Topic-21

Working with Remote Repositories

Topic-21: Working with Remote Repositories

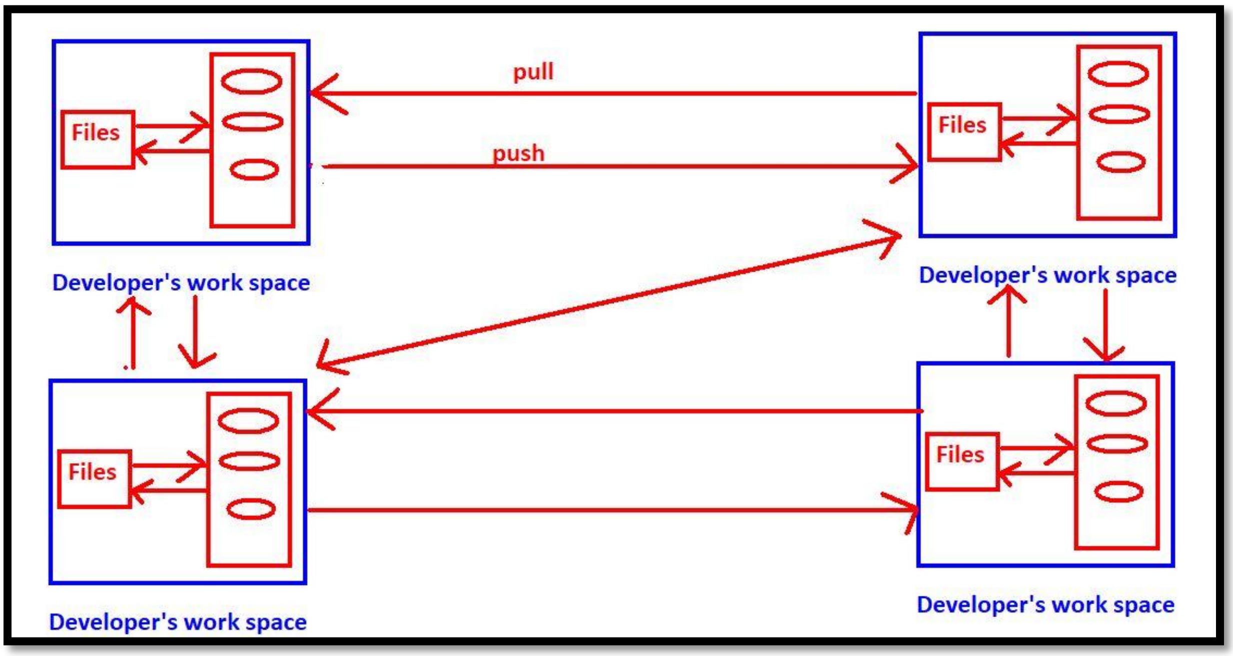
21.1) Need of Remote Repositories:

Upto this we created multiple files and we did several changes. We used multiple git commands. All these operations happend in local repository.

If we required to share our code to the peer developers/team members then the remote repository concept will come in the picture.

As GIT is distributed version control system, every developer has his own local repository. Every developer can share his code to the peer developers/team members.

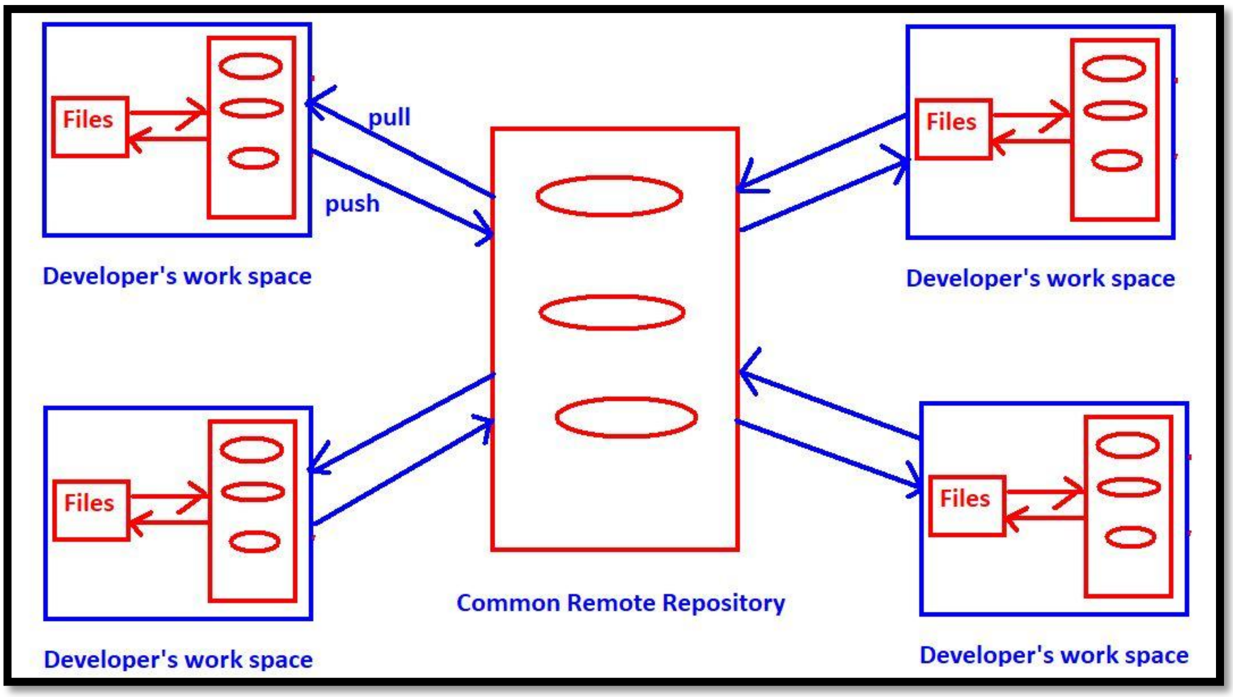
By using push operation, developer can share his code the peer developer.
By using pull operation, developer can get the code of peer developer.



The Problems if Developer communicates directly with other Developers:

1. On every developer's machine we have to install Git Server.
2. Every developer should aware hostname and port number of all remaining developers, then only he can share the code.
3. If there is any change in port number of any developer's machine, intimating this for all developers every time is very difficult task.

We can solve these problems with common remote repository.



Note: **It is not centralized version control system**

The Advantages of Common Remote Repositories:

1. On the developer's machine, we are not required to install GIT server.
GIT Server is available on common remote repository.
2. Developer should aware only url of the common remote repository and he is not required to aware all hostnames and port numbers of all remaining developer's machines.

Note:
1. Using common remote repository is best practice in real time.
2. The most common service providers of remote repositories
Github,Gitlab,Bitbucket etc
3. Some big companies may use their own common remote repository instead of using 3rd party repositories.
4. Sometimes in Real time we may required to work with multiple remote repositories also.

21.2) How to work with GitHub?

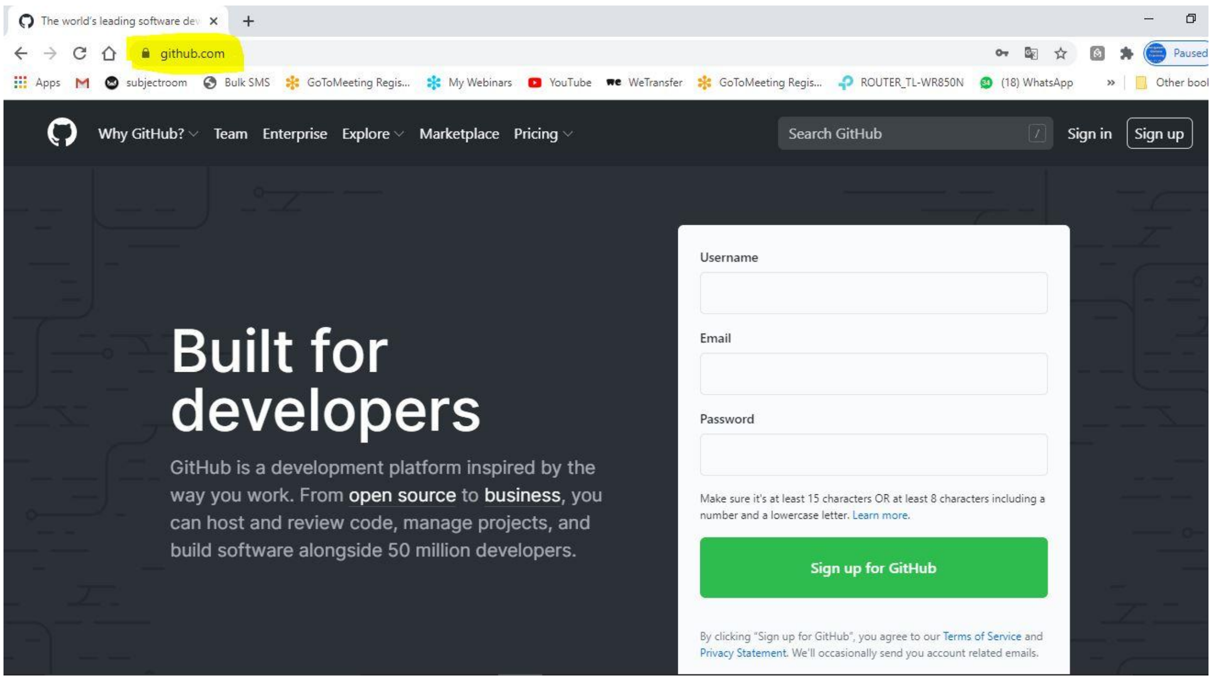
1. How to create account in the Github
2. How to create Remote Repository in the Github
3. Various Git commands to work with remote repository

    git remote
    git push
    git clone
    git pull
    git fetch

Github is an online repository store/remote repository service provider.
We can store our repositories in github so that our code is available for the remaining team members.

