GIT

This is a version control system. You are maintaining your version in local repository and then moving code to cloud repository.

**Note 1-If you are already tracking the file, before doing any modification, check on which branch you are working.**

**Note 2- if you are working in master, files in working area will display content saved in masters version. if you are working in branch, files in branches will be appear in working area.**

**Note 3 – always work on other branches and then merge them in master**

# All commands

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Command** | **Remark** | **Working area** | **Unmodified** | **Modified** | **Staging** |
|  |  |  |  |  |  |
| **Git init** |  |  |  |  |  |
| **Git Status** |  |  | | | |
| **Git add <filename>** |  | **----------------------------------------------------------------🡪**  **🡨-----------------------------------------------------------------**  **🡨-----------------------------------------**    **--------------------🡪**  **--------------------🡪**    **🡨----------------------**  **----------------------🡪**  **----------------------------------------------------------------------**  **----------------------------------------------** | | | |
| **Git rm --cached** |  |
| **Git commit** |  |
| **Git rm <file Name>** | **Remove file from working area** |
| **if file got updated again** | **you need to add to staging than commit** |
| **Git add** |  |
| **Git checkout<file Name>** | **move last committed version workarea** |
| **git commit -a -m"added info in git notes"** |  |
| **Git diff** | **compare file between staging and workarea** |
| **Git diff --cached** | **compare file between staging and committed stage** |
| **Git log**  **Git log -p -2** |  |  |  |  |  |
| **Git branch <branch Name>** | **To create a branch** |  |  |  |  |
| **Git branch** | **Check all available branch** |  |  |  |  |
| **Git -b <branch name>** | **Create and switch** |  |  |  |  |
| **git checkout <branchName>** | **Switch to branch** |  |  |  |  |
| **git checkout master**  **git merge <branchName>** | **To merge branch to master** |  |  |  |  |
| **GITHUB** |  |  |  |  |  |
| **Git remote add origin <url>** | **To add SSH path** |  |  |  |  |
| **git push -u origin master** | **To push to origin path** |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Initialize the directory

Git init

# Check the Status

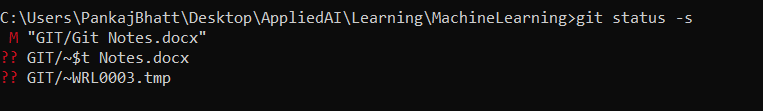
**Git status**

**git status -s**

to check the small status of files, the first box denote file in staging area, and second box denote working area

