Repo URL> <Clone Directory>

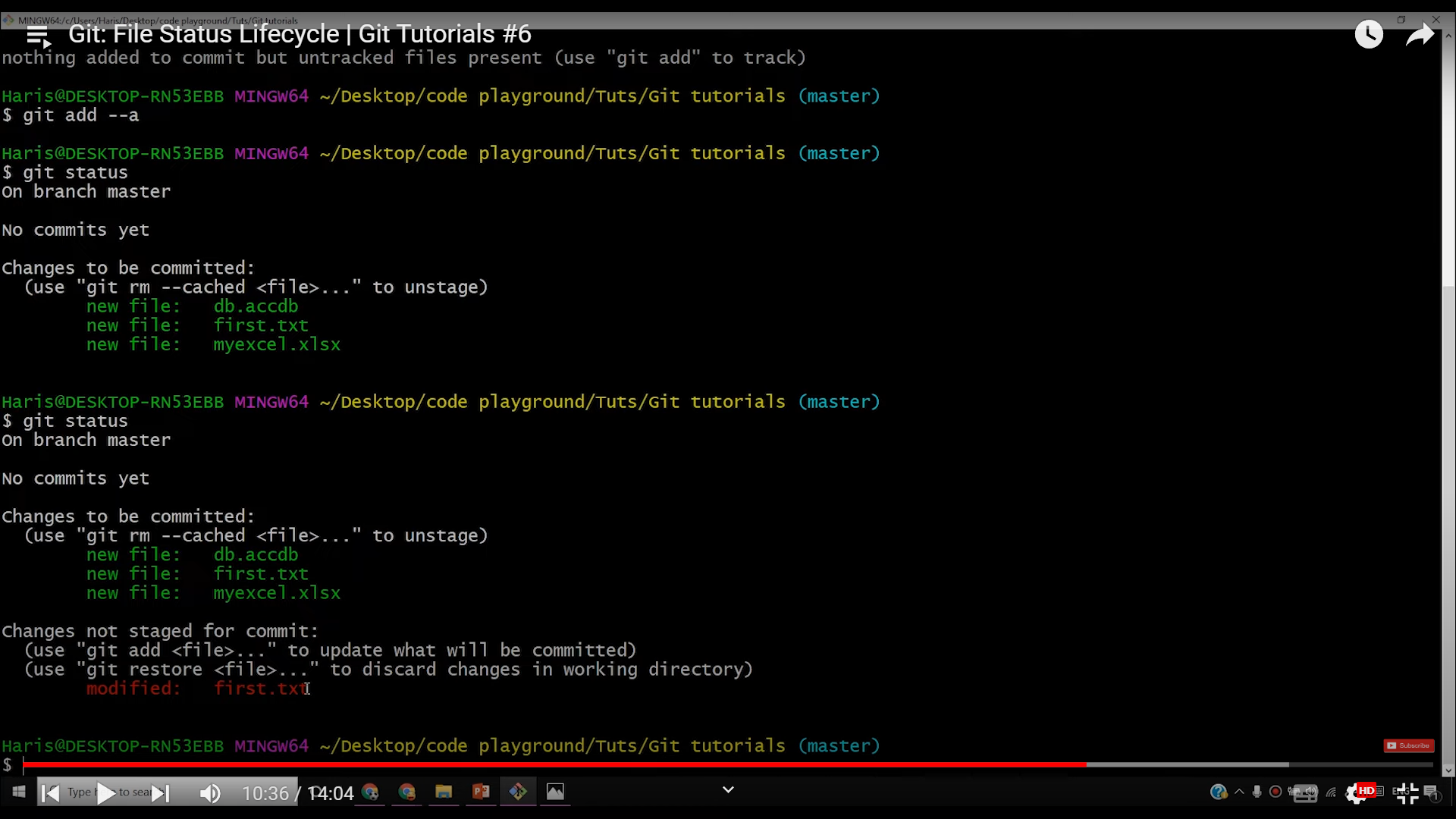
Repository URL From GitHub



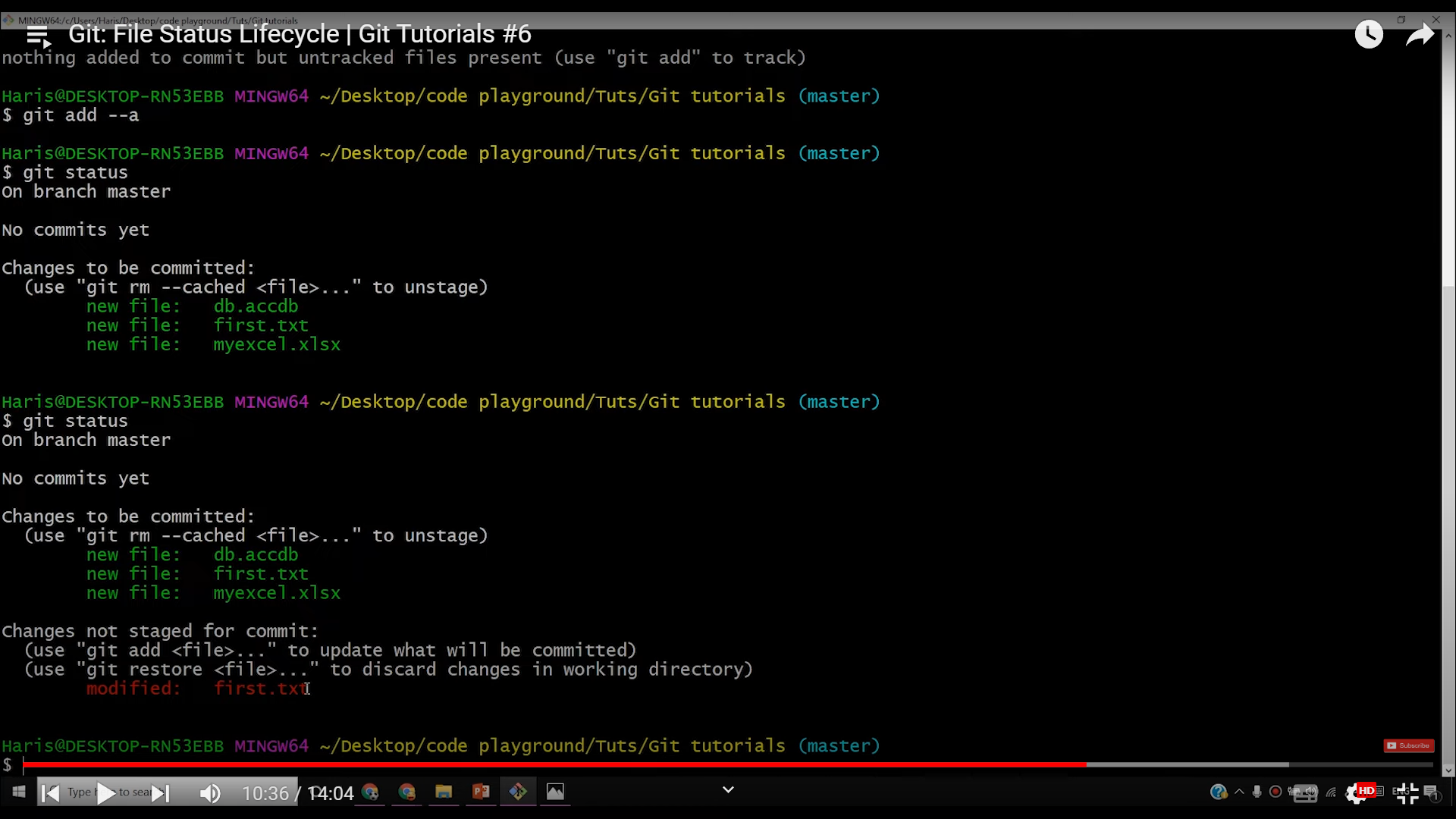
1. File Status Life Cycle



**After adding the files into staging area**



**After performing some changes to already staged file: first.txt**



In order to commit the updated first.txt , in next commit command execution