21.3) How to Create Account in GitHub?

https://github.com



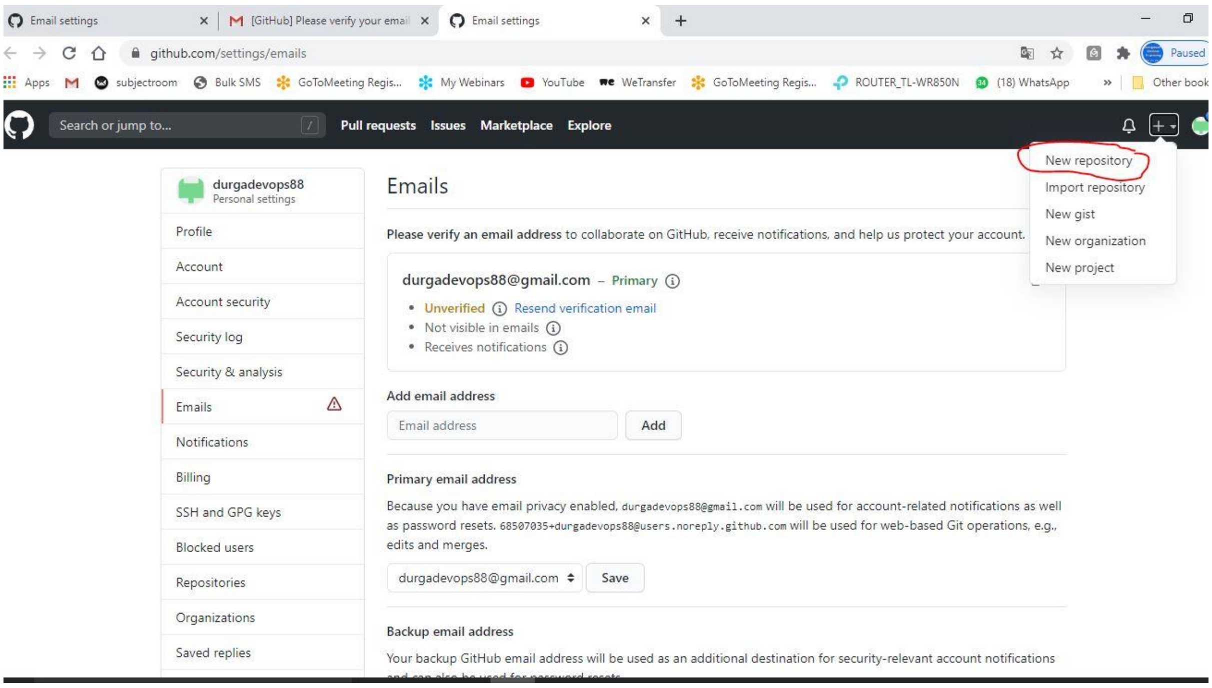
GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

By clicking "Sign up for GitHub", you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account related emails.

We have to signup with our mail id. We should use valid mail id only because we have to verify that mail id to create account successfully.
github requires email verification.

21.4) How to Create a New Reposioty in Github:



Emails

Please verify an email address to collaborate on GitHub, receive notifications, and help us protect your account.

durgadevops88@gmail.com – Primary

- Unverified Resend verification email
- Not visible in emails
- Receives notifications

Add email address

Email address Add

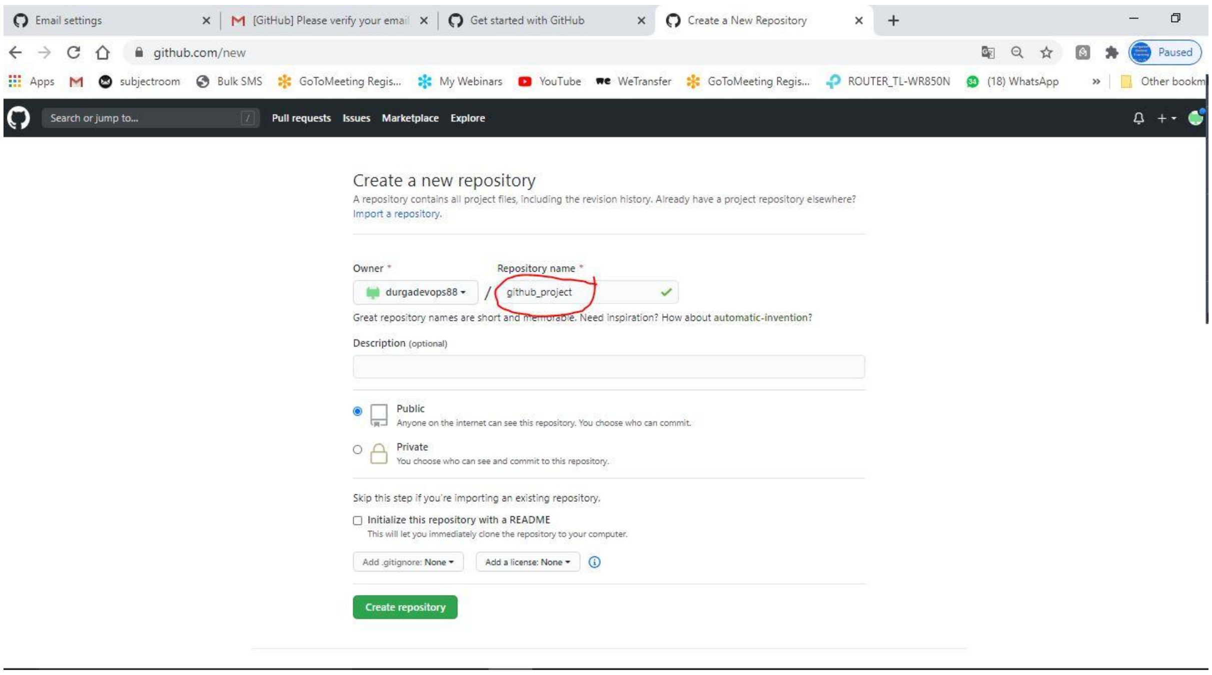
Primary email address

Because you have email privacy enabled, durgadevops88@gmail.com will be used for account-related notifications as well as password resets. 68507035+durgadevops88@users.noreply.github.com will be used for web-based Git operations, e.g., edits and merges.

durgadevops88@gmail.com Save

Backup email address

Your backup GitHub email address will be used as an additional destination for security-relevant account notifications



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * durgadevops88 / Repository name * github_project

Great repository names are short and memorable. Need inspiration? How about automatic-invention?

Description (optional)

(●) Public
Anyone on the internet can see this repository. You choose who can commit.

( ) Private
You choose who can see and commit to this repository.

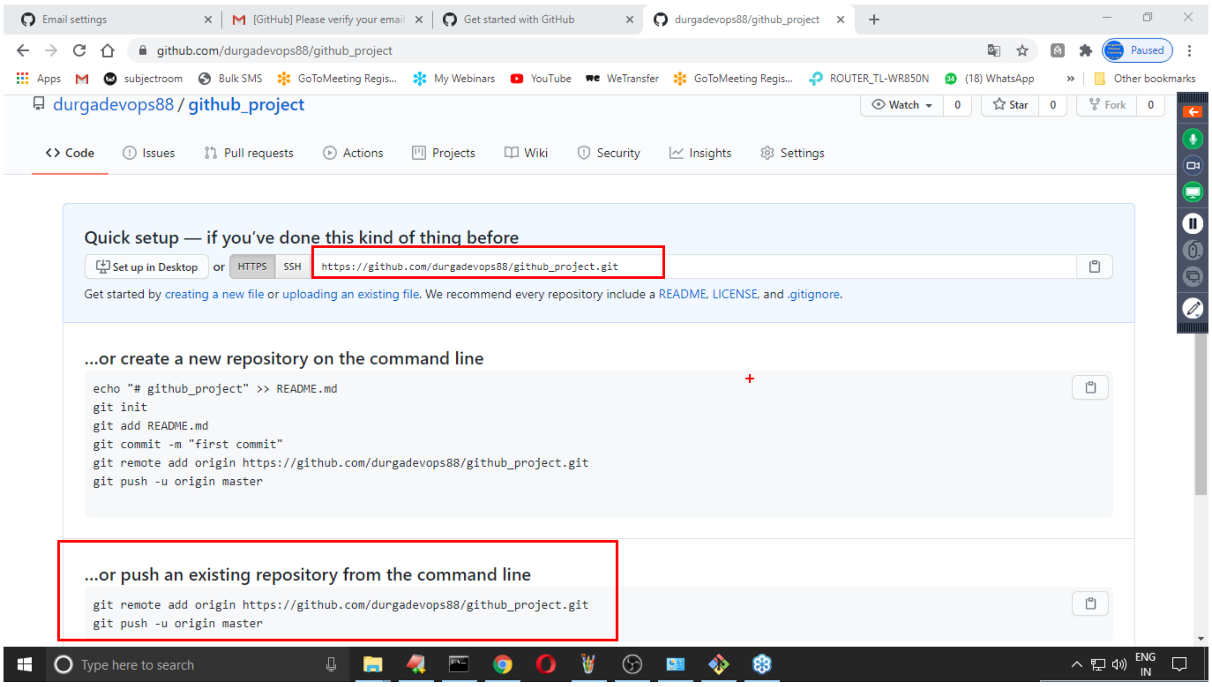
Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None Add a license: None

Create repository

Repository Name is the name of the project folder. We have to choose meaningfull name.