M – file is modified

$ - File is open in editor



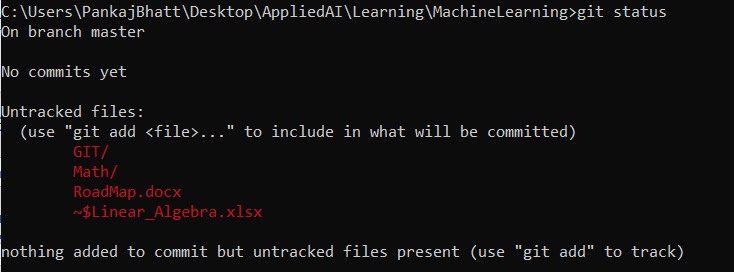
## 2.1 How to read Git status Output

**Branch** – master

**Commit** – No commit means there is no file tracked yet by GIT version control system

**Colour**-

Red – explains files are not yet tracked, that mean they are in working area



# Add file from work area to staging area

## 3.1 For individual file

Git add <filename>

## For group of files

## For all files

Git add -A

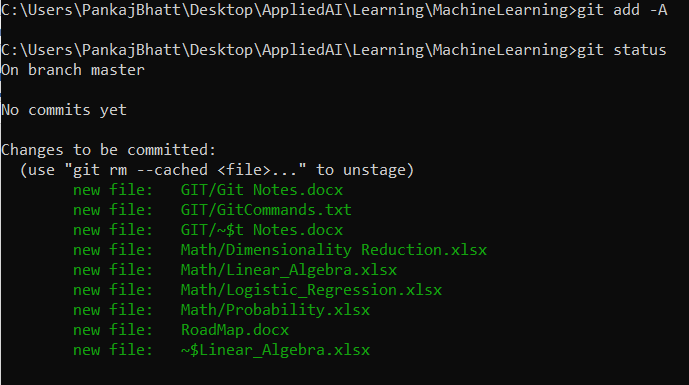
## 3.4 How to read GIT status output

**Branch** – master

**Commits-** No Commit  
This will tell which files are committed in GIT i.e. saved as a version

**File Colour-**

**Green –** tells files are in staging area



# Move file from Staging area to Git tracked area

## 4.1 Without adding a message

Git commit

## 4.2 Work with VM editor

press i to move to insert mode

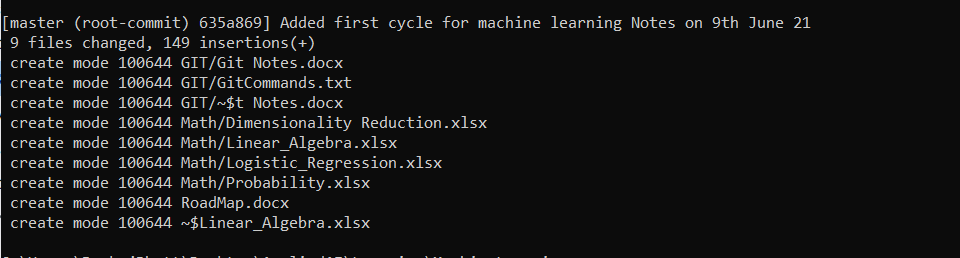
give your comments

esc to move to control mode

:wq to exist the VM editor

q is to exit

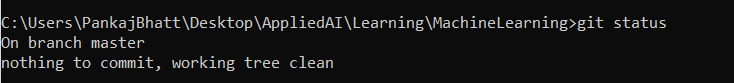
## 4.3 How to read output message

+ sign indicate, files have been added 

## 4.4 Commit with adding a message, avoid VM editor

Git commit -m “message”

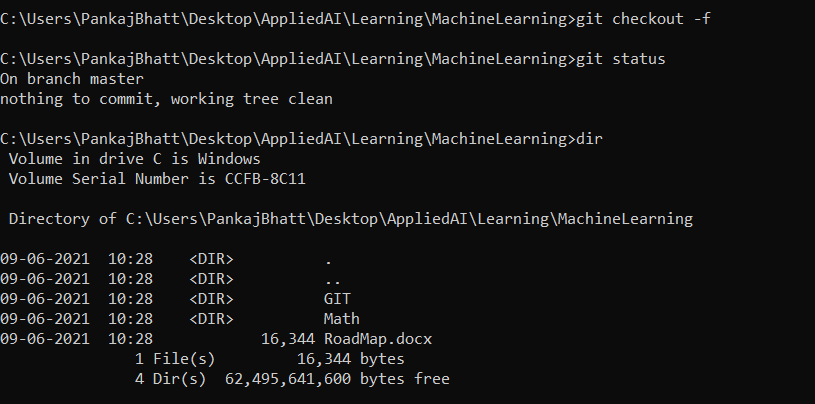
Git status - if all files are committed



# Move committed files to working area

If some unwanted change happened in working files.  
 if data is deleted, you can get last committed files back

* IF files are pushed in cloud
  + Pull it from cloud
* If files are not pushed in cloud
  + Git checkout -f



**git checkout <filename>**

if you done some mistake and save the file , to recover the previous version

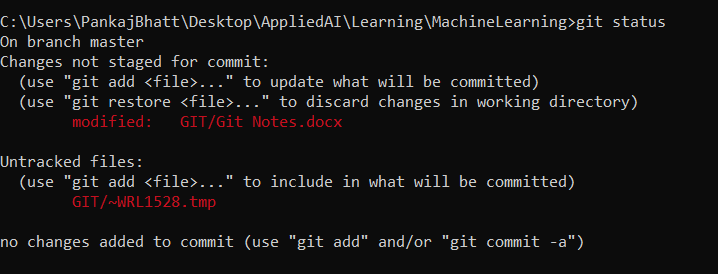
**git checkout -f**

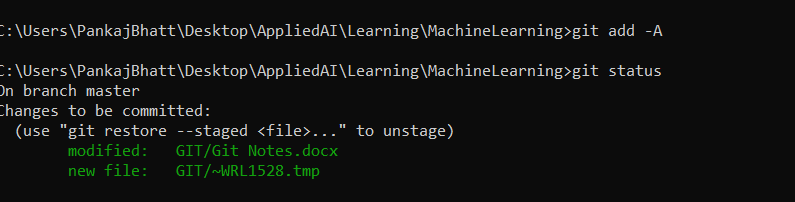
to recover previous version for all file

# Move Tracked Files, if gets modified

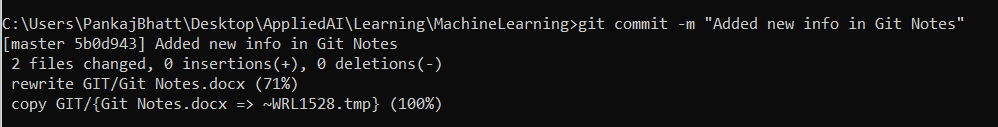
## 6.1 How to read Git status Command

Git status



Git add -A  


Git commit -m “Added new notes”



