

What is Version control system (VSC)

It is a tool which allows all the developer to collaborate to the particular project and keep track on the modification and also can revert back to the original code if necessary.

Type of VCS

There are THREE types

- 1 local version control system
2. Centralized version control system
3. Distributed version control system

Local version control system

Trace the changes only in developer system, not in centralized VCS

Centralized version control system

Can store the file in central branch and make modification and restore it back by checkout process

Distributed version control system

Developer will have a complete copy of the repository with full history and can push/pull and make modification and revert it back to main branch

Example for VCS ---> GIT, SUBVERSION, MERCURIAL

Uses of VCS

1. Helps the developer to have a BRANCH so that main code will not get affected
2. Helps developers to COLLABORATE so that everyone can know the changes done
3. The main branch will have the detailed HISTORY so that pull the main Branch will get the complete History

Platform to perform the git

git hub , git lab , bit bucket , azure

Git Hub : it will host the Git repository

sudo yum install git

What is GIT -----LATEST VERSION --> 2.27.0.WINDOWS.1

GIT

Git is a free and open source distributed version control system designed to handle from small to large project with speed and efficiency

Repository : the data which store the file with their history

Initializing the git in local repository : when we initialize the git , we make sure that all the folder , files what ever created in it will be completely tracked

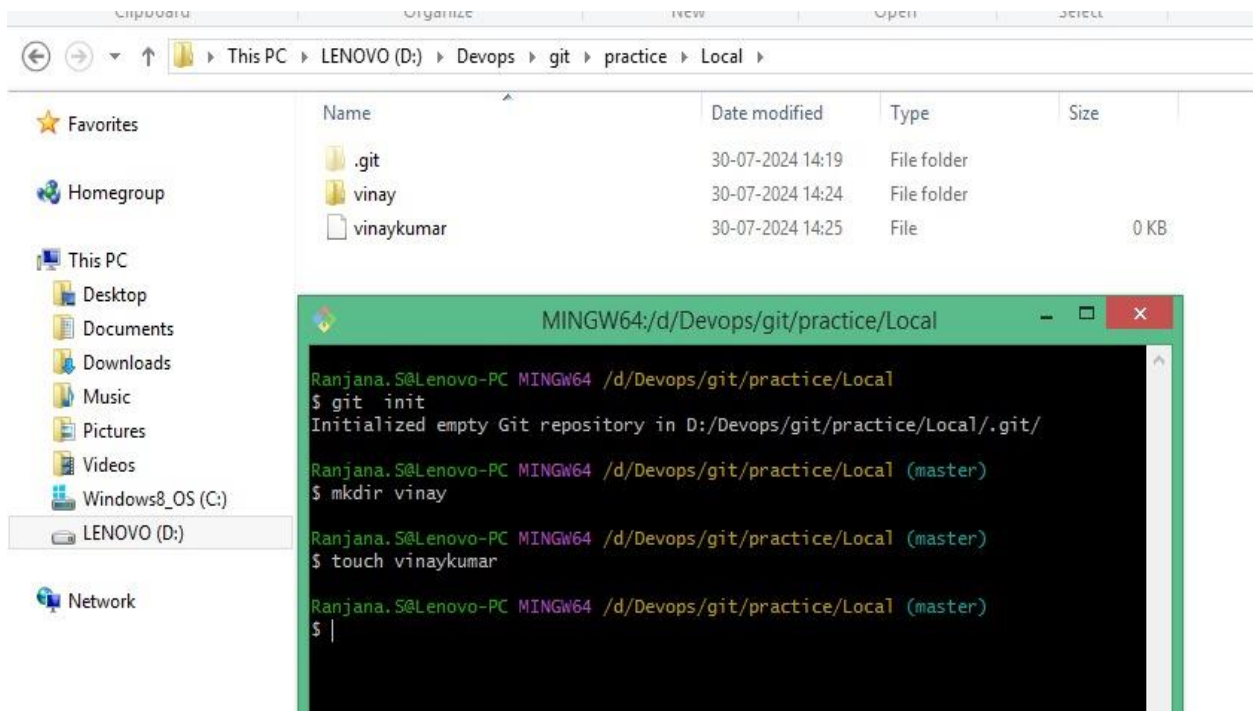
Syntax: git init

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice/Local
$ git init
Initialized empty Git repository in D:/Devops/git/practice/Local/.git/
```

Creating the new Repository (Folder) in local machine/Hub

syntax : mkdir <folder name>

syntax : touch < filename >



Git add : for moving the file which is working space to staging area we use git add.

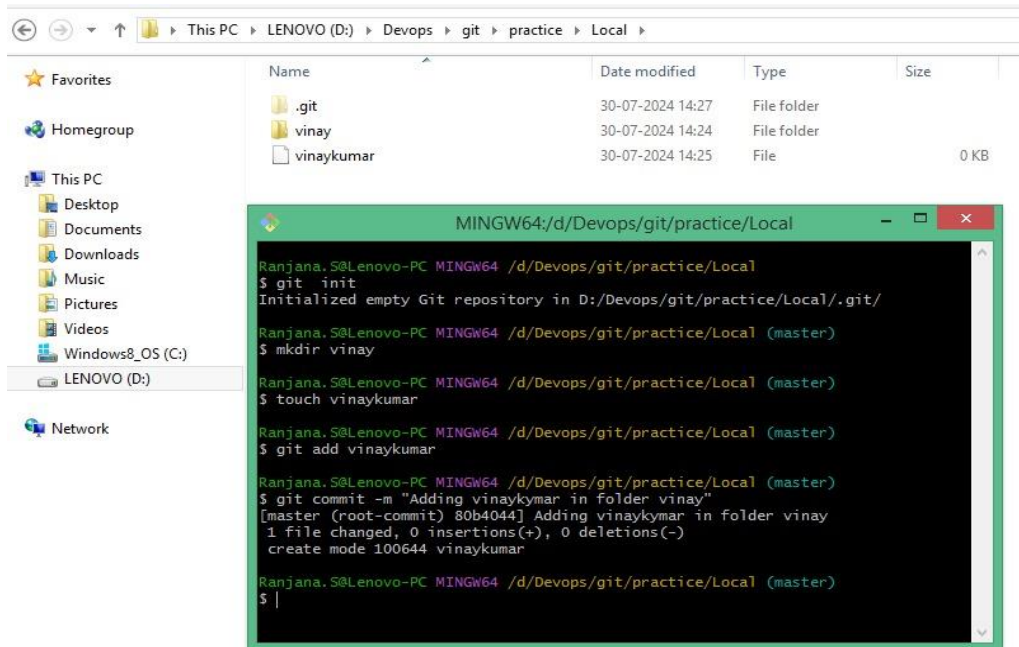
Syntax : git add < filename>

Git commit : use for moving the file from staging area to local repository

Syntax : git commit -m <file name>

GIT

Where “m “ refer to the message given after the changes committing from staging area to local repository



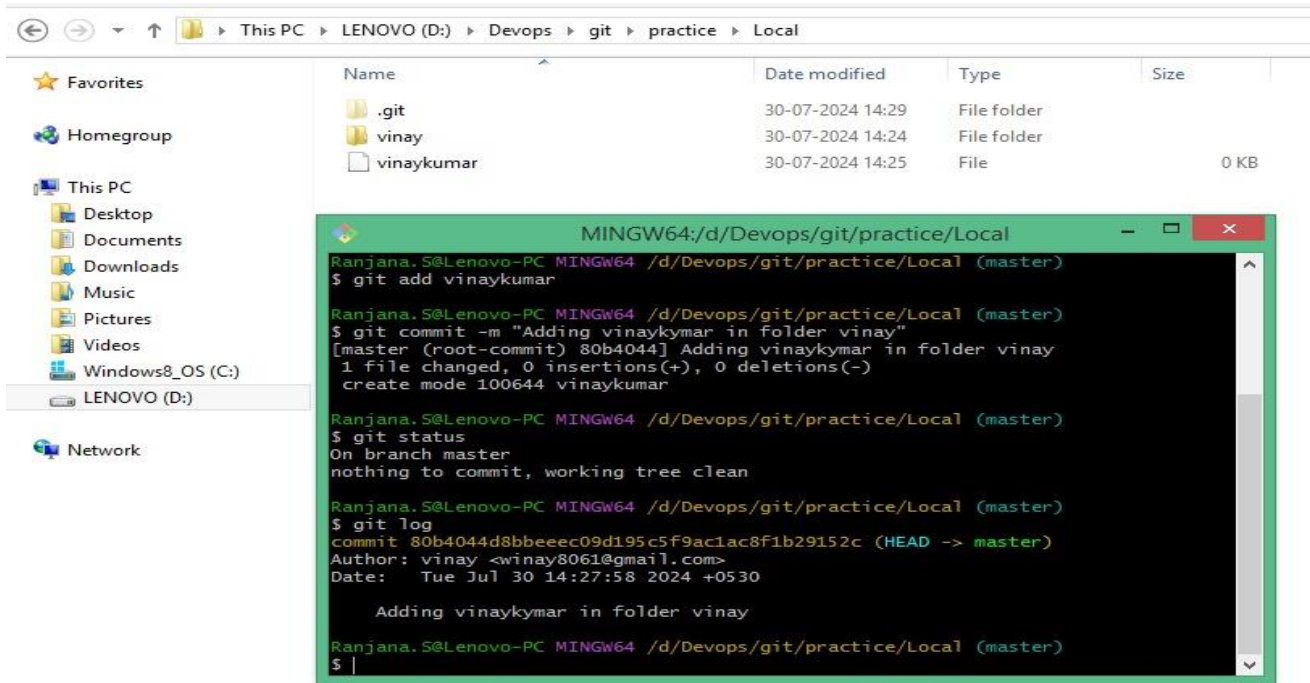
Git status : is used to check the status of the file/folder , like where it is exactly and whether it is added and committed

Syntax : git status