Quick setup — if you've done this kind of thing before

https://github.com/durgadevops88/github_project.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# github_project" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/durgadevops88/github_project.git
git push -u origin master
```

…or push an existing repository from the command line

```
git remote add origin https://github.com/durgadevops88/github_project.git
git push -u origin master
```

This is the url of the remote repository
https://github.com/durgadevops88/github_project.git

By using this url only we can communicate with remote repository.

<u>Note:</u>
1. The max allowed file size in Github public repository: 100MB
2. The max allowed repo size in Github public repository: 2GB

21.5) How to work with Remote Repository:

To work with remote repository we have to use the following commands

1. **git remote**
2. **git push**
3. **git clone**
4. **git pull**
5. **git fetch**

1) git remote:

We can use git remote command to configure remote repository to our local repository.

```
$ git remote add <alias_name>  remote_repository_url
```

Eg:

```
$ git remote add origin https://github.com/durgadevops88/github_project.git
```

Here, origin is alias name of the remote_repository_url. By using this alias name only we can communicate with remote repository.
Instead of origin we can use any name, but it is convention to use origin.

By using git remote command we can also view remote repository information.

$ git remote

It just provides the alias names of remote repositories

$ git remote -v

It provides remote repository urls also.

2) git push:

We can use git push command to send our changes from local repository to remote repository. ie to push our changes from local repo to remote repo.

```
git push <remote_name>  <branch_name>
```

Eg: git push origin master

Demo Example:

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ mkdir remoterepo

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ cd remoterepo

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo
$ git init
Initialized empty Git repository in D:/gitprojects/remoterepo/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ echo 'First Line' > file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
```

```
$ git add . ; git commit -m 'first commit'
[master (root-commit) 0741376] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ echo 'First Line' > file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git add .;git commit -m 'second commit'
[master 553dc63] second commit
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git status
On branch master
nothing to commit, working tree clean

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git log --oneline
553dc63 (HEAD -> master) second commit
0741376 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git log
commit 553dc63d050c6ef6bd4567fddb6903510c47cb28 (HEAD -> master)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 19 20:28:47 2020 +0530

    second commit

commit 074137633893c093572e4a35923579ac9dbf61be
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 19 20:27:53 2020 +0530

    first commit
```

Being a Developer, i want to share to this code to my peer developers. We have to send this code to the remote repository, so that it is available for them.

We have to configure remote repository to our local repository. For this we have to use git remote command.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
```

```
$ git remote add origin https://github.com/durgadevops88/github_project.git
```

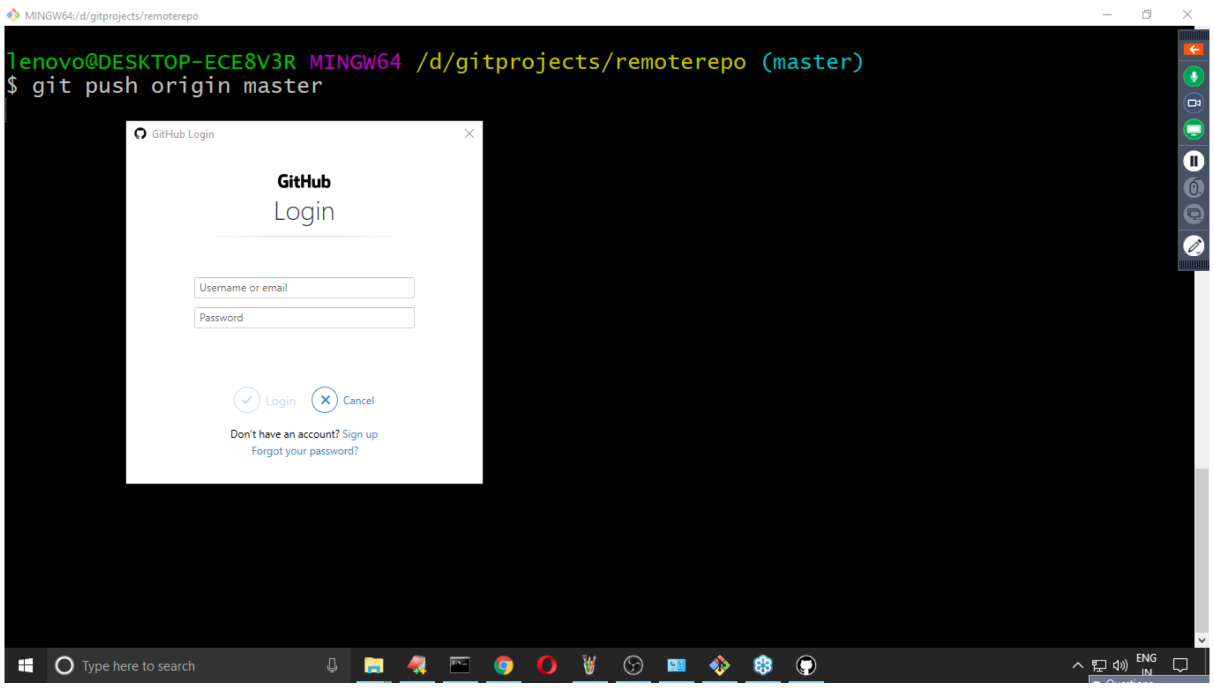
We can check whether it is configured or not by using git remote command.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git remote
origin

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git remote -v
origin  https://github.com/durgadevops88/github_project.git (fetch)
origin  https://github.com/durgadevops88/github_project.git (push)
```

We have to push our changes from local repository to remote repository.
For this we have to use git push command.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 441 bytes | 441.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/durgadevops88/github_project.git
 * [new branch]      master -> master
```



```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git push origin master
```

Note: Whenever we are using remote repository the communication will be happend between local and remote repositories only.

Q) Is it possible to have multiple Remote Repositories?

Yes

Assume we created a new repository named with github_project2 and its url is https://github.com/durgadevops88/github_project2.git

We have to send the files of local repository to this remote repository also.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git remote add my_remote https://github.com/durgadevops88/github_project2.git

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git remote
my_remote
origin

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git remote -v
my_remote       https://github.com/durgadevops88/github_project2.git (fetch)
my_remote       https://github.com/durgadevops88/github_project2.git (push)
origin  https://github.com/durgadevops88/github_project.git (fetch)
origin  https://github.com/durgadevops88/github_project.git (push)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/remoterepo (master)
$ git push my_remote master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 441 bytes | 441.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/durgadevops88/github_project2.git
 * [new branch]      master -> master
```

Note: By using settings tab of remote repository we can perform multiple activities like changing ownership,delete repository etc.

3) git clone:

cloning means creating exactly duplicate copy.

Already there is a project on remote repository.
Being a new team member, we may required complete remote repository into our local repository. For this purpose we have to use git clone command.
i.e we can use git clone command to create local repository with remote repository files.
All the files and commit history will be copied from remote repository into local repository.

Syntax: git clone <remote_repo_url>

Eg:
git clone https://github.com/durgadevops777/github_project.git
Now the project name will become repository project name.

Note: Based on our requirement we can provide a new name for the project.
git clone <remote_repo_url> <new_project_name>

Eg: git clone https://github.com/durgadevops777/github_project.git my_project

Demo Example-1 :

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d
$ mkdir newdeveloper

lenovo@DESKTOP-ECE8V3R MINGW64 /d
$ cd newdeveloper/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ ls

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ git clone https://github.com/durgadevops777/github_project.git
Cloning into 'github_project'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ ls
github_project/
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ cd github_project/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper/github_project (master)
$ ls -la
total 6
drwxr-xr-x 1 lenovo 197121  0 Jul 22 20:43 ./
drwxr-xr-x 1 lenovo 197121  0 Jul 22 20:43 ../
drwxr-xr-x 1 lenovo 197121  0 Jul 22 20:43 .git/
-rw-r--r-- 1 lenovo 197121 12 Jul 22 20:43 file1.txt
-rw-r--r-- 1 lenovo 197121 12 Jul 22 20:43 file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper/github_project (master)
$ git log
commit b918d0d3d2560e53e6260f838a882b87a8ec45ee (HEAD -> master, origin/master,
origin/HEAD)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 19 21:51:35 2020 +0530

    second commit

commit dc9556591b99a10a7e0d99a8ae88b9fe99db1edf
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 19 21:51:05 2020 +0530

    first commit
```

Demo Example -2: With our own customized Project Folder Name

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ git clone https://github.com/durgadevops777/github_project.git my_project
Cloning into 'my_project'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ ls
github_project/  my_project/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper
$ cd my_project/
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/newdeveloper/my_project (master)
$ ls -la
total 7
drwxr-xr-x 1 lenovo 197121  0 Jul 22 20:52 ./
drwxr-xr-x 1 lenovo 197121  0 Jul 22 20:52 ../
drwxr-xr-x 1 lenovo 197121  0 Jul 22 20:52 .git/
-rw-r--r-- 1 lenovo 197121 25 Jul 22 20:52 _config.yml
-rw-r--r-- 1 lenovo 197121 12 Jul 22 20:52 file1.txt
-rw-r--r-- 1 lenovo 197121 12 Jul 22 20:52 file2.txt
```

Q1) Before using git clone command is it required to use git init command?

Ans: No
git clone command itself will create local repository and hence we are not required to use git init command explicitly.

Q2) In how many ways we can create a new local repository?

Ans: In 2 ways

1. By using git init command to create empty local repository.
2. By using git clone command to create local repository with the files of remote repository.

4) git fetch Command:

We can use get fetch command to check whether any updates available at remote repository or not. This command will retrieve only latest meta-data info from the remote repository. It won't download updates from remote repository into local repository.