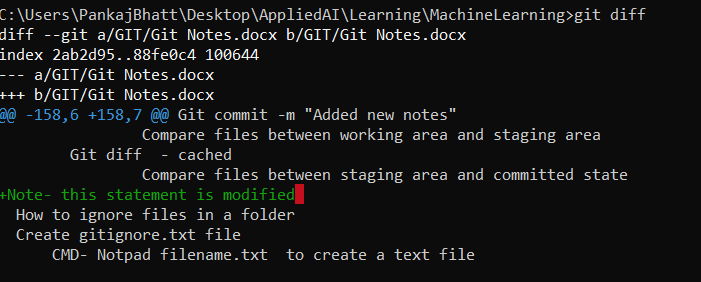
# How to know the difference among the files

**Git diff**

Compare files between working area and staging area

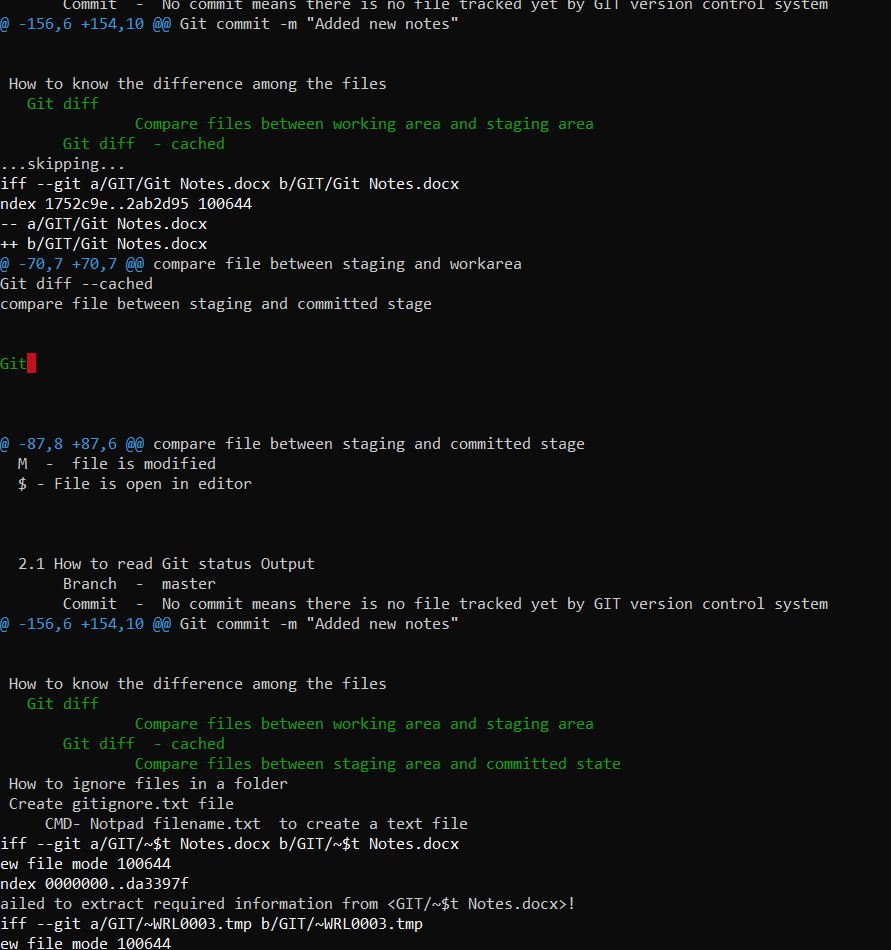
+ sign denotes the change

Note- this statement is modified



**Git diff –cached**

Compare files between staging area and committed state



# How to ignore files in a folder

## Create gitignore.txt file

**CMD**- Notpad filename.txt to create a text file

**Git bash** , there i can use linux command, to add a file we can use Touch

create a .gitignore file in directory using notpad .gitignore

or

if using git bash then touch .gitignore.

