## Understanding Git

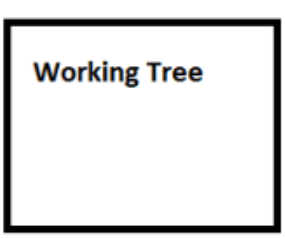
- To understand Git, we need to understand 5 areas of git
- Lets start from folder level
- Create an empty directory

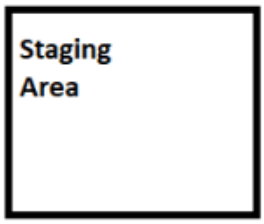
```
mkdir learning
```

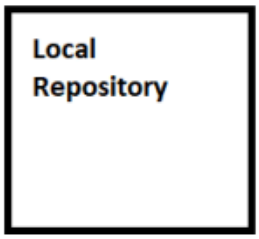
- Lets make this folder intelligent with Version Control,

```
cd learning
git init
```

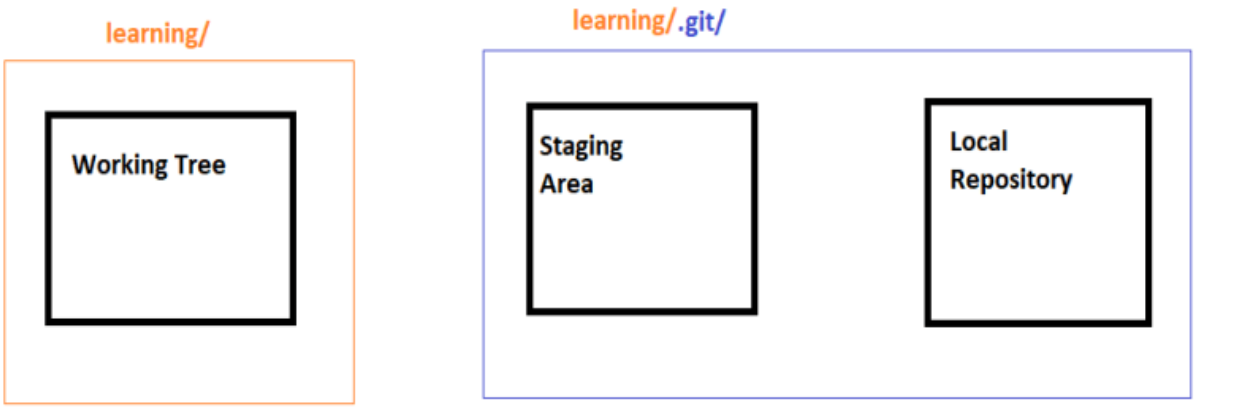
- Lets start our journey with three areas of git
  - Working Tree/Working Directory
  - Staging Area
  - Local Repository
- BY adding git init we have made learning folder as repository




- If you want physical locations according to above example
-



- Now make some changes and execute the following commands

```
touch  readme.txt
```

- Now the `git status` command will describe the changes to the developer

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        readme.txt

nothing added to commit but untracked files present (use "git add" to track)
```

- The changes are in working tree and we need to move the changes from working tree to staging, This operation is called as *add* (We are adding changes)

```
git add --help
# add all the changes in current directory
git add .
git status
```

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add .

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   readme.txt


QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- Now we need to move the changes from staging to local repository, bcoz local repository has all the feature which we

need in Version control System. This operation is as commit and to perform this operation, git should know your username and your email id

```
git config --global user.name "your git hub username"
git config --global user.email "your github email"
```

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git config --global user.name "qtdevops"

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git config --global user.email "qtdevops@gmail.com"
```

- Now lets add changes from staging area to local repository

```
git commit -m "This is my first commit"
git status
git log
```

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git commit -m "This is my first commit"
[master (root-commit) 2ec768d] This is my first commit
 1 file changed, 4 insertions(+)
 create mode 100644 readme.txt

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
nothing to commit, working tree clean

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git log
commit 2ec768d70494295aa4e0109c9797fb043617a556 (HEAD -> master)
Author: qtdevops <qtdevops@gmail.com>
Date:   Tue Apr 28 08:18:14 2020 +0530

    This is my first commit

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- Now let this developer add one more change, add some text to existing file and create a new