Syntax: $ git fetch origin

5) git pull Command:

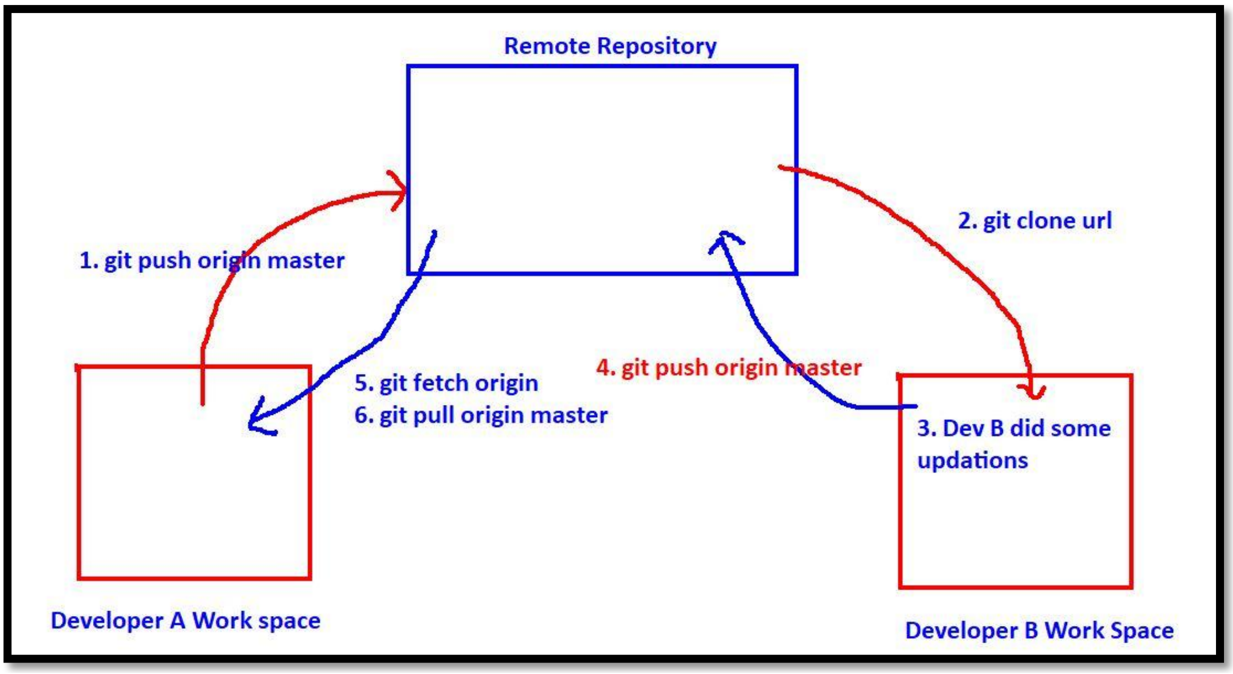
We can use git pull command to download and merge updates from remote repository into local repository.

git pull = fetch+merge

Syntax:

```
$ git pull <remote> <branch>
$ git pull origin master
```

Demo Example:


Remote Repository
1. git push origin master
2. git clone url
3. Dev B did some updations
4. git push origin master
5. git fetch origin
6. git pull origin master
Developer A Work space
Developer B Work Space

On Developer A Machine:

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d
$ mkdir dev_A_work_space

lenovo@DESKTOP-ECE8V3R MINGW64 /d
$ cd dev_A_work_space/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space
$ git init
Initialized empty Git repository in D:/dev_A_work_space/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ echo 'First line by Dev A' > file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ git add .;git commit -m 'first commit'
[master (root-commit) cc73176] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt
```

Now we have to create a remote repository in the github named with remote_repo.
The corresponding url is:
https://github.com/durgadevops777/remote_repo.git

push the changes from local repository to remote repository.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ git remote add origin https://github.com/durgadevops777/remote_repo.git

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/durgadevops777/remote_repo.git
 * [new branch]      master -> master
```

On Developer B Machine:

Developer B is responsible to clone this remote repository into local repository and he can do some updates.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d
$ mkdir dev_B_work_space

lenovo@DESKTOP-ECE8V3R MINGW64 /d
$ cd dev_B_work_space/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space
$ git clone https://github.com/durgadevops777/remote_repo.git
Cloning into 'remote_repo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space
$ ls
remote_repo/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space
$ cd remote_repo/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space/remote_repo (master)
$ ls
file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space/remote_repo (master)
$ git log --oneline
cc73176 (HEAD -> master, origin/master, origin/HEAD) first commit
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space/remote_repo (master)
$ echo 'Line 2 added by Dev B' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space/remote_repo (master)
$ echo "First line" > file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space/remote_repo (master)
$ git add .;git commit -m 'commit by devB'
[master 47eae54] commit by devB
 2 files changed, 2 insertions(+)
 create mode 100644 file2.txt
```

DevB has to update these changes to the remote repository by using git push command.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_B_work_space/remote_repo (master)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 332 bytes | 332.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/durgadevops777/remote_repo.git
   cc73176..47eae54  master -> master
```

Now updates are available on Remote repository.

How Developer A will get these updates from Remote repo?

Developer has to use git fetch and git pull commands.
git fetch → To check whether updates are available or not

git pull → To download and merge those updates with local repo.
We can use directly git pull without using git fetch.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ git fetch origin
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 312 bytes | 1024 bytes/s, done.
From https://github.com/durgadevops777/remote_repo
   cc73176..47eae54  master     -> origin/master
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ ls
file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ git pull origin master
From https://github.com/durgadevops777/remote_repo
 * branch            master     -> FETCH_HEAD
Updating cc73176..47eae54
Fast-forward
 file1.txt | 1 +
 file2.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ ls
file1.txt  file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ cat file1.txt
First line by Dev A
Line 2 added by Dev B

lenovo@DESKTOP-ECE8V3R MINGW64 /d/dev_A_work_space (master)
$ git log --oneline
47eae54 (HEAD -> master, origin/master) commit by devB
cc73176 first commit
```

Note:

1. To perform push operation, local repository and remote repository should be in sync. ie the local repository should be upto date to the remote repository.
If any updates in remote repo, git push is not allowed. It will ask us to pull first.

2. If any conflicts are there, we have to sit with peer developer to resolve the conflicts.

Topic-22

Git Tagging

(git Tag Command)

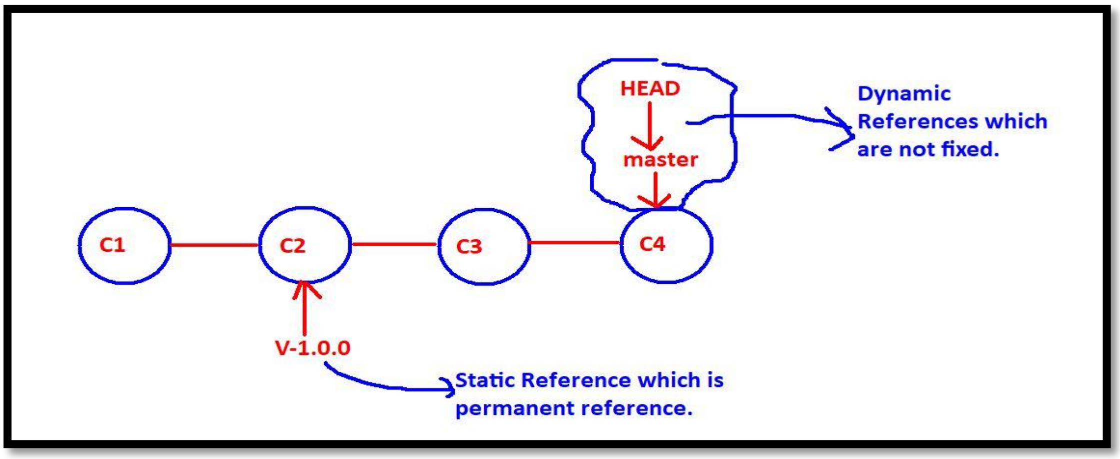
Topic-22: Git Tagging (git Tag Command)

22.1) Introduction to Tagging:

In our repository there may be a chance of multiple commits. HEAD/Branch is always pointing to latest commit and its value keep on changing as new commits happend.

Sometimes, we have to mark significant events or mile stones(like version numbering) with some label in our repository commits. We can do this labeling by using git tag command.

Tag is nothing but a label or mark to a particular commit in our repository.
Tag is static and permanent reference to a particular commit where as HEAD/Branch is a dynamic reference.
In general Tags can be used for release verions.


HEAD
master
Dynamic References which are not fixed.
C1
C2
C3
C4
V-1.0.0
Static Reference which is permanent reference.

There are two types of tags
1) Light Weight/Simple Tags
2) Annotated Tags [Tags with Information]

22.2) Creation of a Light Weight Tag:

We have to use git tag command.

Syntax: $ git tag <tag_name>

Eg: $ git tag V-1.0.0.0

V-1.0.0.0 tag acts as a permanent reference to the current latest commit. But based on our requirement we can define a tag for old commits also.

The tags will be stored in our repository inside tags folder(.git/refs/tags).

We can list available tags by using -l or --list option.

```
$ git tab -l
$ git tab --list
```

We can use tag directly where ever commit id is required in our commands.

Eg: git show <commitid>

For the specified commitid, if any tag is defined then we can use directly that tag instead of commit id.