Git log : it is used to get the commit id , it will display the all the details like commit id , author, date and time of create and modification

Syntax : git log

GIT



Git branch : creating the new branch which is originated from main branch, it will be the replica of the main branch .

Syntax - git branch < branch name > - create the new branch

git branch -d <branch name> - for deleting the branch

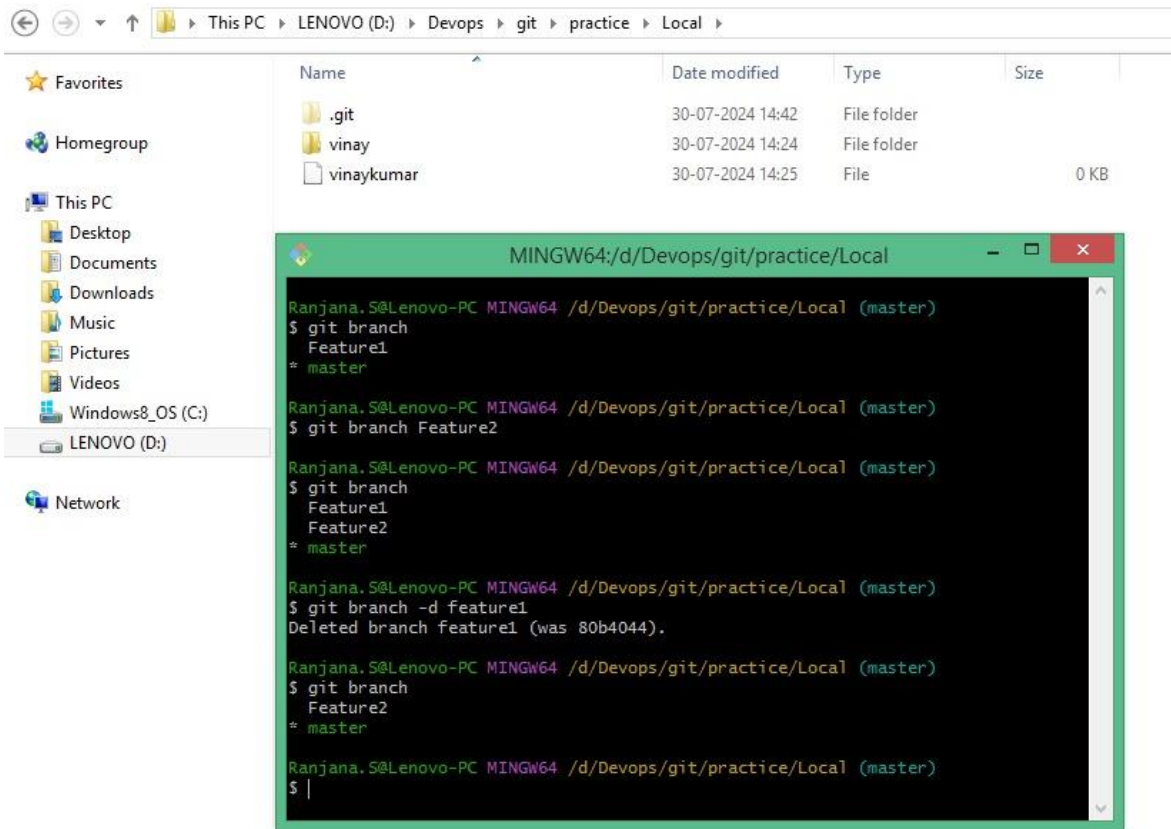
git branch - to list the branches in the repository

git checkout : switching from one branch to another in the repository

syntax : git checkout < branch name >

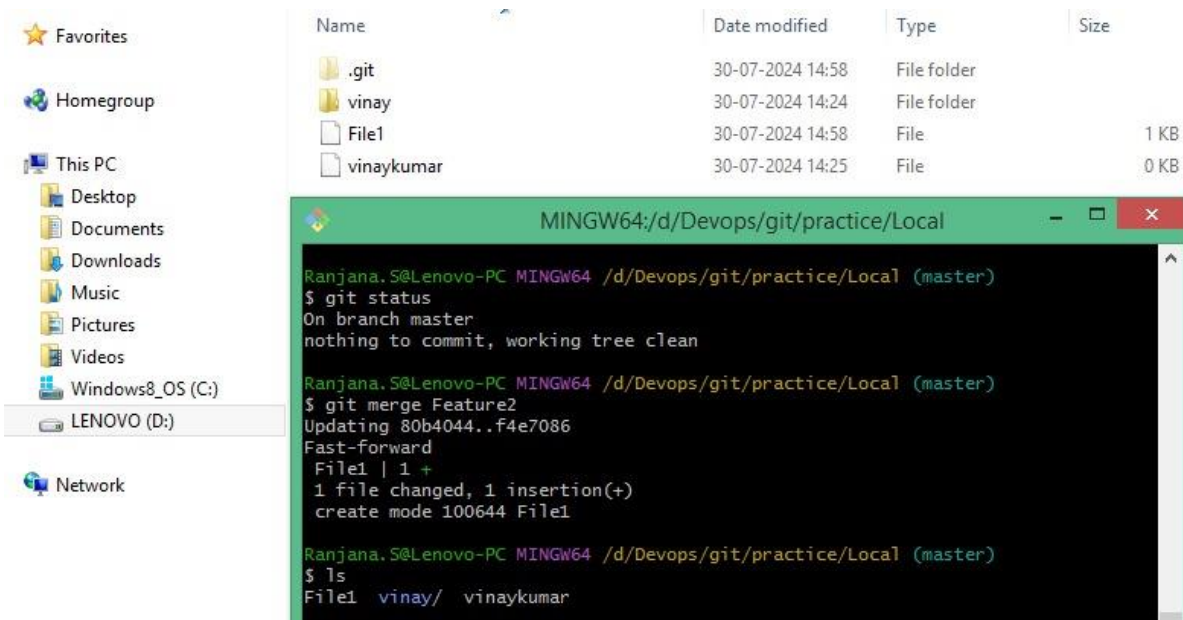
here we are switching from master branch to Feature2 branch

GIT



git merge : used to merge the two branches or merging feature branch to main branch after the new development or modification (after testing and confirmation that new feature is working as per client requirement)

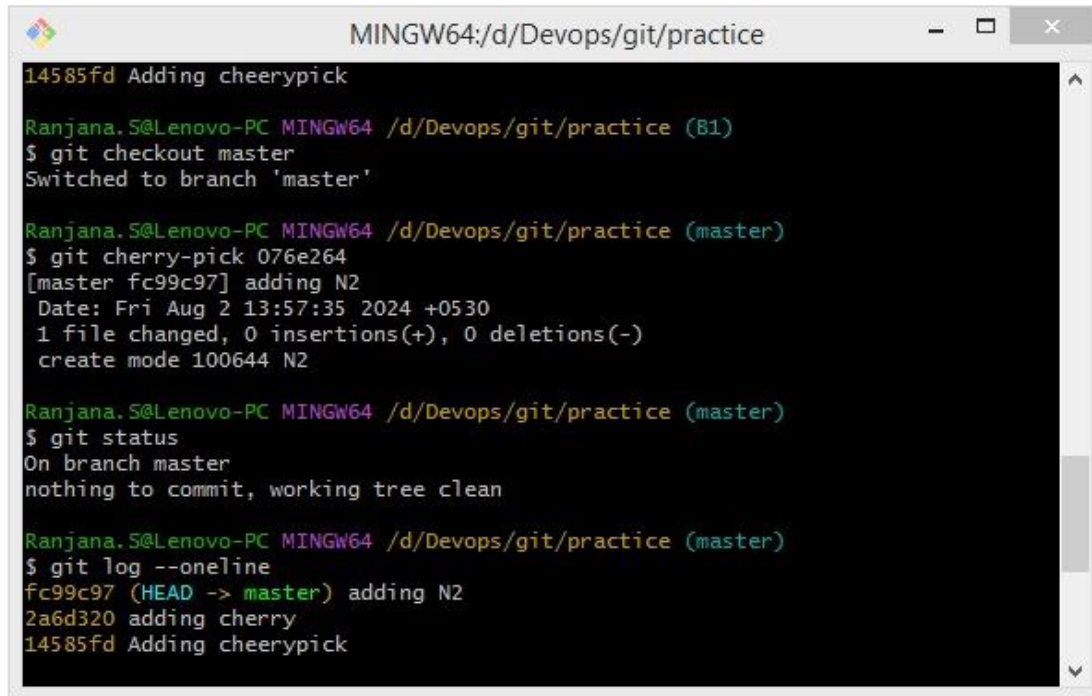
for merging any branch , the developer should be in main/ master branch



GIT

Git cherry-pick : it is used to pick up the particular commit ID from Branch to another Branch

Name	Date modified	Type	Size
.git	02-08-2024 14:00	File folder	
cherry	02-08-2024 13:54	File	0 KB
cherrypick	02-08-2024 13:51	File	0 KB
N2	02-08-2024 13:59	File	0 KB