## 8.2 Add file and folder in gitignore file

git add -A to move all file to staging area

now add remove file which you want to ignore

\*.txt ----to ignore all file with some specific extension

/filename.txt ------to check some file in specfic path , it start from where gitignore file

folder/ --- to ignore a folder/

# How to view commit logs

git log to check all commit

git log -p -2 to check last 2 commit

# Branches

copy of same project , we can perform the development here. this will not effect the other branches

To add a new branch

**git branch <branchName>**

To check all available branch

**git branch**

the green color tell on which branch you are

Create the new branch and checkout to it **git checkout <branchName>**

**git -b <branchname >**

**Note - remember if you are in branch master, all file and code are, which you have committed in branch master**

**when you switch to other bracnch - file will get change and you will see changes you have done on that branch**

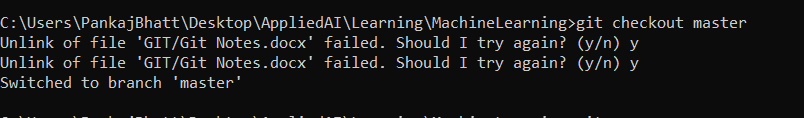
to switch to mentioned branch

**git checkout <branchname>**

to merge the branch to master first you need to switch to master, then use merge command

**git checkout master**

**git merge <branchName>**

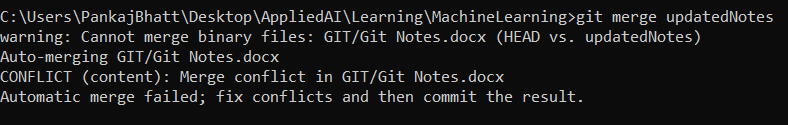
**if files are open and you want to switch the branch , you will get below error**

**case 1**– I have updated git notes file in branch updated Notes

and

I have also update the master branch file.

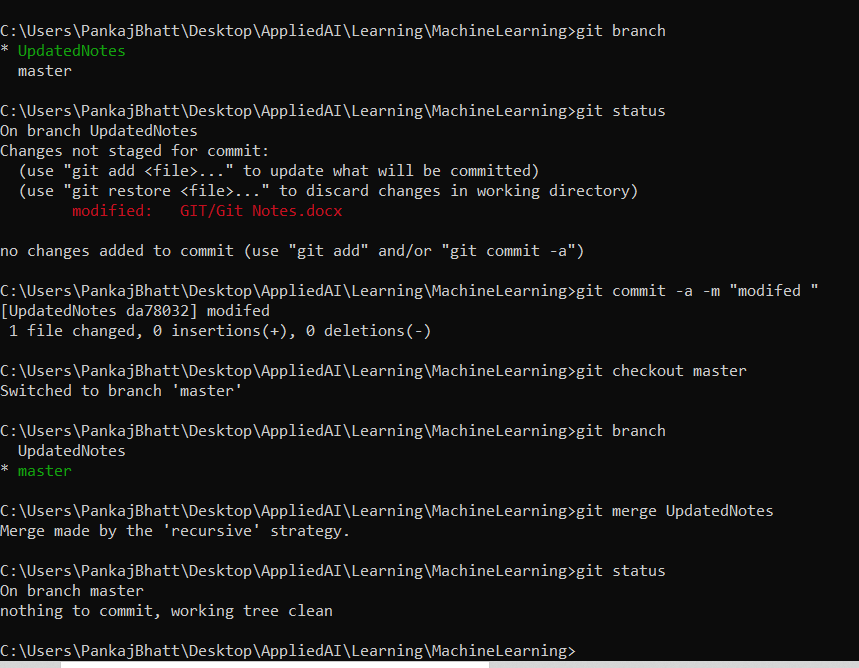
**Got below error**



**Case 2- Both had same files, no change,**

**Merge successful with recursive way**

**Case 3 – updated only Updated Notes Branch**



# Git hub

## Steps to push repository

1. create your repository in git hub
2. Copy SSH key <url>
3. **Git remote add origin <url>**

* This command will add the url to variable name origin
* git remote command will connect your local repository to your remote repository

1. **git remote -v**

* shows fetch and push URL

1. **git push -u origin master**

* push master to origin

1. **git pull origin master**

This will pull all the commits performed in git hub to your local repository

## What is significance of SSH key

1. Go to profile
2. Click on setting
3. Come to SSH and GPG key section
4. There we can add SSH key by click on new SSH key . This key give access to my local computer to push my code into server
5. Read notes, it will help.

SSH and GPG Key , -- lets says you have private repository , so to communicate with private repository , you should use these certificate key

# Clone a repository

1. Go to repository which you want to clone
2. Copy the SSH key
3. Git clone <url>
   1. This command will create a folder
4. Git clone <url> .
   1. This will create a repository

# Pull Request

Refer- https://www.youtube.com/watch?v=e3bjQX9jIBk

Let’s say a team has two player.

* Player A manage the git repository.
* Player B can clone the repository to local machine/ fork the repository to its own GITHUB
* make changes.
* add changes to player B own repository.
* Then send a pull request to player A(owner ) to merge the code.
* Player A will get a pull request notification.
* He can review the code.
* And if looks good , he can merge changes

# GitHub pages