```
git show tag_name
```

Demo Example:

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ mkdir tagging

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ cd tagging

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging
$ git init
Initialized empty Git repository in D:/gitprojects/tagging/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ echo 'First Line' > file1.txt
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git add .;git commit -m 'first commit'
[master (root-commit) 009784d] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ echo 'First Line' > file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git add .;git commit -m 'second commit'
[master 322d0ab] second commit
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ echo 'First Line' > file3.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git add .;git commit -m 'third commit'
[master a927d06] third commit
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ echo 'First Line' > file4.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git add .;git commit -m 'fourth commit'
[master 3350c49] fourth commit
```

```
 1 file changed, 1 insertion(+)
 create mode 100644 file4.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
3350c49 (HEAD -> master) fourth commit
a927d06 third commit
322d0ab second commit
009784d first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag V-1.0.0

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
3350c49 (HEAD -> master, tag: V-1.0.0) fourth commit
a927d06 third commit
322d0ab second commit
009784d first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ echo 'First Line' > file5.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git add .;git commit -m 'fifth commit'
[master 7e6aada] fifth commit
 1 file changed, 1 insertion(+)
 create mode 100644 file5.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
7e6aada (HEAD -> master) fifth commit
3350c49 (tag: V-1.0.0) fourth commit
a927d06 third commit
322d0ab second commit
009784d first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag --list
V-1.0.0
```

Instead of using commit id we can use the corresponding tag in our commands.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
```

```
7e6aada (HEAD -> master) fifth commit
3350c49 (tag: V-1.0.0) fourth commit
a927d06 third commit
322d0ab second commit
009784d first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git show 3350c49
commit 3350c492538f826818aab0f590146012d123f117 (tag: V-1.0.0)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Fri Jul 24 21:11:55 2020 +0530

    fourth commit

diff --git a/file4.txt b/file4.txt
new file mode 100644
index 0000000..603cb1b
--- /dev/null
+++ b/file4.txt
@@ -0,0 +1 @@
+First Line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git show V-1.0.0
commit 3350c492538f826818aab0f590146012d123f117 (tag: V-1.0.0)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Fri Jul 24 21:11:55 2020 +0530

    fourth commit

diff --git a/file4.txt b/file4.txt
new file mode 100644
index 0000000..603cb1b
--- /dev/null
+++ b/file4.txt
@@ -0,0 +1 @@
+First Line
```

All tags information will be stored in .git/refs/tags/ folder.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ cd .git/refs/tags/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging/.git/refs/tags (GIT_DIR!)
$ ls
V-1.0.0
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging/.git/refs/tags (GIT_DIR!)
$ cat V-1.0.0
3350c492538f826818aab0f590146012d123f117
```

22.3) How to delete a Tag?

We have to use git tag command with -d or --delete option.
git tag -d <tagname>

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag --list
V-1.0.0

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag --delete V-1.0.0
Deleted tag 'V-1.0.0' (was 3350c49)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag --list

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
7e6aada (HEAD -> master) fifth commit
3350c49 fourth commit
a927d06 third commit
322d0ab second commit
009784d first commit
```

Summary:

```
git tag <tagname>
git tag --list
git tag --delete <tagname>
```

Note: Within the repository, tag name should be unique.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging/.git/refs/tags (GIT_DIR!)
$ git tag V-1.0.0
fatal: tag 'V-1.0.0' already exists
```

Symantic Versioning:

Almost all software products follow Symantic Versioning.

V 1.2.12
Where 1 is considered as Major version.
2 is considered as Minor version.
12 is considered as Patch version.

We can get more info on semantic versioning from the following url: https://semver.org/

Note: First Alpha Release and then Beta Release and then finally original major version release.

Limitation of LightWeight Tags:

LightWeight Tag is simply text reference to the specified commit. It won't maintain any information like tagger name, date of creation, message etc.

If we want maintain such type of information, we should go for Annotated Tags.

22.4) Annotated Tags (Tags with Information)

Annotated Tag is exactly same as Light weight tag except that it maintains information like tagger name, date of creation, description etc.
Annotated Tag internally maintained as object form in git repository.
Annotated tags will be stored in .git/refs/tags folder and .git/objects folder.

We can create annotated tag by using git tag command with -a option.

```
$ git tag -a <tagname>
```

Here -a indicates that it is annotated tag.
Now default editor will be opened for tag message.

```
$ git tag -a <tagname> -m <tag_message>
```

Demo Example:

```
$ git log --oneline
f7ec107 (HEAD -> master) fifth commit
2a1c1f4 (tag: V-1.0.0) fourth commit
b83ee22 third commit
ce4a690 second commit
121a2cb first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag -a V-1.1.0 -m 'Release 1.1.0'
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
f7ec107 (HEAD -> master, tag: V-1.1.0) fifth commit
2a1c1f4 (tag: V-1.0.0) fourth commit
b83ee22 third commit
ce4a690 second commit
121a2cb first commit
```

For light weight tag(V-1.0.0), no extra information available.
But for Annotated tag(V-1.1.0) extra information is avaialble.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git show V-1.0.0
commit 2a1c1f405542d19565a46481ec7f18f3a9c4cb86 (tag: V-1.0.0)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 26 21:46:58 2020 +0530

    fourth commit

diff --git a/file4.txt b/file4.txt
new file mode 100644
index 0000000..603cb1b
--- /dev/null
+++ b/file4.txt
@@ -0,0 +1 @@
+First Line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git show V-1.1.0
tag V-1.1.0
Tagger: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Wed Jul 29 20:43:43 2020 +0530

Release 1.1.0

commit f7ec10796b2c55f59dbfd1722e529127956025af (HEAD -> master, tag: V-1.1.0)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 26 21:51:08 2020 +0530

    fifth commit

diff --git a/file5.txt b/file5.txt
new file mode 100644
index 0000000..603cb1b
--- /dev/null
```

```
+++ b/file5.txt
@@ -0,0 +1 @@
+First Line
```

LightWeight tag is not implemented as object but Annotated tag implemented as object.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag -v V-1.0.0
error: V-1.0.0: cannot verify a non-tag object of type commit.

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag -v V-1.1.0
object f7ec10796b2c55f59dbfd1722e529127956025af
type commit
tag V-1.1.0
tagger Ravi <durgasoftonlinetraining@gmail.com> 1596035623 +0530

Release 1.1.0
error: no signature found
```

Annotated tag information will be stored in .git/refs/tags folder and .git/objects folder.

```
$ cat .git/refs/tags/V-1.1.0
378a767274b8e2991b6fcfa9d987fe6ec25fd34b
```

This is hash of tag object.

```
$ cat .git/objects/37/8a767274b8e2991b6fcfa9d987fe6ec25fd34b
x%░A
░0=░░░%ǐ      ░Ѓ░m░   ░6)░*░{+░░23BY[░SE'░░1a░░4Y░░░
                              %%░░ul░RD░░x\lg;░░'░]░uzm░░░░░ꜫT░░J-
5░░Be░░v░░3=:░░░`o░xf░Y░{_*░/░
```

To know the type of object, we have to use -t option with cat-file command.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git cat-file -t 378a767274b8e2991b6fcfa9d987fe6ec25fd34b
tag
```

To print the information of object, we have to use -p option with cat-file command.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git cat-file -p 378a767274b8e2991b6fcfa9d987fe6ec25fd34b
object f7ec10796b2c55f59dbfd1722e529127956025af
type commit
tag V-1.1.0
```

```
tagger Ravi <durgasoftonlinetraining@gmail.com> 1596035623 +0530

Release 1.1.0
```

Note: Extra information maintained by Annotated Tag when compared with Light Weight tag and hence it is recommended to use Annotated Tag.

Light Weight Tag vs Annotated Tag

Light Weight Tag	Annotated Tag
1) git tag <tagname>	1) git tag -a <tagname> -m <tagmessage>
2) It is just text label and not implemented as object	2) It is implemented as object
3) It is stored in .git/refs/tags folder	3) It is stored in both .git/refs/tags and .git/objects folders.
4) No extra information stored.	4) It stores tag message, tagger name, date of creation etc.

22.5) How to Tag a previous Commit?

Sometimes we may forgot to tag a particular commit. It is possible to tag for previous commits.

```
$ git tag -a <tagname> <commitid> -m <tagmessage>

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
f7ec107 (HEAD -> master, tag: V-1.1.0) fifth commit
2a1c1f4 (tag: V-1.0.0) fourth commit
b83ee22 third commit
ce4a690 second commit
121a2cb first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag -a V-1.0.0-beta ce4a690 -m '1.0.0 Beta Release'

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
f7ec107 (HEAD -> master, tag: V-1.1.0) fifth commit
2a1c1f4 (tag: V-1.0.0) fourth commit
b83ee22 third commit
ce4a690 (tag: V-1.0.0-beta) second commit
121a2cb first commit
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag --list
V-1.0.0
V-1.0.0-beta
V-1.1.0
```

22.6) How to update an existing Tag?

Sometimes we tag for a wrong commit. It is possible to update the tag to the correct commit.

We can do this by using the following 2 ways.