```
MINGW64:/d/Devops/git/practice
14585fd Adding cheerypick

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (B1)
$ git checkout master
Switched to branch 'master'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git cherry-pick 076e264
[master fc99c97] adding N2
Date: Fri Aug 2 13:57:35 2024 +0530
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 N2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
fc99c97 (HEAD -> master) adding N2
2a6d320 adding cherry
14585fd Adding cheerypick
```

Git reset : is used to delete the latest commit or making it to settle in stageing adra or I working area

It has three stages i:e,

git reset --hard HEAD~1 ----> It will completely delet the file /commit id

git reset --mixed HEAD~1 -->it will move the file /commit id to the staging area

git reset --soft HEAD~1 ----> it will move the file /commit id to the working space

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
e3bd255 (HEAD -> master) add file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git reset --soft HEAD~1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file3

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file3
```

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
d0096f5 (HEAD -> master) adding updated file3
ffdd9dd adding updayed file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git reset --mixed HEAD~1
Unstaged changes after reset:
M       file3

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file3
```

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
e3bd255 (HEAD -> master) add file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git reset --hard HEAD~1
HEAD is now at 060aeda adding file3

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
060aeda (HEAD -> master) adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ vi file3
```

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
ffdd9dd (HEAD -> master) adding updayed file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git revert 1bd717a
hint: Waiting for your editor to close the file... |
```



```
1 Revert "add file2"
2
3 This reverts commit 1bd717a274b067e8c120ea3521a9bc69d3b13a2c.
4
5 # Please enter the commit message for your changes. Lines starting
6 # with '#' will be ignored, and an empty message aborts the commit.
7 #
8 # On branch master
9 # Changes to be committed:
10 #   deleted:   file2
11 #
12
```

Git Revert : it will change the any commit id of the file , not only the recent commit and it will add the extra commit for revert message created during the revert process ,

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
ffdd9dd (HEAD -> master) adding updayed file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git revert 1bd717a
[master cf337fb] kindly uodate the Revert "add file2"
1 file changed, 2 deletions(-)
delete mode 100644 file2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
cf337fb (HEAD -> master) kindly uodate the Revert "add file2"
ffdd9dd adding updayed file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1
```