file

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        src/

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

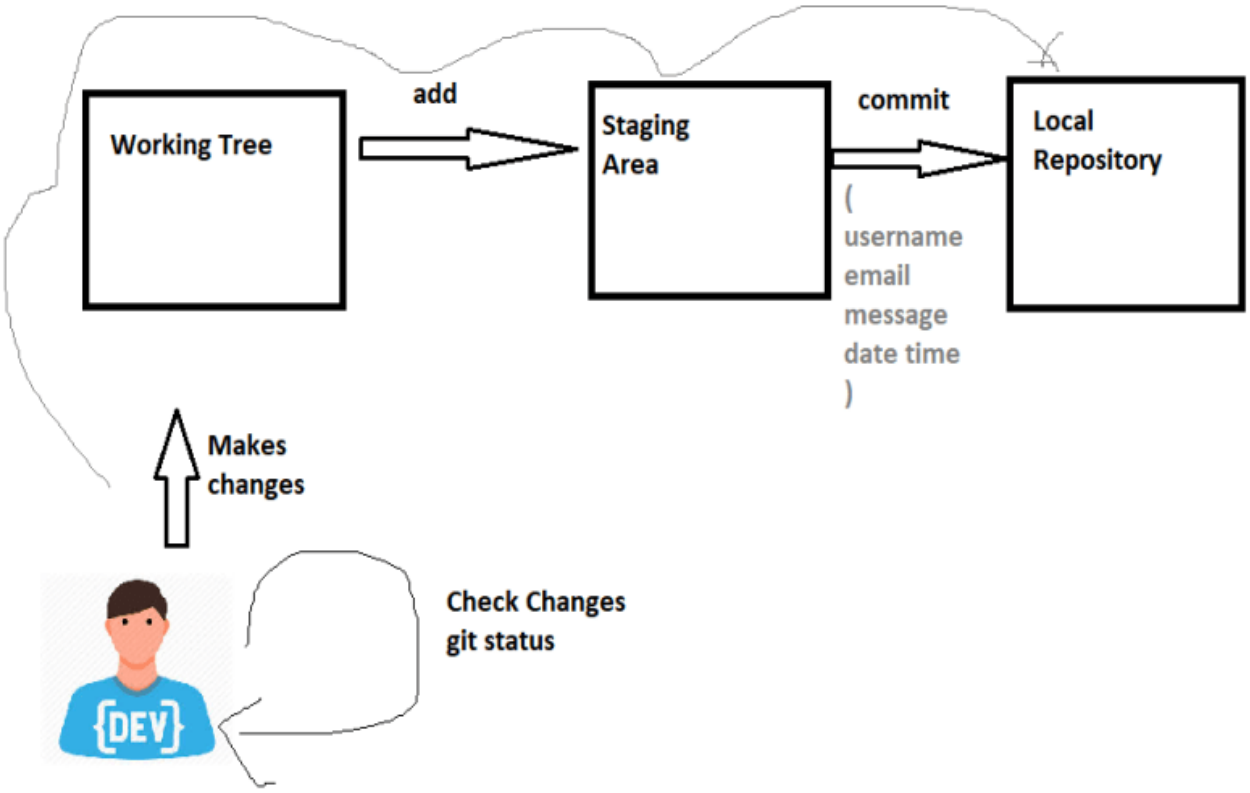
- Add these changes to staging area and then local repo

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add .

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt
        new file:   src/main.py


QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git commit -m "this is my second commit"
[master 8f0e6ea] this is my second commit
 2 files changed, 3 insertions(+)
 create mode 100644 src/main.py

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git log
commit 8f0e6eaaa1e70ddc2c0307139f3f0428f6059a59 (HEAD -> master)
Author: qtdevops <qtdevops@gmail.com>
Date:   Tue Apr 28 08:25:49 2020 +0530

    this is my second commit

commit 2ec768d70494295aa4e0109c9797fb043617a556
Author: qtdevops <qtdevops@gmail.com>
```

- Terms:
    1. Untracked: Is a file which was never part of local-repository. Newly added files are generally untracked

2. Modified: Making changes to existing file in local repository
3. Color Significance:
    - Red => Working Tree
    - Green => Added to staging area

- Workflow:




- Dont Remember git commands use cheatsheets [Refer Here]

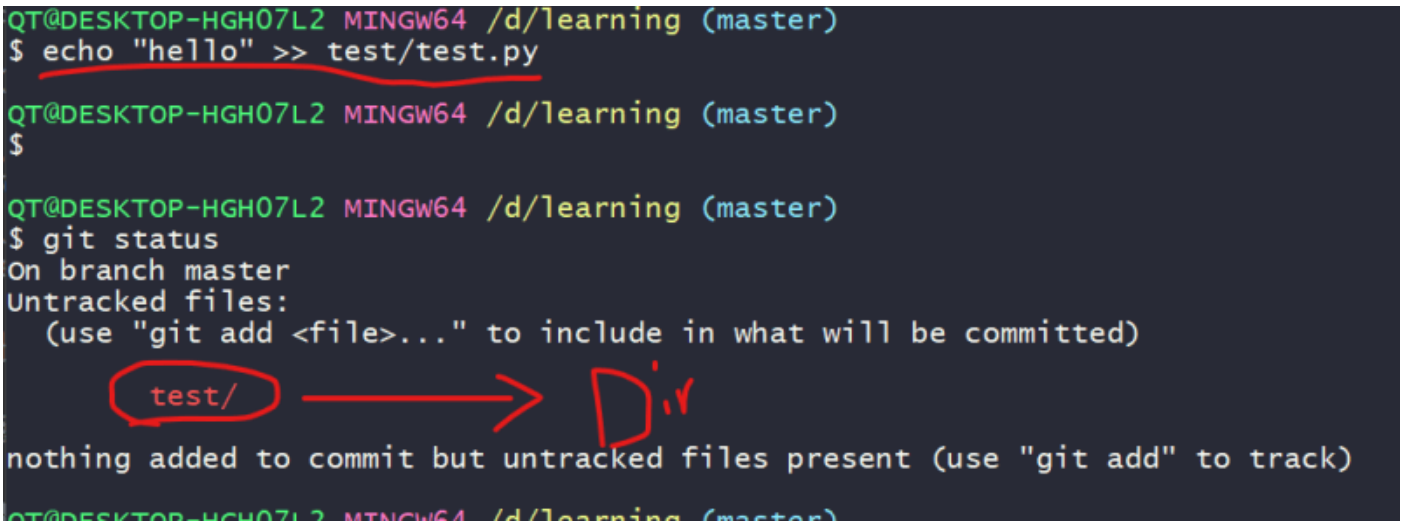
- Lets try adding an empty folder as a change