1) Delete the tag and Recreate tag with corresponding correct commit id.
2) By using -f or --force option to replace an existing tag without deleting.

```
$ git tag -a  <tagname> -f <newcommitid> -m <tagmessage>
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
f7ec107 (HEAD -> master, tag: V-1.1.0) fifth commit
2a1c1f4 (tag: V-1.0.0) fourth commit
b83ee22 third commit
ce4a690 (tag: V-1.0.0-beta) second commit
121a2cb first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag -a  V-1.0.0-beta -f b83ee22 -m 'Release 1.0.0 Beta Corrected'
Updated tag 'V-1.0.0-beta' (was 0466c7c)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
f7ec107 (HEAD -> master, tag: V-1.1.0) fifth commit
2a1c1f4 (tag: V-1.0.0) fourth commit
b83ee22 (tag: V-1.0.0-beta) third commit
ce4a690 second commit
121a2cb first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git show V-1.0.0-beta
tag V-1.0.0-beta
Tagger: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Thu Jul 30 20:28:35 2020 +0530
```

```
Release 1.0.0 Beta Corrected

commit b83ee2279d28490fcdc5f8cb3592b9014355c8cd (tag: V-1.0.0-beta)
Author: Ravi <durgasoftonlinetraining@gmail.com>
Date:   Sun Jul 26 21:46:43 2020 +0530

    third commit

diff --git a/file3.txt b/file3.txt
new file mode 100644
index 0000000..603cb1b
--- /dev/null
+++ b/file3.txt
@@ -0,0 +1 @@
+First Line
```

Note: For the same commit we can define multiple tags also based on our requirement.

```
$ git log --oneline
7e6aada (HEAD -> master, tag: V-1.0.0) fifth commit
3350c49 fourth commit
a927d06 (tag: V-0.9-beta) third commit
322d0ab second commit
009784d first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag V-0.9-b a927d06

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
7e6aada (HEAD -> master, tag: V-1.0.0) fifth commit
3350c49 fourth commit
a927d06 (tag: V-0.9-beta, tag: V-0.9-b) third commit
322d0ab second commit
009784d first commit
```

For the commit id a927d06, two tags are defined.

Q) Is it Possible to use same Tag for multiple Commits?

Not possible

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git log --oneline
7e6aada (HEAD -> master, tag: V-1.0.0) fifth commit
3350c49 fourth commit
```

```
a927d06 (tag: V-0.9-beta, tag: V-0.9-b) third commit
322d0ab second commit
009784d first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag V-1.0.0 322d0ab
fatal: tag 'V-1.0.0' already exists
```

22.7) How to compare Tags:

Tags are nothing but just references to our commits. Hence we can compare two tags.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git diff V-1.0.0 V-1.1.0
diff --git a/file5.txt b/file5.txt
new file mode 100644
index 0000000..603cb1b
--- /dev/null
+++ b/file5.txt
@@ -0,0 +1 @@
+First Line
```

Note: **We can use difftool command also.**

```
$ git difftool V-0.9-beta V-0.9-b
```

22.8) How to Push Tags to remote Repository?

Bydefault push command wont push tags to the remote repository. We have to push tags separately.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git remote add origin https://github.com/durgadevops777/tagging.git

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git push origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 967 bytes | 241.00 KiB/s, done.
Total 11 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/durgadevops777/tagging.git
 * [new branch]      master -> master
```

Whenever we are using push command, only code pushed to the remote repository but not tags.

tag-2.JPG

Observe releases link in github repository page.

How to push a Single Tag?

git push origin <tagname>

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git push origin V-1.0.0-beta
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 187 bytes | 187.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/durgadevops777/tagging.git
 * [new tag]       V-1.0.0-beta -> V-1.0.0-beta
```

Observe releases link in github repository page, you can see this tag.
From there, you can download complete source code upto that release.

How to push all Tags?

We have to use --tags option with git push command.
$ git push origin master --tags

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git push origin master --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 171 bytes | 171.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/durgadevops777/tagging.git
 * [new tag]       V-1.0.0 -> V-1.0.0
 * [new tag]       V-1.1.0 -> V-1.1.0
```

How to delete a Tag from the Remote Repository?

git push origin :V-1.0.0-beta

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git push origin :V-1.0.0-beta
To https://github.com/durgadevops777/tagging.git
 - [deleted]        V-1.0.0-beta
```

Now, the tag V-1.0.0-beta deleted from the remote repository, but not from local repository.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/tagging (master)
$ git tag --list
V-1.0.0
V-1.0.0-beta
V-1.1.0
```

Topic-23

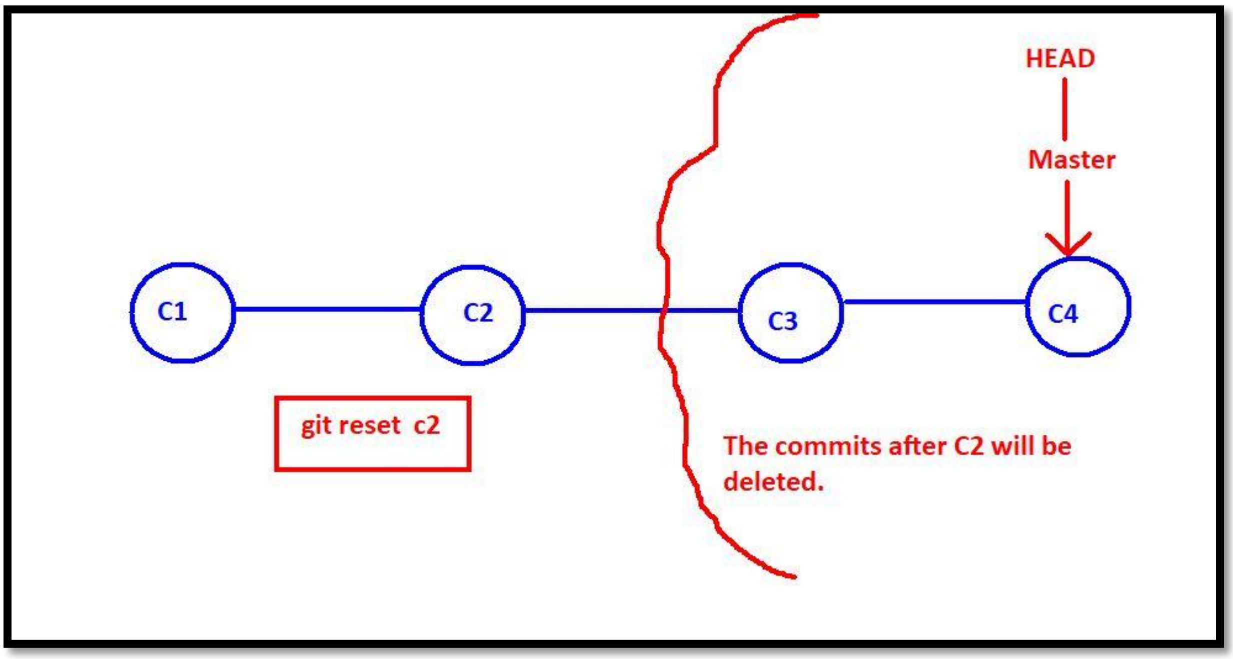
git revert Command

Topic-23: git revert Command

23.1) Need of revert Command:

In the last videos we covered already git reset command.

$ git reset c2
After c2, all the next commits will be deleted.



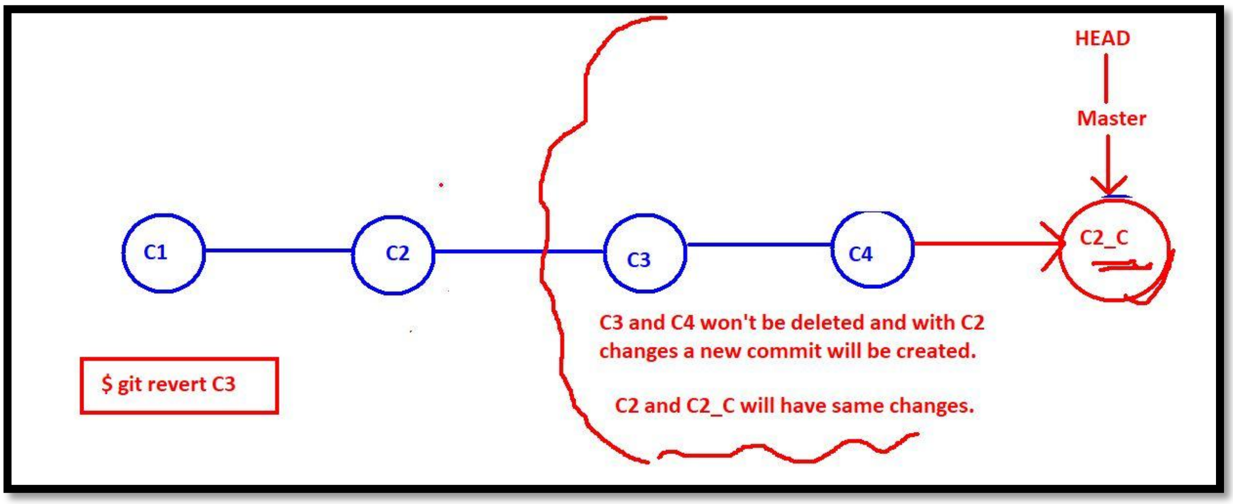
Suppose if this code already pushed to the remote repository and already several people pulled this code and starts working on that code,git reset command may create big problem.

git reset command deletes commit history and hence it is destructive command and not recommended to use on public repositories.

To overcome this problem we have to go for git revert command.
git revert command won't delete commit history. It reverts the required commit by creating a new commit. i.e it will undo a particular commit without deleting commit history.