Difference between git reset and git revert

	Git reset	Git revert
1	it can be done only for resent changes	it can be done on any commit ID
2	entire commit id will get deleted	it creates an extra commit ID for revert
3	syntax :	
	git reset --hard HEAD~1	git revert < commit ID>
	git reset --soft HEAD~1	
	git rest --mixed HEAD~1	

MERGE CONFLICT : it is generated when there is no aupdate between the developer

Eg : Developer 1 s working on File1 in feature 1 and Developer 2 is alos working on File1 in Feature2

Now developer 1 is completed the work and merged the Featute 1 with main branch and when the Developer 2 is about to merge file2 with feature 2 update , it will show the merge conflict

Shown In below pic

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git branch feature1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add feature1
fatal: pathspec 'feature1' did not match any files

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git branch feature2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git checkout feature1
Switched to branch 'feature1'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ vi file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git add file1
warning: in the working copy of 'file1', LF will be replaced by CRLF the next ti
me Git touches it

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git commit -m "adding updated file1 in feature1"
[feature1 388852a] adding updated file1 in feature1
1 file changed, 1 insertion(+), 1 deletion(-)

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git status
On branch feature1
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git checkout master
Switched to branch 'master'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git merge feature1
Updating cf337fb..388852a
Fast-forward
 file1 | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git merge feature1
Updating cf337fb..388852a
Fast-forward
 file1 | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
388852a (HEAD -> master, feature1) adding updated file1 in feature1
cf337fb (feature2) kindly uodate the Revert "add file2"
ffdd9dd adding updayed file3
060aeda adding file3
1bd717a add file2
4ff8c74 adding file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git checkout feature2
Switched to branch 'feature2'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature2)
$ vi file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature2)
$ git add file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature2)
$ git commit -m 'adding file1 with feature 2 update'
> ^C

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature2)
$ git commit -m "adding file1 with feature 2 update"
[feature2 d9bca7b] adding file1 with feature 2 update
 1 file changed, 1 insertion(+), 1 deletion(-)

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature2)
$ git status
On branch feature2
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature2)
$ git checkout master
Switched to branch 'master'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git merge feature2
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then commit the result.

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master|MERGING)
$ vi file1
```

GIT

```
hello file1
<<<<<< HEAP
wellcome to OPQ
=====
new change  file2 by feature2
>>>>>> feature2
~
~
~
~
~
~
~
~
~
~
~
~
```

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git merge feature2
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then commit the result.

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master|MERGING)
$ vi file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master|MERGING)
$

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master|MERGING)
$ vi file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master|MERGING)
$ git add file1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master|MERGING)
$ git commit -m "adding file two after the merge conflict recover"
[master 21b7d08] adding file two after the merge conflict recover

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ |
```

GIT MERGE :

Used to merge the branches During merging the commit will not be removed instead it will generate the new commit id

for merging feature

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice
$ git init
Initialized empty Git repository in D:/Devops/git/practice/.git/

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ touch t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ touch t2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git commit -m 'adding t1'
> ^C

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git commit -m "adding t1"
[master (root-commit) a16c1a7] adding t1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add t2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git commit -m "adding t2"
[master bb3f416] adding t2
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
bb3f416 (HEAD -> master) adding t2
a16c1a7 adding t1
```

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git branch feature1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git checkout feature1
Switched to branch 'feature1'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ touch t3

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git add t3

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git commit -m "adding t3"
[feature1 4c42b4b] adding t3
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t3

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ touch t4

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git add t4

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git commit -m "adding t4"
[feature1 9fa065a] adding t4
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t4

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git log --oneline
9fa065a (HEAD -> feature1) adding t4
4c42b4b adding t3
bb3f416 (master) adding t2
a16c1a7 adding t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git checkout master
Switched to branch 'master'
```


GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git checkout master
Switched to branch 'master'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ touch t5

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add t5

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git commit -m "adding t5"
[master 56b10c8] adding t5
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t5

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ touch t6

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add t6

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git commit -m "adding t6"
[master 30e2be3] adding t6
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t6

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
30e2be3 (HEAD -> master) adding t6
56b10c8 adding t5
bb3f416 adding t2
a16c1a7 adding t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git merge feature1
Merge made by the 'ort' strategy.
t3 | 0
t4 | 0
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 t3
create mode 100644 t4

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
bbc328b (HEAD -> master) yes Merge branch 'feature1'
30e2be3 adding t6
56b10c8 adding t5
9fa065a (feature1) adding t4
4c42b4b adding t3
bb3f416 adding t2
a16c1a7 adding t1
```

Difference between merge and rebase

GIT

	Merge	Rebase
1	History will not be re written	History will be re written
2	it gives a separate branch for merge in graph	it gives a linear view in the graph
3	easy to understand	bit difficult to understand
4	new commit d generated for merge	no extra commit is generated , instead it will replace the commit with new ID

Git rebase

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git reset --hard HEAD~!
fatal: ambiguous argument 'HEAD~!': unknown revision or path not in the working
tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git reset --hard HEAD~1
HEAD is now at 30e2be3 adding t6

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
30e2be3 (HEAD -> master) adding t6
56b10c8 adding t5
bb3f416 adding t2
a16c1a7 adding t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git checkout feature1
Switched to branch 'feature1'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git log --oneline
9fa065a (HEAD -> feature1) adding t4
4c42b4b adding t3
bb3f416 adding t2
a16c1a7 adding t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (feature1)
$ git checkout master
Switched to branch 'master'

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git rebase feature1
Successfully rebased and updated refs/heads/master.

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
e4578d5 (HEAD -> master) adding t6
e1ad26c adding t5
9fa065a (feature1) adding t4
4c42b4b adding t3
bb3f416 adding t2
a16c1a7 adding t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ |
```

git diff :

GIT

It will display the difference between the file when it is working space and when it is in the staging area

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git file
git: 'file' is not a git command. See 'git --help'.