```
mkdir test
git status
```

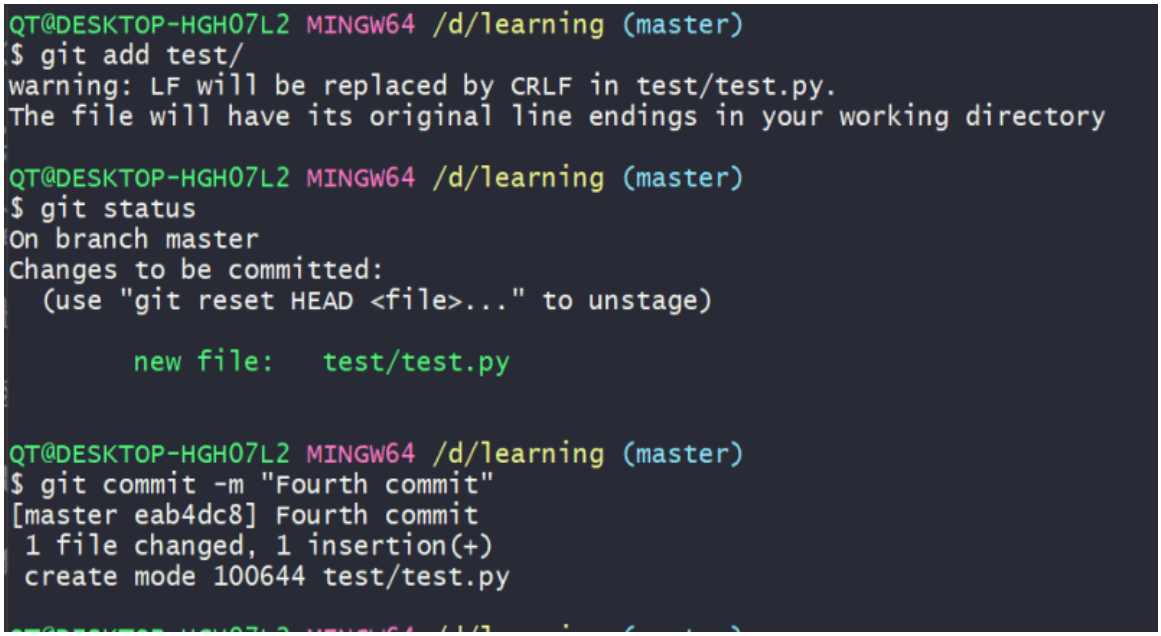
```
OT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
nothing to commit, working tree clean

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- Git will identify only files not folders. so lets add some file to test directory

```
echo "hello" >> test/test.py
```




- Add changes to the Staging Area and then commit to local repo
-




- Add all changes from Working tree to staging

```
git add --all
```

- Add only modified changes from working tree to staging area

```
git add -u
```

- Recomendation: `git add --help`

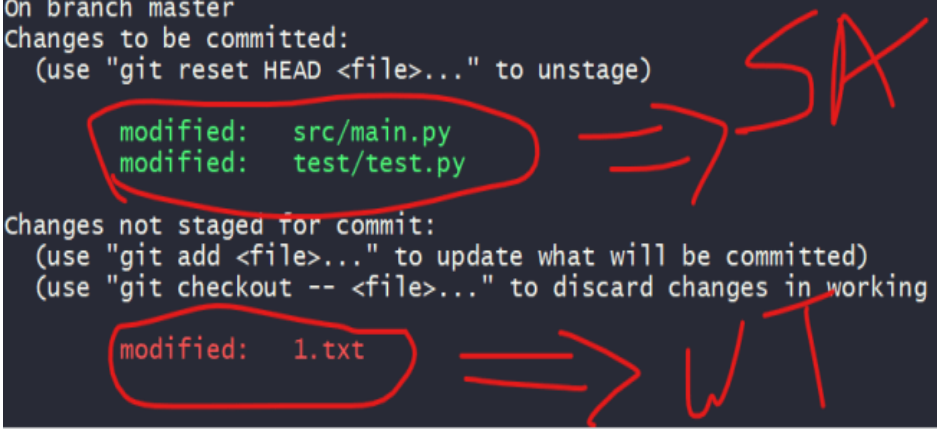
- Multiple adds from working area to staging area and commit them

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   1.txt
        modified:   src/main.py
        modified:   test/test.py

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

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add src/main.py

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/main.py

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

        modified:   1.txt
        modified:   test/test.py
```

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add test/test.py
warning: LF will be replaced by CRLF in test/test.py.
The file will have its original line endings in your working directory

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/main.py
        modified:   test/test.py

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

        modified:   1.txt
```

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add test/test.py
warning: LF will be replaced by CRLF in test/test.py.
The file will have its original line endings in your working directory

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/main.py
        modified:   test/test.py

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

        modified:   1.txt
```




```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git commit -m "fifth commit"
[master 627721c] fifth commit
 2 files changed, 3 insertions(+), 2 deletions(-)

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   1.txt

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

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- 
- Lets Travel Back to older changes (Versions)
  - Whenever you commit the changes to local repo, a commit-id gets created. commit Id has two versions long and short