```
$ git revert C3
```

git will revert the changes of C3, means with state of C2, a new commit will be created.


HEAD
Master
C1
C2
C3
C4
C2_C
$ git revert C3
C3 and C4 won't be deleted and with C2 changes a new commit will be created.
C2 and C2_C will have same changes.

In git revert, commit history won't be deleted and hence there is no effect on peer developers. It is safe operation.

23.2) Demo Example to revert last commit:

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ mkdir revertdemo

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ cd revertdemo

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo
$ git init
Initialized empty Git repository in D:/gitprojects/revertdemo/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ echo 'First Line' > file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git add .;git commit -m 'first commit'
[master (root-commit) 794f9c4] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ echo 'Second Line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
```

```
$ git  commit -am 'second commit'
[master 44cec33] second commit
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ echo 'Third Line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git  commit -am 'third commit'

[master f6f93c8] third commit
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ echo 'Fourth Line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git  commit -am 'fourth commit'
[master b2abe3a] fourth commit
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ cat file1.txt
First Line
Second Line
Third Line
Fourth Line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git log --oneline
b2abe3a (HEAD -> master) fourth commit
f6f93c8 third commit
44cec33 second commit
794f9c4 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git revert b2abe3a
[master a295e3d] Revert "fourth commit"
 1 file changed, 1 deletion(-)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git log --oneline
a295e3d (HEAD -> master) Revert "fourth commit"
b2abe3a fourth commit
f6f93c8 third commit
```

```
44cec33 second commit
794f9c4 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ cat file1.txt
First Line
Second Line
Third Line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git diff f6f93c8 a295e3d
```

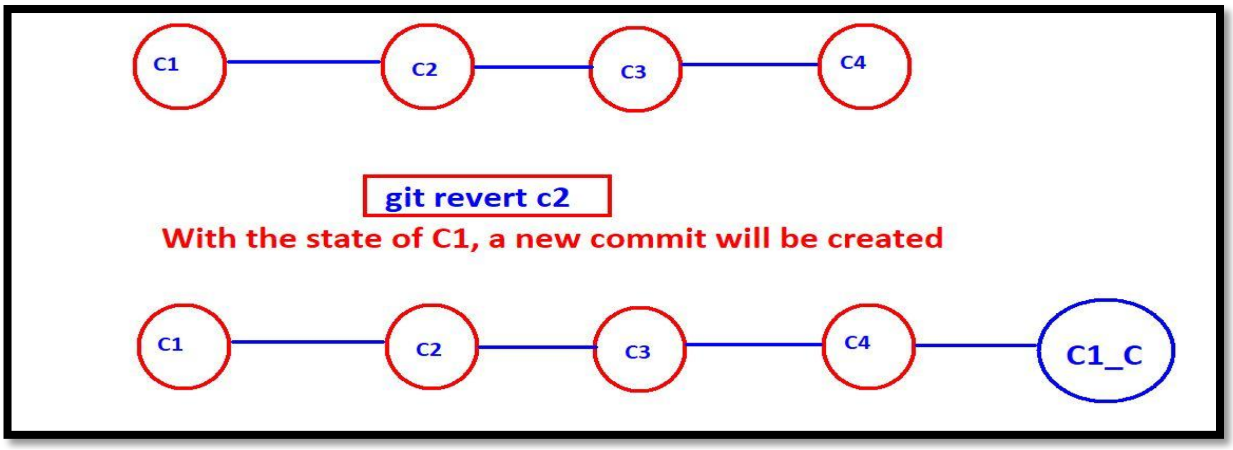
There is no difference because both commits are showing same state.

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo (master)
$ git diff b2abe3a a295e3d
diff --git a/file1.txt b/file1.txt
index 79cdfbb..b0d816c 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1,4 +1,3 @@
 First Line
 Second Line
 Third Line
-Fourth Line
```

There is difference in the last commit and last but one commit.

Note: We can revert any commit, need not be last commit.

23.3) Demo Example to revert a Particular Commit



```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ mkdir revertdemo2

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ cd revertdemo2

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2
$ git init
Initialized empty Git repository in D:/gitprojects/revertdemo2/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ echo 'First Line' > file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git add .;git commit -m 'first commit'
[master (root-commit) 2d87928] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ echo 'Second Line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git commit -am 'second commit'
[master 6fa1d8e] second commit
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ echo 'Third Line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git commit -am 'third commit'
[master 3589136] third commit
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ echo 'Fourth Line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git commit -am 'fourth commit'
[master 85f0294] fourth commit
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ cat file1.txt
```

```
First Line
Second Line
Third Line
Fourth Line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git log --oneline
85f0294 (HEAD -> master) fourth commit
3589136 third commit
6fa1d8e second commit
2d87928 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git revert 6fa1d8e
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
error: could not revert 6fa1d8e... second commit
hint: after resolving the conflicts, mark the corrected paths
hint: with 'git add <paths>' or 'git rm <paths>'
hint: and commit the result with 'git commit'

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master|REVERTING)
$ vi file1.txt
```

We have to edit the file to resolve merge conflict

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master|REVERTING)
$ git status
On branch master
You are currently reverting commit 6fa1d8e.
  (fix conflicts and run "git revert --continue")
  (use "git revert --skip" to skip this patch)
  (use "git revert --abort" to cancel the revert operation)

Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master|REVERTING)
$ git add .;git commit -m 'second commit reverted'
[master f37578f] second commit reverted
 1 file changed, 3 deletions(-)
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ cat file1.txt
First Line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/revertdemo2 (master)
$ git log --oneline
f37578f (HEAD -> master) second commit reverted
85f0294 fourth commit
3589136 third commit
6fa1d8e second commit
2d87928 first commit
```

Note:

With git revert we can use -n option also.
-n or --no-commit
Usually the git revert command automatically creates some commits with commit log messages stating which commits were reverted. This flag applies the changes necessary to revert the named commits to your working tree and the index, but does not make the commits. In addition, when this option is used, your index does not have to match the HEAD commit. The revert is done against the beginning state of your index.

This is useful when reverting more than one commits' effect to your index in a row.

```
file1.txt
git add file1.txt
file2.txt
git add file2.txt
file3.txt
git add file3.txt
git commit '3 files modified'
```

Topic-24

Cherry-Picking

(git cherry-pick Command)

Topic-24: Cherry-Picking (git cherry-pick Command)

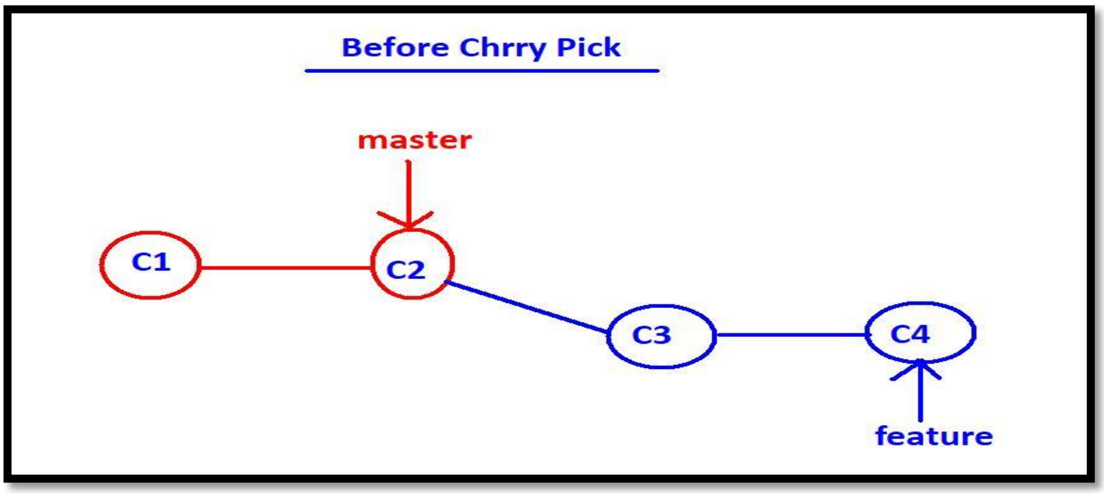
24.1) Need of Cherry-Picking:

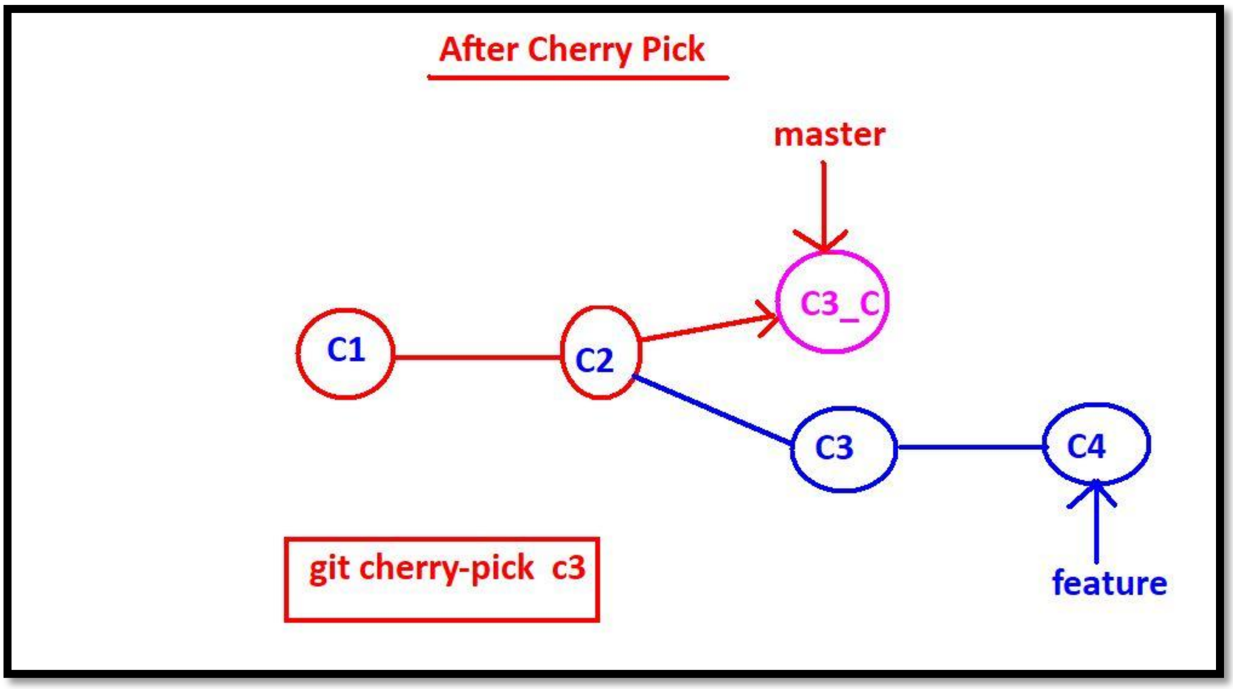
Assume that we have two branches are there like master and feature. If we merge feature branch to the master branch, then all commits and the corresponding changes will be added to master branch. Insted of all commits, we can pick an arbitraty commit of the feature branch and we can append that commit to the master branch. It is possible by using cherry-pick command.