we will have to add the first.txt into staging area by

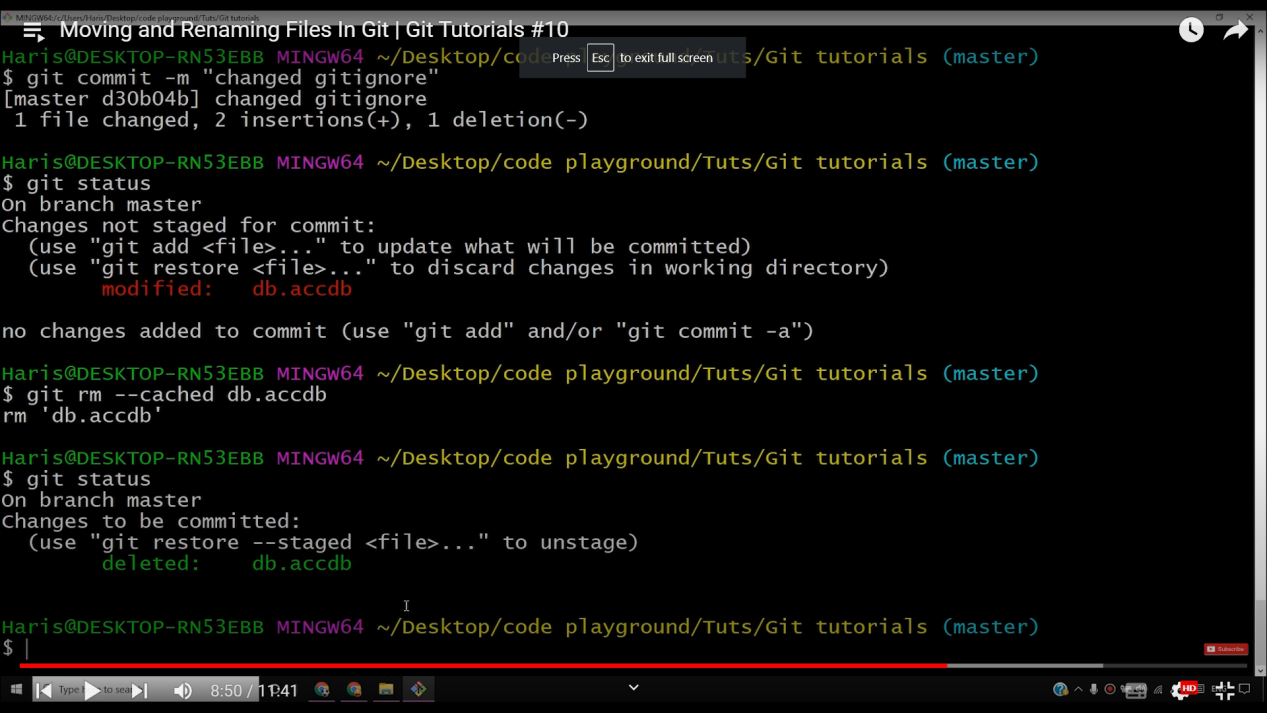
git add first.txt

Then,

git commit -m “<commit message>”

Else if we directly execute commit command without staging the modification, the modifications won’t get committed.