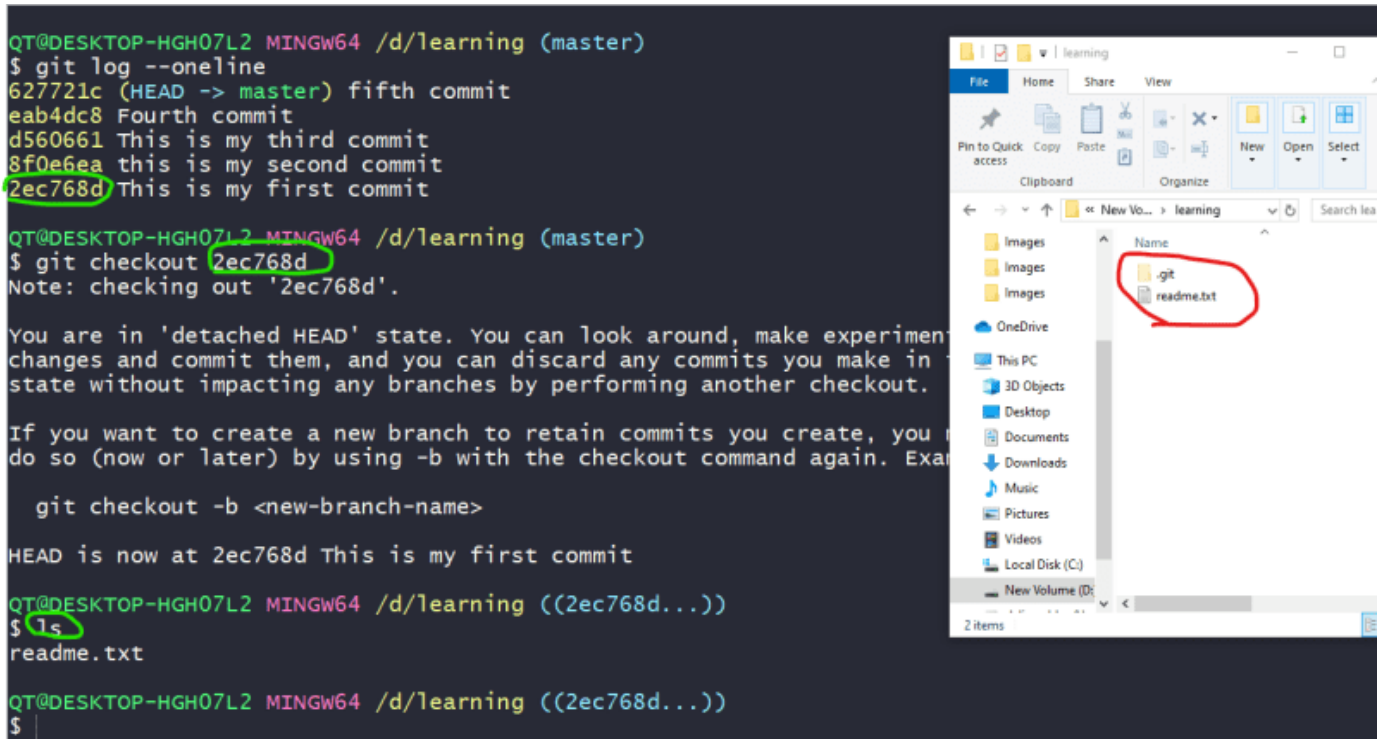
- Short Version

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git log --oneline
627721c (HEAD -> master) fifth commit
eab4dc8 Fourth commit
d560661 This is my third commit
8f0e6ea this is my second commit
2ec768d This is my first commit

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

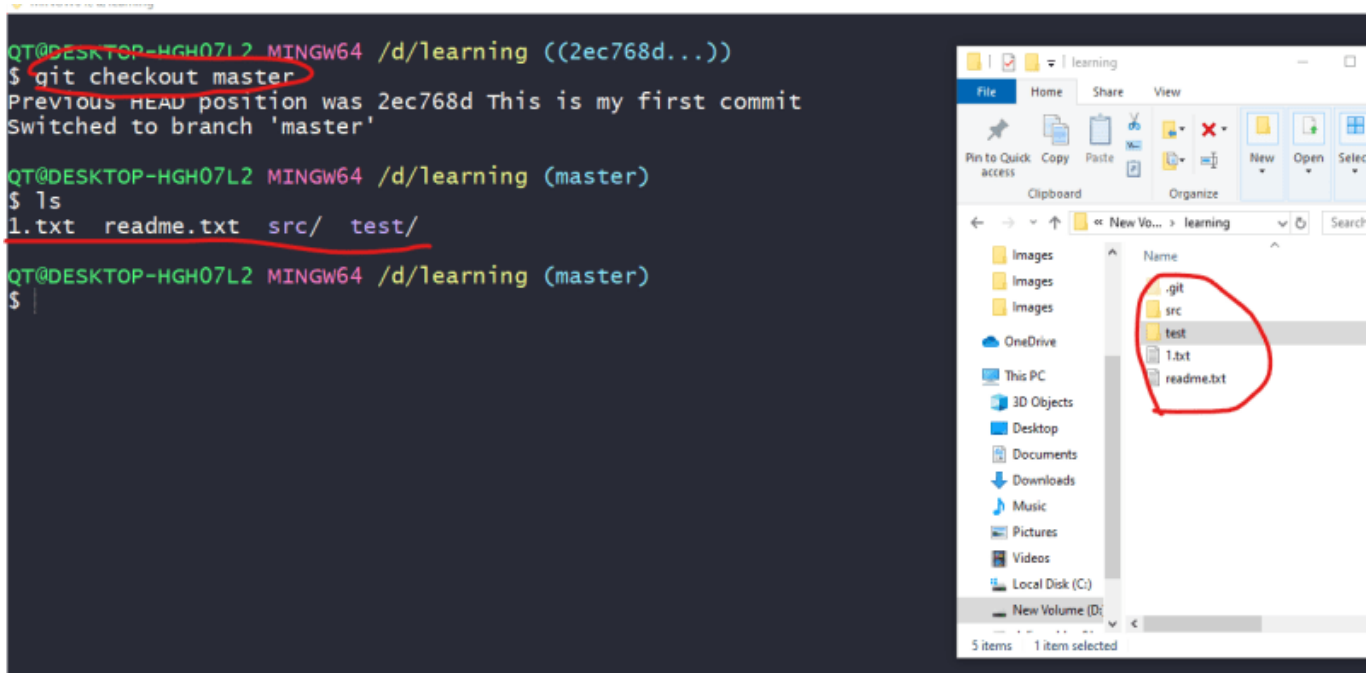
- Long Version

```
commit 627721ca3d791cf41710f5baebe1018fa19cd86f (HEAD -> master)
Author: qtdevops <qtdevops@gmail.com>
Date:   Wed Apr 29 08:19:11 2020 +0530

    fifth commit

commit eab4dc893bf4b7b49249844dae8f40339c7e0f65
Author: qtdevops <qtdevops@gmail.com>
Date:   Wed Apr 29 08:02:17 2020 +0530

    Fourth commit

commit d5606610e73a847d758d21d101a67353f6930493
Author: qtdevops <qtdevops@gmail.com>
Date:   Wed Apr 29 07:55:23 2020 +0530

    This is my third commit

commit 8f0e6eaaa1e70ddc2c0307139f3f0428f6059a59
Author: qtdevops <qtdevops@gmail.com>
Date:   Tue Apr 28 08:25:49 2020 +0530

    this is my second commit

commit 2ec768d70494295aa4e0109c9797fb043617a556
Author: qtdevops <qtdevops@gmail.com>
Date:   Tue Apr 28 08:18:14 2020 +0530

    This is my first commit
:
```