24.2) Use cases of cherry-pick:

1. Cherry-pick allows to share code between branches.
2. We can use for bug hot fixes.
Assume that a developer has started working on a new feature. During that new feature development he identify a pre-existing bug. The developer creates an explicit commit patching this bug. This new patch commit can be cherry-picked directly to the master branch to fix the bug before it effects more users.

24.3) Demo Example for cherry-picking





```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ mkdir cherrypick

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects
$ cd cherrypick/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick
$ git init
Initialized empty Git repository in D:/gitprojects/cherrypick/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ echo 'First line' > file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git add .;git commit -m 'first commit'
[master (root-commit) 3bbcfa0] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ echo 'Second line' >> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git add .;git commit -m 'second commit'
[master c1d9b78] second commit
 1 file changed, 1 insertion(+)
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git log --oneline
c1d9b78 (HEAD -> master) second commit
3bbcfa0 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ cat file1.txt
First line
Second line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git checkout -b feature
Switched to a new branch 'feature'

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ ls
file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ git log --oneline
c1d9b78 (HEAD -> feature, master) second commit
3bbcfa0 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ echo 'Third Line by feature branch'>> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ git commit -am 'third commit by feature'
[feature 2ca8bd2] third commit by feature
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ echo 'Fourth Line by feature branch'>> file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ git commit -am 'fourth commit by feature'
[feature 3b8f8b8] fourth commit by feature
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ cat file1.txt
First line
Second line
Third Line by feature branch
Fourth Line by feature branch
```

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ git log --oneline
3b8f8b8 (HEAD -> feature) fourth commit by feature
2ca8bd2 third commit by feature
c1d9b78 (master) second commit
3bbcfa0 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (feature)
$ git checkout master
Switched to branch 'master'

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ cat file1.txt
First line
Second line

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git log --oneline
c1d9b78 (HEAD -> master) second commit
3bbcfa0 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git cherry-pick 2ca8bd2
[master cceff21] third commit by feature
 Date: Tue Aug 4 20:54:40 2020 +0530
 1 file changed, 1 insertion(+)

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ git log --oneline
cceff21 (HEAD -> master) third commit by feature
c1d9b78 second commit
3bbcfa0 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/cherrypick (master)
$ cat file1.txt
First line
Second line
Third Line by feature branch
```

<u>Note:</u>

1. When every we are using cherry-pick command, a brand new commit object will be created but with same changes of the picked commit of feature branch.
2. Cherry-pick may cause duplicate commits and hence in most of the scenarios we can use merge operation instead of cherry-pick.

3. We can use -n or --no-commit option with cherry-pick, in that case new commit won't be created. This is helpful whenever we are doing cherry-picking multiple commits. With all these changes we can create a single commit.

```
$ git checkout master
$ git cherry-pick -n commitid1
$ git cherry-pick -n commitid2
$ git cherry-pick -n commitid3
$ git add .;git commit -m 'single commit for multiple cherry-pick options'
```

Topic-25

git reflog Command

Topic-25: git reflog Command

reflog → means reference log
We can use git reflog command to display all git operations what ever we performed on local repository.

We can use git reflog command to know only local repository operations but not remote repository operations.

Syntax:

$ git reflog
It will show all git operations performed on local repository.

$ git reflog <branch_name>
It will show all git operations performed on local repository related to that particular branch.

Note: reflog will maintain 90 days git operations history bydefault.
We can use reflog command to go to specific state of the repository.

By mistake if we did any unwanted destruction operation(like git reset --hard), still we can recover by using git reflog command.

Demo Example:

```
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo
$ git init
Initialized empty Git repository in D:/gitprojects/reflogdemo/.git/

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ echo 'First line' > file1.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git add .; git commit -m 'first commit'
[master (root-commit) cdd9f31] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ echo 'First line' > file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git add .; git commit -m 'second commit'
```

```
[master 9f6864b] second commit
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ echo 'First line' > file3.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git add .; git commit -m 'third commit'
[master 8c37a9b] third commit
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git reflog
8c37a9b (HEAD -> master) HEAD@{0}: commit: third commit
9f6864b HEAD@{1}: commit: second commit
cdd9f31 HEAD@{2}: commit (initial): first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git checkout -b feature
Switched to a new branch 'feature'

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ echo 'first line' > a.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ git add .;git commit -m 'first commit by feature'
[feature 051e1be] first commit by feature
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ echo 'first line' > b.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ git add .;git commit -m 'second commit by feature'
[feature b5d2b02] second commit by feature
 1 file changed, 1 insertion(+)
 create mode 100644 b.txt
lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ git reflog
b5d2b02 (HEAD -> feature) HEAD@{0}: commit: second commit by feature
051e1be HEAD@{1}: commit: first commit by feature
8c37a9b (master) HEAD@{2}: checkout: moving from master to feature
```

```
8c37a9b (master) HEAD@{3}: commit: third commit
9f6864b HEAD@{4}: commit: second commit
cdd9f31 HEAD@{5}: commit (initial): first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ git reflog master
8c37a9b (master) master@{0}: commit: third commit
9f6864b master@{1}: commit: second commit
cdd9f31 master@{2}: commit (initial): first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ git reflog feature
b5d2b02 (HEAD -> feature) feature@{0}: commit: second commit by feature
051e1be feature@{1}: commit: first commit by feature
8c37a9b (master) feature@{2}: branch: Created from HEAD

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (feature)
$ git checkout master
Switched to branch 'master'

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git reset --hard cdd9f31
HEAD is now at cdd9f31 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git log --oneline
cdd9f31 (HEAD -> master) first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git reflog
cdd9f31 (HEAD -> master) HEAD@{0}: reset: moving to cdd9f31
8c37a9b HEAD@{1}: checkout: moving from feature to master
b5d2b02 (feature) HEAD@{2}: commit: second commit by feature
051e1be HEAD@{3}: commit: first commit by feature
8c37a9b HEAD@{4}: checkout: moving from master to feature
8c37a9b HEAD@{5}: commit: third commit
9f6864b HEAD@{6}: commit: second commit
cdd9f31 (HEAD -> master) HEAD@{7}: commit (initial): first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git reset --hard 8c37a9b
HEAD is now at 8c37a9b third commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git log --oneline
```

```
8c37a9b (HEAD -> master) third commit
9f6864b second commit
cdd9f31 first commit

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ ls
file1.txt  file2.txt  file3.txt

lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/reflogdemo (master)
$ git reflog --stat
8c37a9b (HEAD -> master) HEAD@{0}: reset: moving to 8c37a9b
 file3.txt | 1 +
 1 file changed, 1 insertion(+)
cdd9f31 HEAD@{1}: reset: moving to cdd9f31
 file1.txt | 1 +
 1 file changed, 1 insertion(+)
8c37a9b (HEAD -> master) HEAD@{2}: checkout: moving from feature to master
 file3.txt | 1 +
 1 file changed, 1 insertion(+)
b5d2b02 (feature) HEAD@{3}: commit: second commit by feature
 b.txt | 1 +
 1 file changed, 1 insertion(+)
051e1be HEAD@{4}: commit: first commit by feature
 a.txt | 1 +
 1 file changed, 1 insertion(+)
8c37a9b (HEAD -> master) HEAD@{5}: checkout: moving from master to feature
 file3.txt | 1 +
 1 file changed, 1 insertion(+)
8c37a9b (HEAD -> master) HEAD@{6}: commit: third commit
 file3.txt | 1 +
 1 file changed, 1 insertion(+)
9f6864b HEAD@{7}: commit: second commit
 file2.txt | 1 +
 1 file changed, 1 insertion(+)
cdd9f31 HEAD@{8}: commit (initial): first commit
 file1.txt | 1 +
 1 file changed, 1 insertion(+)

$ git reflog --oneline
8c37a9b (HEAD -> master) HEAD@{0}: reset: moving to 8c37a9b
cdd9f31 HEAD@{1}: reset: moving to cdd9f31
8c37a9b (HEAD -> master) HEAD@{2}: checkout: moving from feature to master
b5d2b02 (feature) HEAD@{3}: commit: second commit by feature
051e1be HEAD@{4}: commit: first commit by feature
8c37a9b (HEAD -> master) HEAD@{5}: checkout: moving from master to feature
```

```
8c37a9b (HEAD -> master) HEAD@{6}: commit: third commit
9f6864b HEAD@{7}: commit: second commit
cdd9f31 HEAD@{8}: commit (initial): first commit
```

Note: Whatever options we can use with log command, all those options can be used for reflog also.

Q) By Mistake if we did git reset --hard Command and some Files got removed from Local Repository. Is it possible to recover those changes OR not?

Yes. For this purpose we can use git reflog command.