The most similar commands are
  cat-file
  ls-files

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git ls
git: 'ls' is not a git command. See 'git --help'.

The most similar command is
  lfs

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ ls
t1 t2 t3 t4 t5 t6

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ vi t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git diff
warning: in the working copy of 't1', LF will be replaced by CRLF the next time Git touches it
diff --git a/t1 b/t1
index e69de29..9b07203 100644
--- a/t1
+++ b/t1
@@ -0,0 +1,2 @@
+
+Hello world

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ |
```

Git stash : moving the file or directory from staging area to some secret area

Git pop : moving file from stash area to local directory

Git apply : it is same as git pop, but after committing it will still be in the stash area and also get committed...whereas in git pop it will permanently get removed from the stash area

GIT

File Explorer view of the directory: This PC > LENOVO (D:) > Devops > git > practice

Name	Date modified	Type	Size
.git	30-07-2024 20:33	File folder	
f1	30-07-2024 20:32	File	0 KB

Terminal window (MINGW64) showing Git commands and output:

```
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ touch f2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add f2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git stash
Saved working directory and index state WIP on master: ccac279 add t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git stash list
stash@{0}: WIP on master: ccac279 add t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$
```

File Explorer view of the directory: This PC > LENOVO (D:) > Devops > git > practice

Name	Date modified	Type	Size
.git	30-07-2024 20:33	File folder	
f1	30-07-2024 20:32	File	0 KB

Terminal window (MINGW64) showing Git commands and output:

```
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ touch f2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git add f2