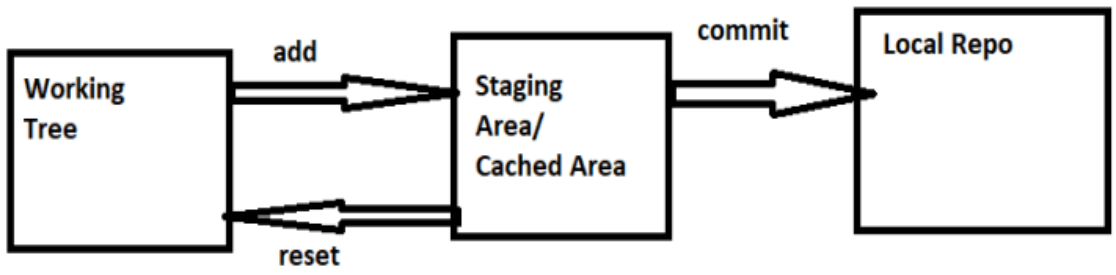
- Let me go to the first commit




- To comeback to latest version execute `git checkout master`



- **Git Areas**
  **Update**




- Is there Any way to revert the changes
  - From Working Tree:

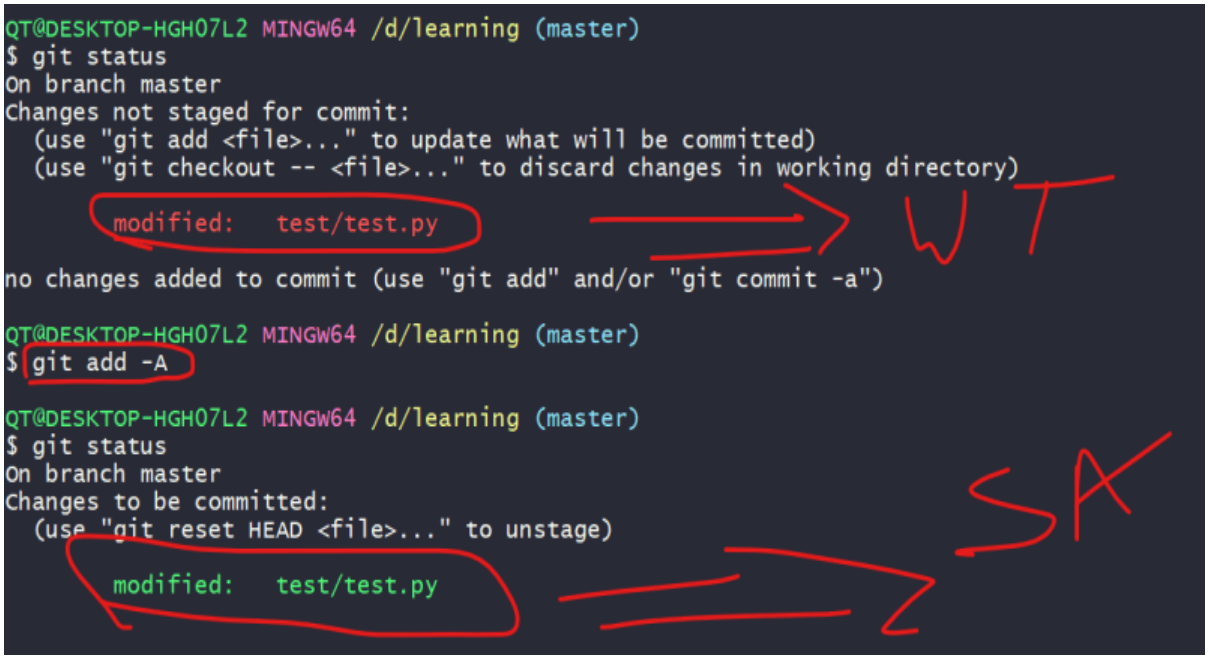
```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme.txt

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

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git checkout -- readme.txt

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
nothing to commit, working tree clean

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

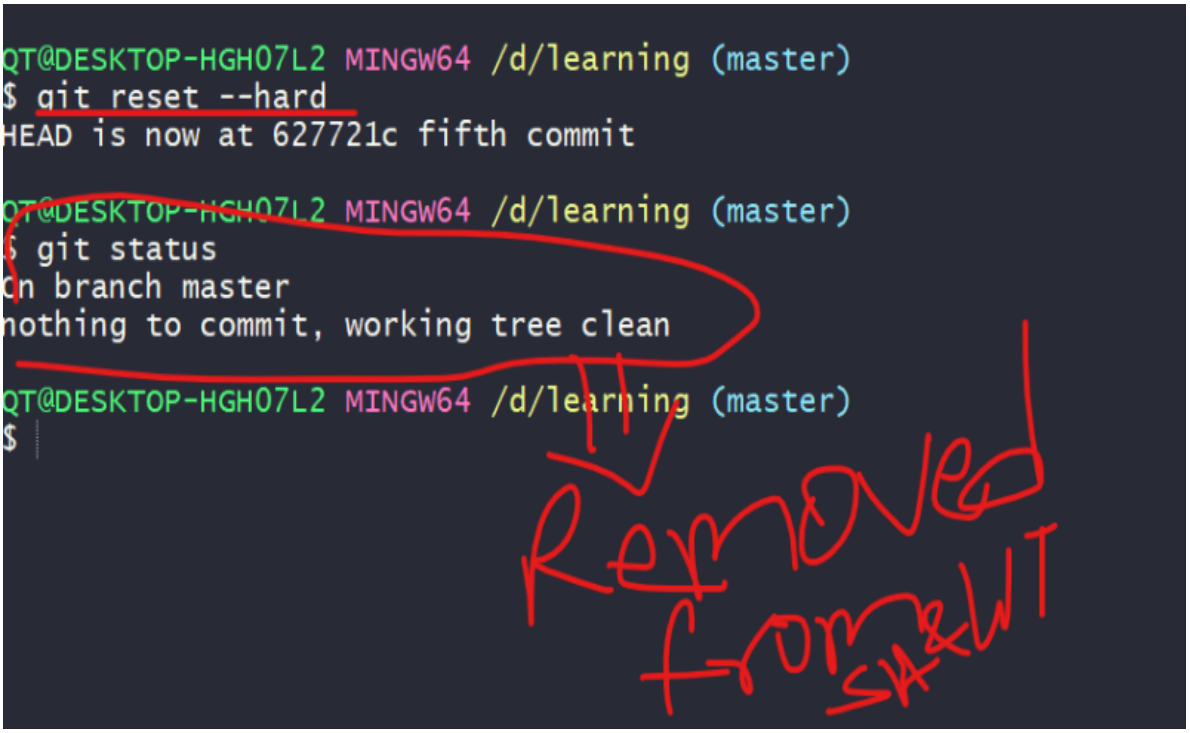
```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ ls
1.txt  readme.txt  src/  test/

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme.txt

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

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add readme.txt

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt


QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- From Staging Area To Working Tree

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt


QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git reset readme.txt
Unstaged changes after reset:
M       readme.txt

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme.txt

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

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- From staging area:




- Exercise: There is also a command `git reset --soft` findout what it does

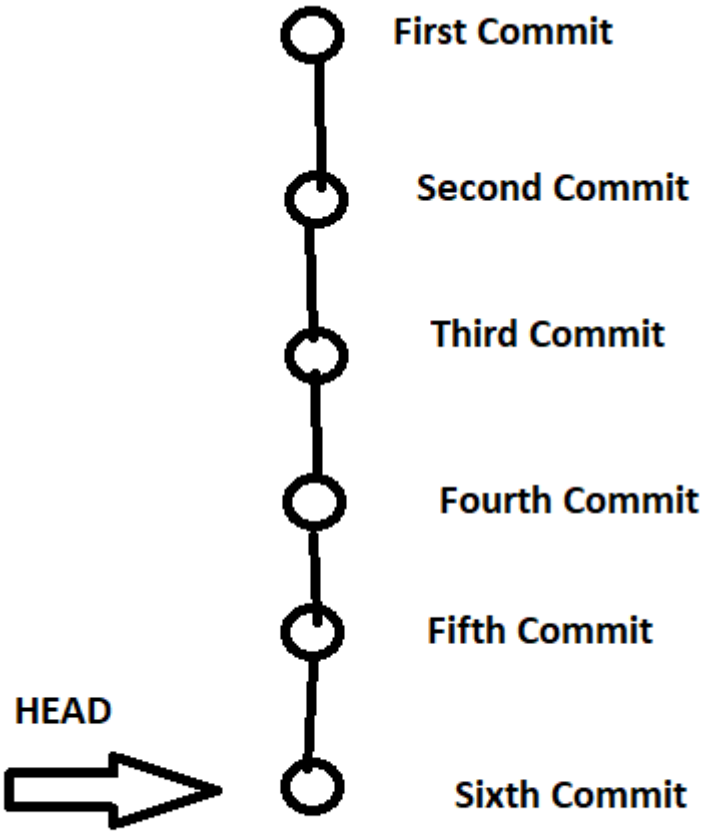
- How to deal with deleting the files

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ rm 1.txt

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    1.txt

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

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git add 1.txt

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    1.txt


QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

- History so far

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git log --oneline
f3d26f3 (HEAD -> master) Sixth Commit
627721c fifth commit