**To move a file from Tracked to UnTracked**



Steps:

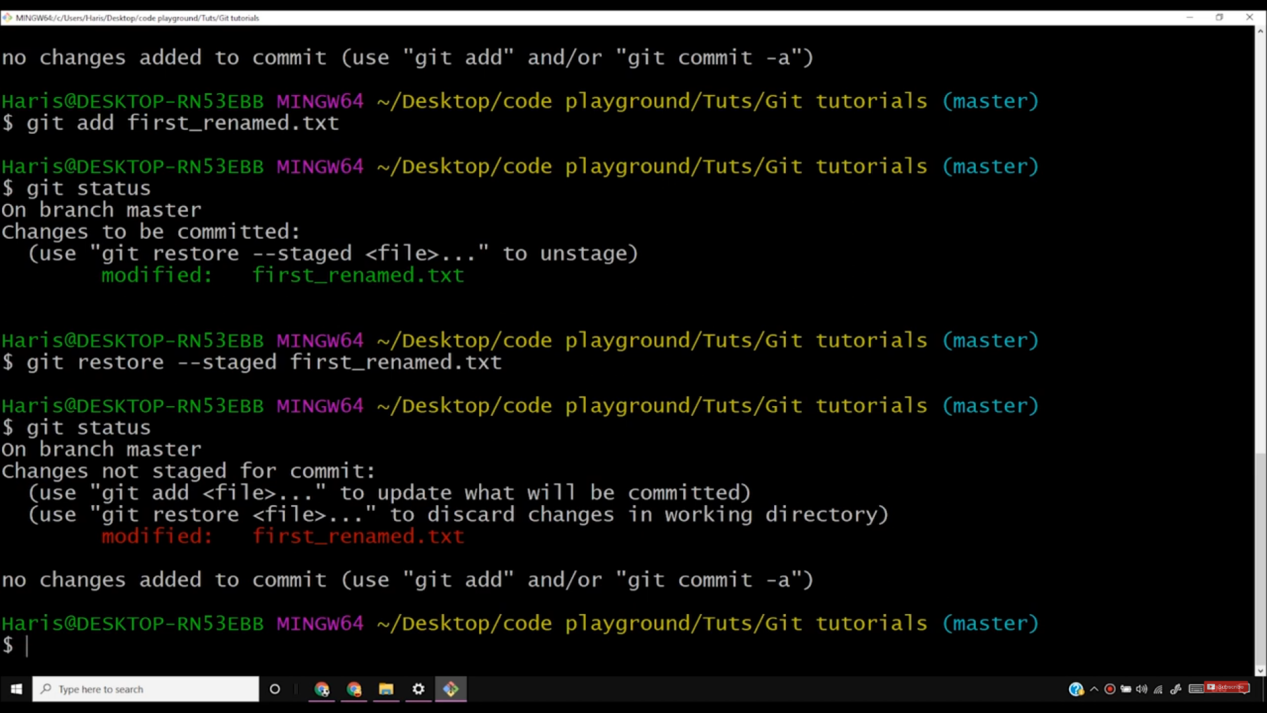
a)git rm --cached <fileName> => To make the file untracked(note this will not delete the file)

1. Add the <fileName> record to .gitignore file => To make the file not appear in untracked files

**To remove a file from git repository**

git rm <fileName>

**To Unstage a file**



Command:

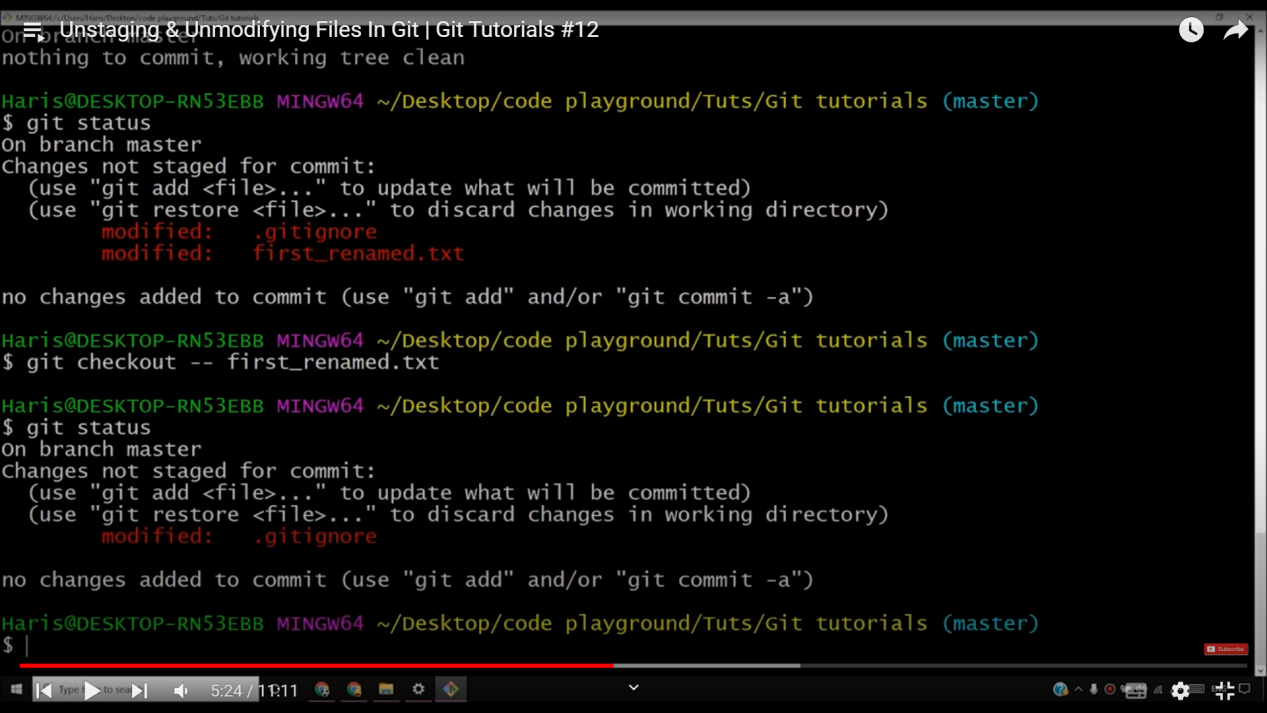
git restore --staged <filename>

**To restore a modified file to it’s last commit**

Command:

git checkout -- <filename>

\*This will only work if the file is in unstaged area



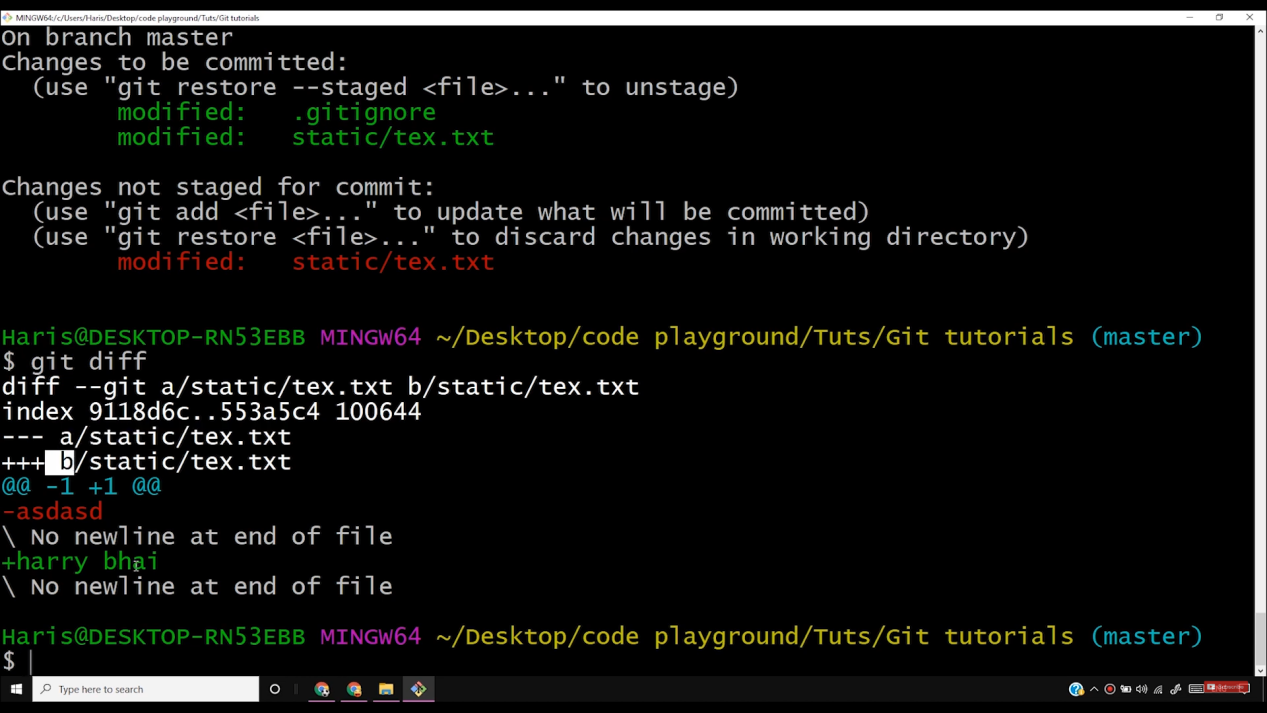
**To restore the whole working directory to last commit**

git checkout -f

\*This will only work if the file is in unstaged area

git pull -p => to sync local repo with remote repo

1. Comparing the staged and modified(if exists) version of the same file



Here tex.txt is compared

Command used:

git diff

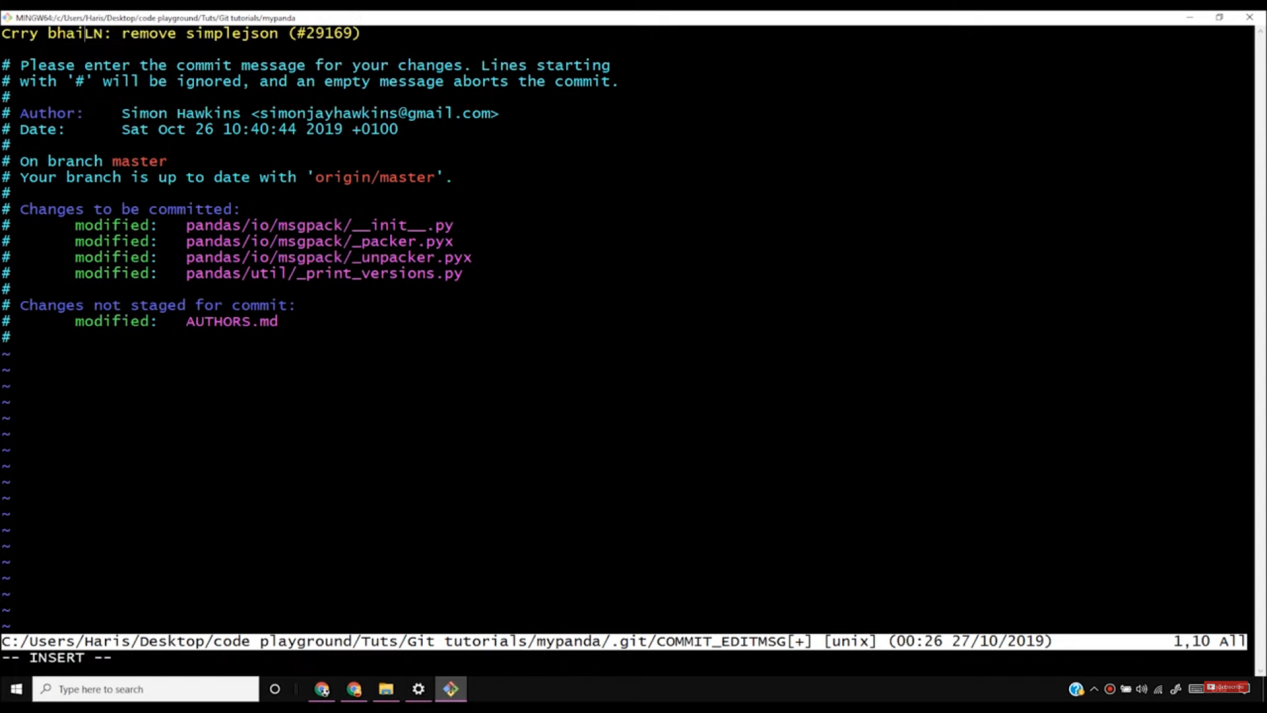
1. Comparing last commit with current staging area

git diff --staged

1. Merging commit with last commit after staging the modifications

git commit --amend

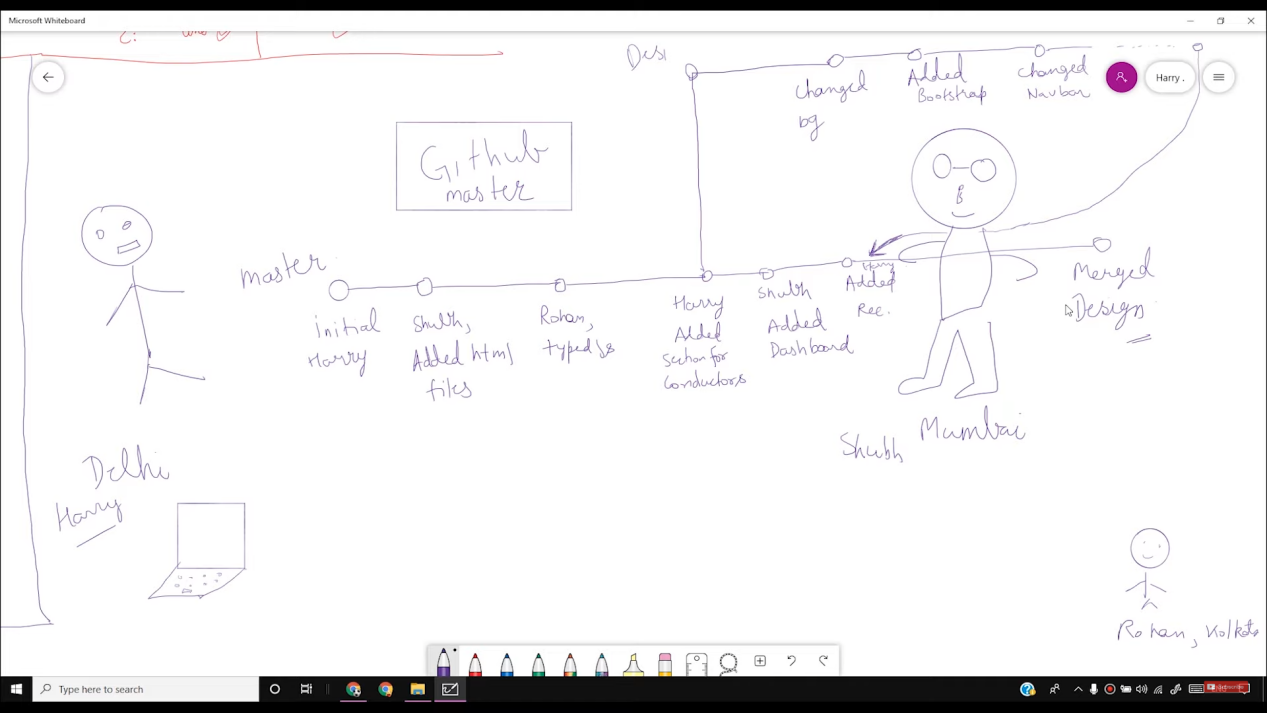
Vim editor will open(Edit it to change the message and authors……)