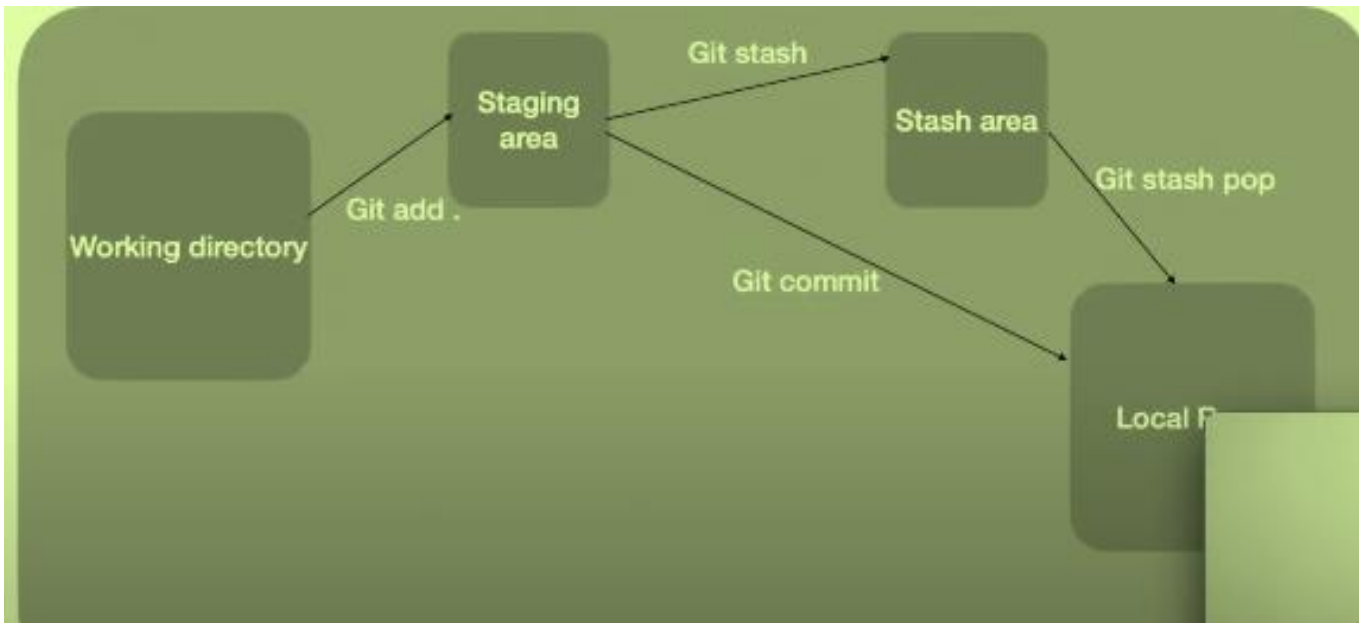
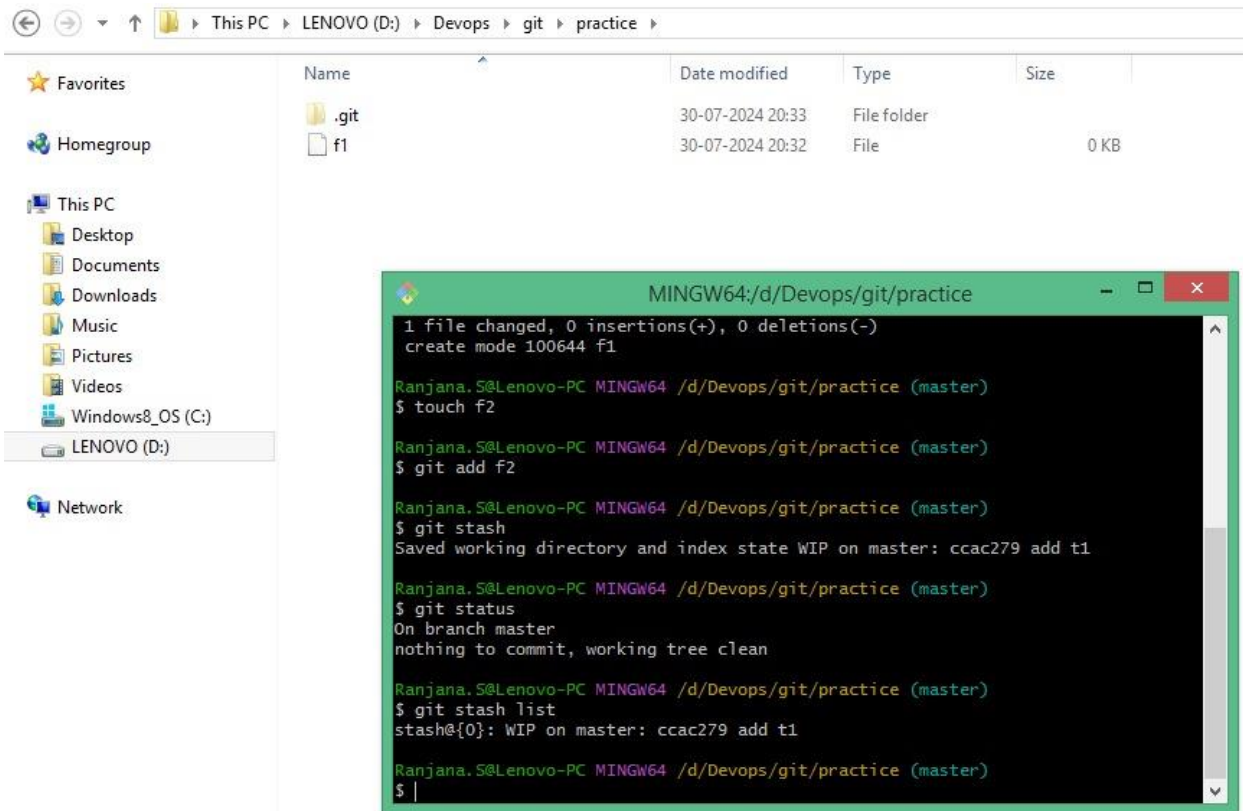
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git stash
Saved working directory and index state WIP on master: ccac279 add t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git status
On branch master
nothing to commit, working tree clean

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git stash list
stash@{0}: WIP on master: ccac279 add t1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$
```

GIT



Git tag :

GIT

it is a name given to set of versions of files and directories, it is easy to remember the tag names, it indicates milestone of a project.

There are Two types

Lightweight tag : it just gives the tag name

Annotated tag : it will give the details of tag like author , date & time

Name	Date modified	Type	Size
.git	02-08-2024 15:14	File folder	
T1	02-08-2024 15:14	File	1 KB
T2	02-08-2024 15:05	File	0 KB

```
MINGW64:/d/Devops/git/practice

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
08a5524 (HEAD -> master) adding T2
76cb12c Adding T1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git tag

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git tag V1.0.0

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git tag
V1.0.0

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
08a5524 (HEAD -> master, tag: V1.0.0) adding T2
76cb12c Adding T1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ |
```

Git annotated tag

GIT

```
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
62ebbd6 (HEAD -> master) Adding T4
32473c4 Adding t3
08a5524 (tag: V1.0.0) adding T2
76cb12c Adding T1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git tag -a v2.0.0

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
62ebbd6 (HEAD -> master, tag: v2.0.0) Adding T4
32473c4 Adding t3
08a5524 (tag: V1.0.0) adding T2
76cb12c Adding T1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git show -v2.0.0
fatal: unrecognized argument: -v2.0.0

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git show v2.0.0
tag v2.0.0
Tagger: vinay <winay8061@gmail.com>
Date:   Fri Aug 2 15:29:05 2024 +0530

Adding "Annotatedtag"

commit 62ebbd671ecb4c07ffe708b2d8865eac86b9c500 (HEAD -> master, tag: v2.0.0)
Author: vinay <winay8061@gmail.com>
Date:   Fri Aug 2 15:23:45 2024 +0530




    Adding T4

diff --git a/T4 b/T4
new file mode 100644
index 0000000..58fa7b1
--- /dev/null
+++ b/T4
@@ -0,0 +1 @@
+hello T4

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ clear
```

Git checkout tag

GIT

.git			
	.git	02-08-2024 15:32	File folder
	T1	02-08-2024 15:14	File 1 KB
	T2	02-08-2024 15:05	File 0 KB