eab4dc8 Fourth commit
d560661 This is my third commit
8f0e6ea this is my second commit
2ec768d This is my first commit
```

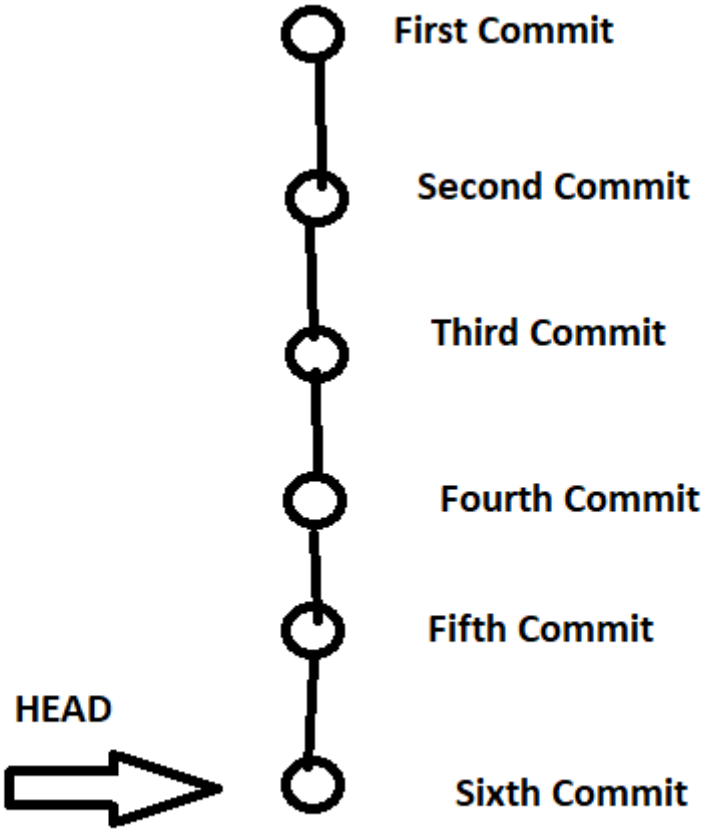
- Concept of




  Head

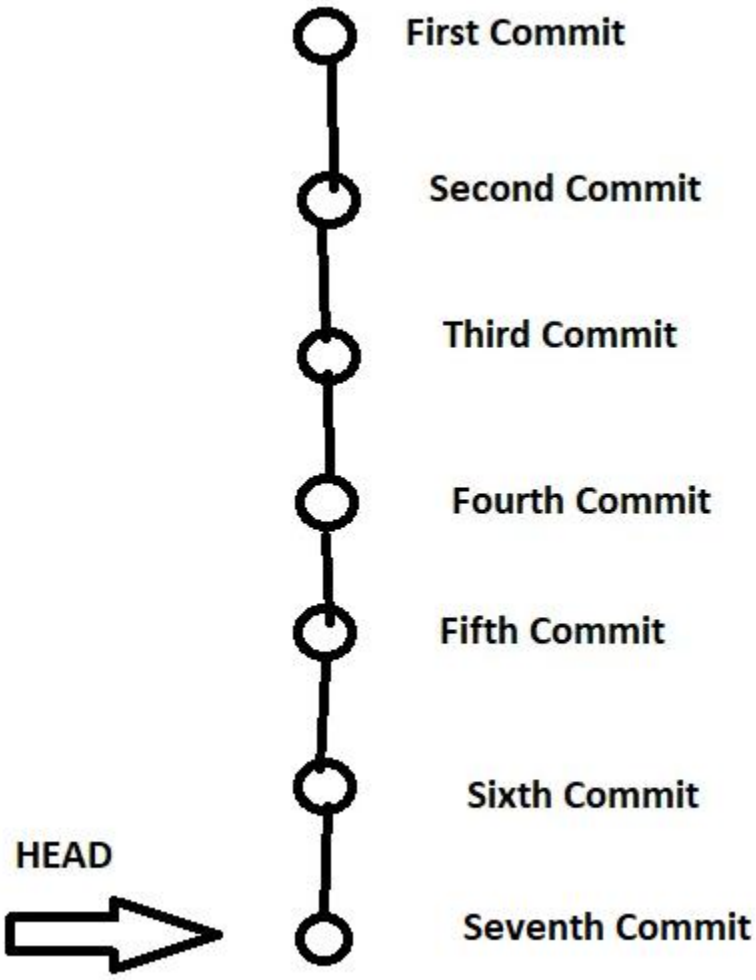
## Git Conceptual Understanding

- Lets make 7th




commit

- Now along with 7 commit the HEAD pointer moves to the seventh commit, so by default HEAD is looking always at a latest

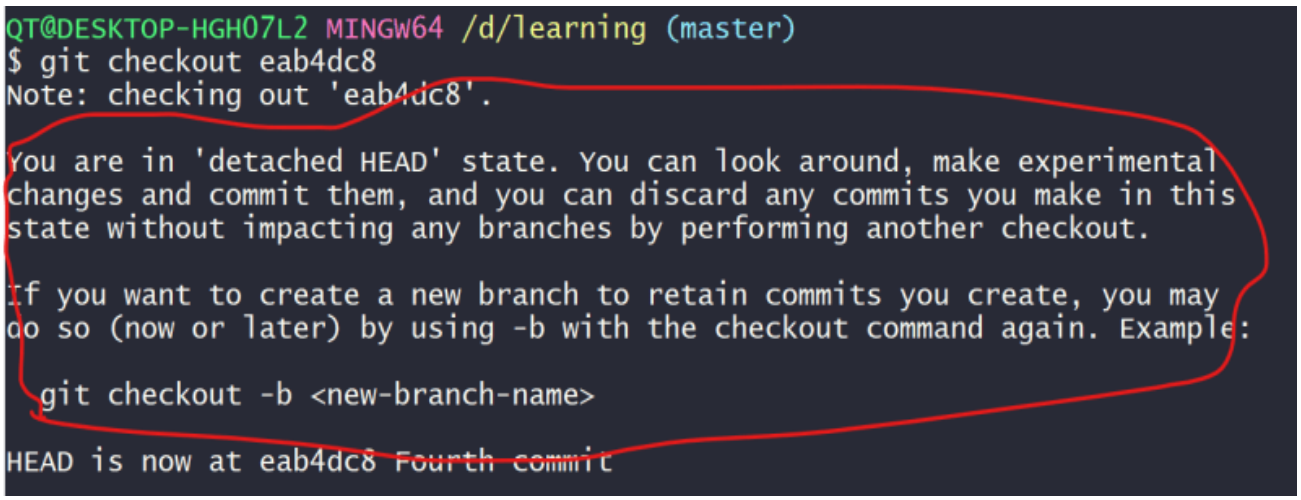



version.

- To Remove untracked files git uses `git clean`

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git clean --help

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git clean -fd .
Removing New Text Document.txt
```