Press “I” to make things editable

Press “Esc” + Type in: “:wq” to save and EXIT

1. Branching and merging branches

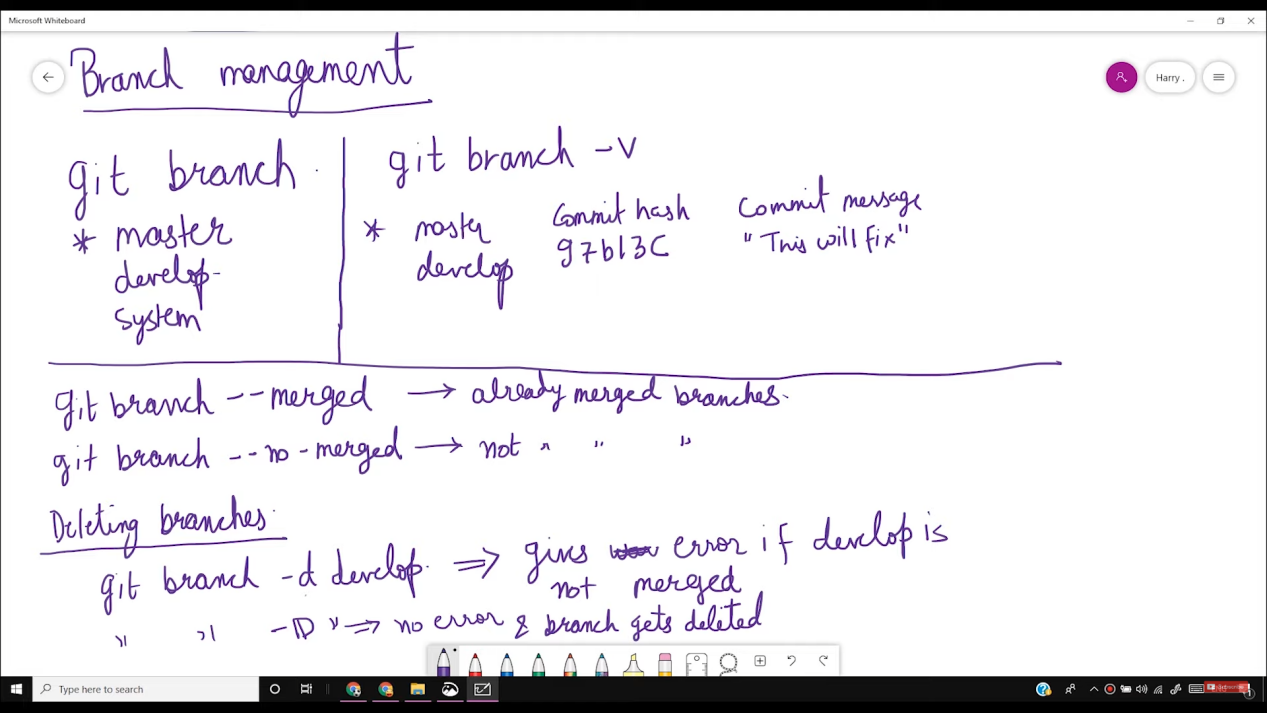


**To Create and switch branch**

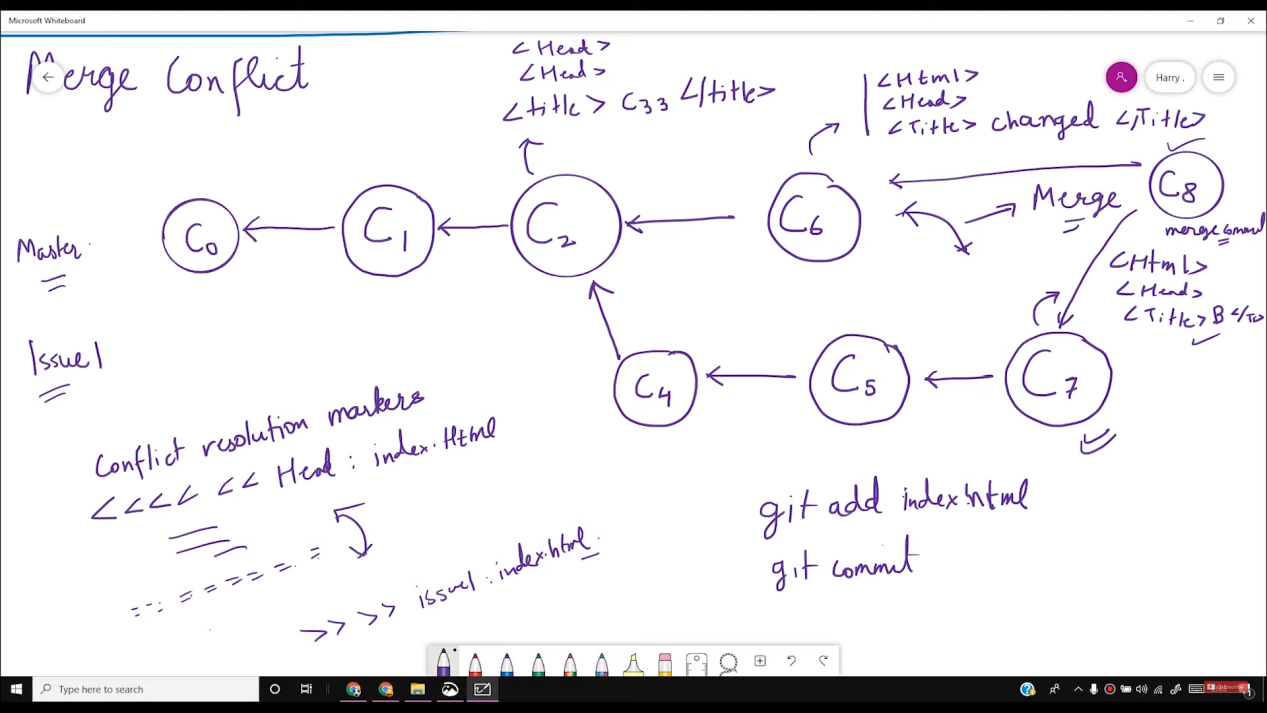
git checkout -b <branchname> => to create and switch to the new branch

git checkout <branchname> => to move to the specified branch

\*Always checkout or switch branch after committing (i.e. keep the directory clean)