```
MINGW64:/d/Devops/git/practice

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git checkout V1.0.0
Note: switching to 'V1.0.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -






Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 08a5524 adding T2

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice ((V1.0.0))
$ |
```

GIT

Tag delete

 .git	02-08-2024 15:34	File folder
 T1	02-08-2024 15:14	File
 T2	02-08-2024 15:05	File
 T3	02-08-2024 15:33	File
 T4	02-08-2024 15:33	File

```
MINGW64:/d/Devops/git/practice
Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
62ebbd6 (HEAD -> master, tag: v2.0.0) Adding T4
32473c4 Adding t3
08a5524 (tag: V1.0.0) adding T2
76cb12c Adding T1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git tag -d V1.0.0
Deleted tag 'V1.0.0' (was 08a5524)

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ GIT LOG --ONELINE
git: 'LOG' is not a git command. See 'git --help'.

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ git log --oneline
62ebbd6 (HEAD -> master, tag: v2.0.0) Adding T4
32473c4 Adding t3
08a5524 adding T2
76cb12c Adding T1

Ranjana.S@Lenovo-PC MINGW64 /d/Devops/git/practice (master)
$ |
```

Branching strategy :

1. Master Branch (main): This branch represents the stable production code. It's where the latest release-ready code resides.

2. Develop Branch:

Also known as the integration branch, this branch serves as the main integration point for ongoing development work. • It contains code that is actively being developed and integrated from feature branches.

GIT

3. Feature Branches:

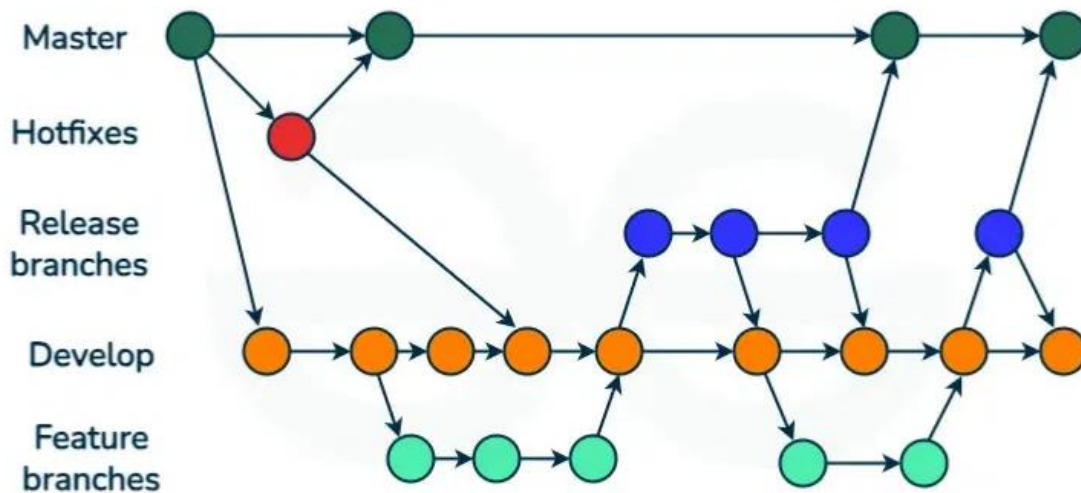
Each feature or piece of work is developed in its own branch derived from the development branch (Dev Branch). Once development is complete, the feature branch is merged back into the development branch through a pull request or merge request process.

4. Release Branches:

When develop is ready for release, a new branch (release/x.x) is created. This branch allows for final testing and preparation before merging into the master. Once finalized, the release branch is merged into the main branch for deployment.

5. Hotfix Branches:

If bugs are found in the master branch, a hotfix branch (hotfix/x.x.x) is created from master. Once fixed, it's merged back into both master and develop.



	git remote	git clone
1	it is used to create the copy	it is used to manage the connection with other repo
2	it is used at the time of biginig	it is used at every stage of the process

Difference between Git merge and git rebase

	Merge	Rebase
1	History will not be re written	History will be re written
2	it gives a separate branch for merge in graph	it gives a linear view in the graph
3	easy to understand	bit difficult to understand

GIT

4	new commit d generated for merge	no extra commit is generated , instead it will replace the commit with new ID
---	----------------------------------	---

Build tool : it is a tool which converts the code in to executable file..

we have a particular build tool for particular application

Application	Tool	Looks For
Java	maven/ANT	.XML
C/C++	MAKE	MAKE
PYTHAN	BAZEL	.PY
ANDROID	GRADEL	
NODJS		.JSON
SBT		.SCALE

Function of Build tool

compilation : converting the cod into to machine readable form

linking : combining the complied file in to a single file

dependency management : downloading the all the software which are required to carry out the process

testing : it will under take a unit testing

packing : it package the complied file in to format suitable for development

automation : Automate the repetitive task like developing , testing and deploying .

Maven : it is a tool used to build and manage the java based project

Maven life cycle :

VALIDATE : It will check and download the all dependency software require to run the project

compile : concert the code in to executable file

GIT

test :it runs the unit test

packaging : convert the file in to a distributable form

integrating testing : it will run the integration test on the project

verify : it will verify the project

Install : install the project in to local repo

deploy : install the package code in to remote repo , so that all developer can work on it.