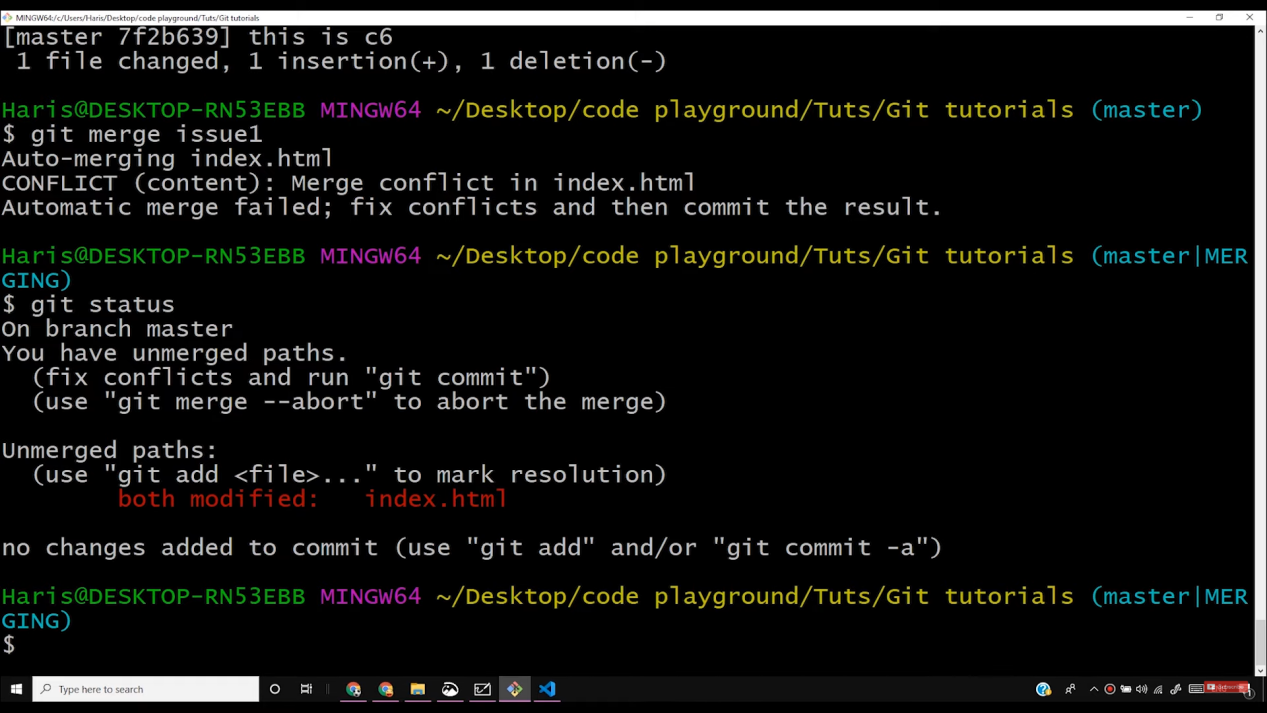
**For Merging Steps along with resolving conflicts**



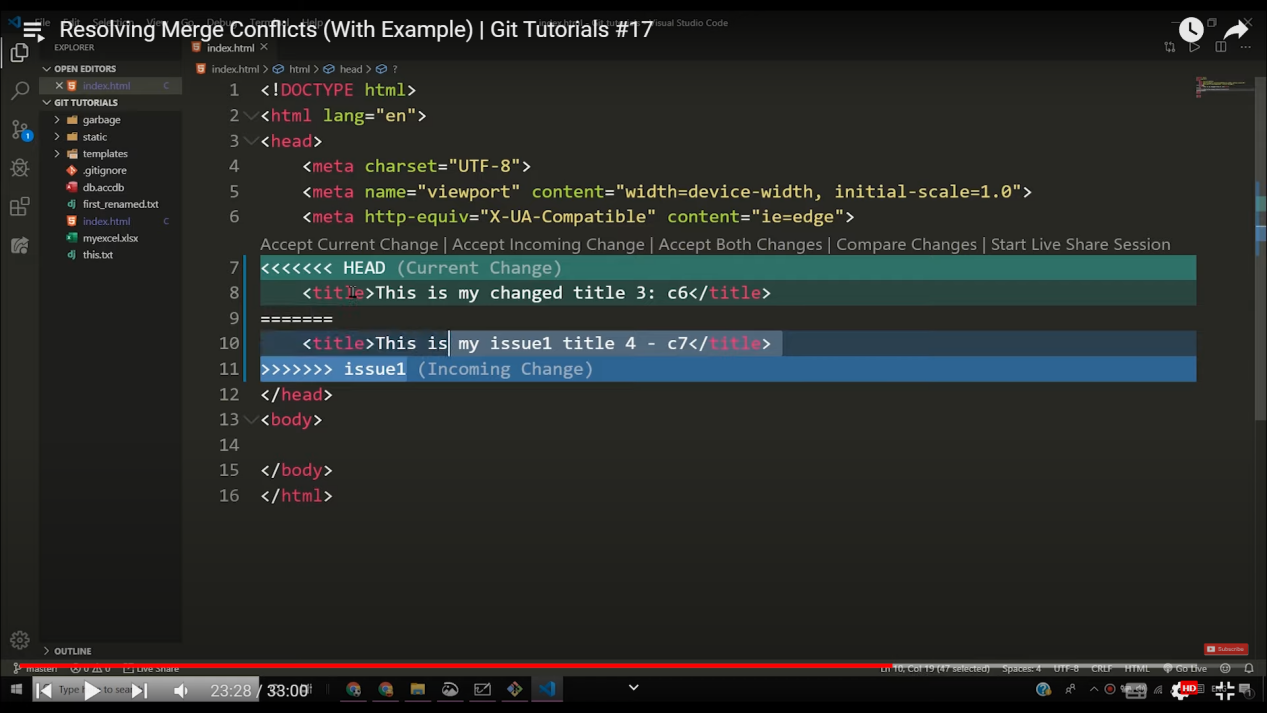
Steps:

1. git checkout master => move to master branch
2. git merge <branchname> => to merge the specified branch to master
3. Resolve conflicts (using the IDE) => whose change will be consider(the master branch or the merging branch)

\*\*To resolve conflict, simply delete the modification that you don’t want, using an IDE in the conflicting file



Inside IDE:



\*\* Changes done in master branch are marked by “<<<<<<<<<<”

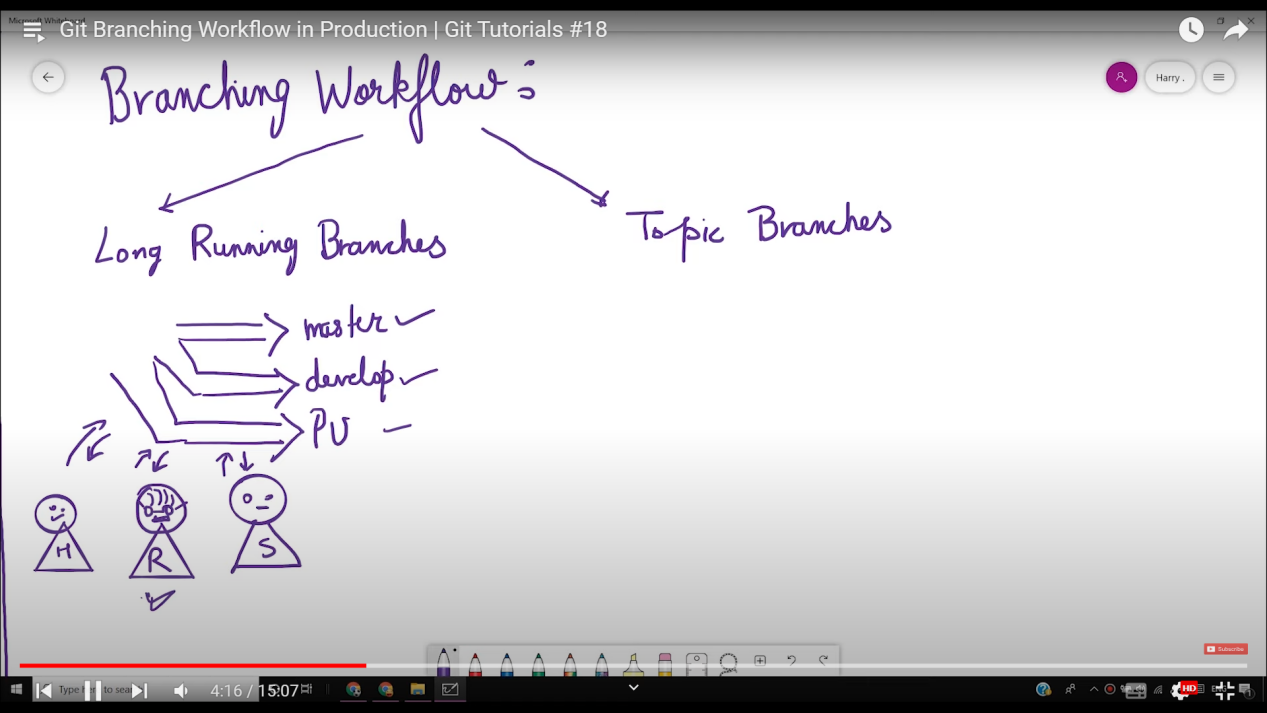
\*\* Changes done in merging branch are marked by “>>>>>>>>>>”

\*\* And Both are separated by “===============”

1. git add . => Puts all the merged changes in master branch into Staging area
2. git commit -m “<message>” => Committing the changes
3. Branching work flow

Long running Branches => will exist for the whole project’s lifetime

Topic Brancher => will get created to solve an issue and then will be removed



Inorder to push a branch to github(we need to do that explicitly), Steps:

1. First move into the branch using checkout command
2. git push <remote repo alias> <branchname to be pushed>

git push -d origin/<branchname> => To delete a branch in remote repository

Question:

Remote master got changed

I have worked on my local master

I have also created a branch in local

I want to merge the branch with the remote master but my local master is out of sync.

How to tackle this?

Soln.

First sync up local master with remote master without losing the changes in local master(How??)

Merge local branch with synced local master (can be done)