- Lets Travel back in history to fourth commit

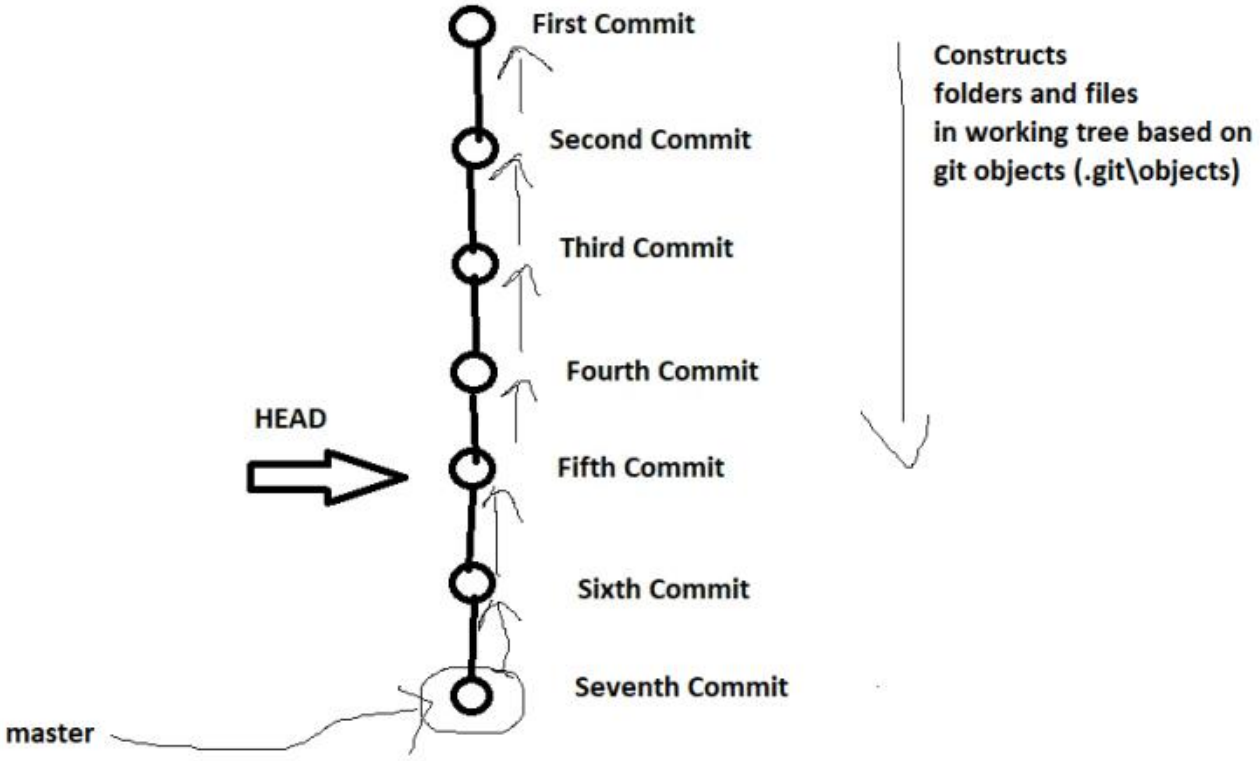



- Whenever you move back in history by commit id, you will get an error/warning called as DETACHED_HEAD
- Local => git repo => .git\config
- Global => ~.gitconfig
- 
- System => <install-dir>/etc/gitconfig

## Git Branches and How Git Works:

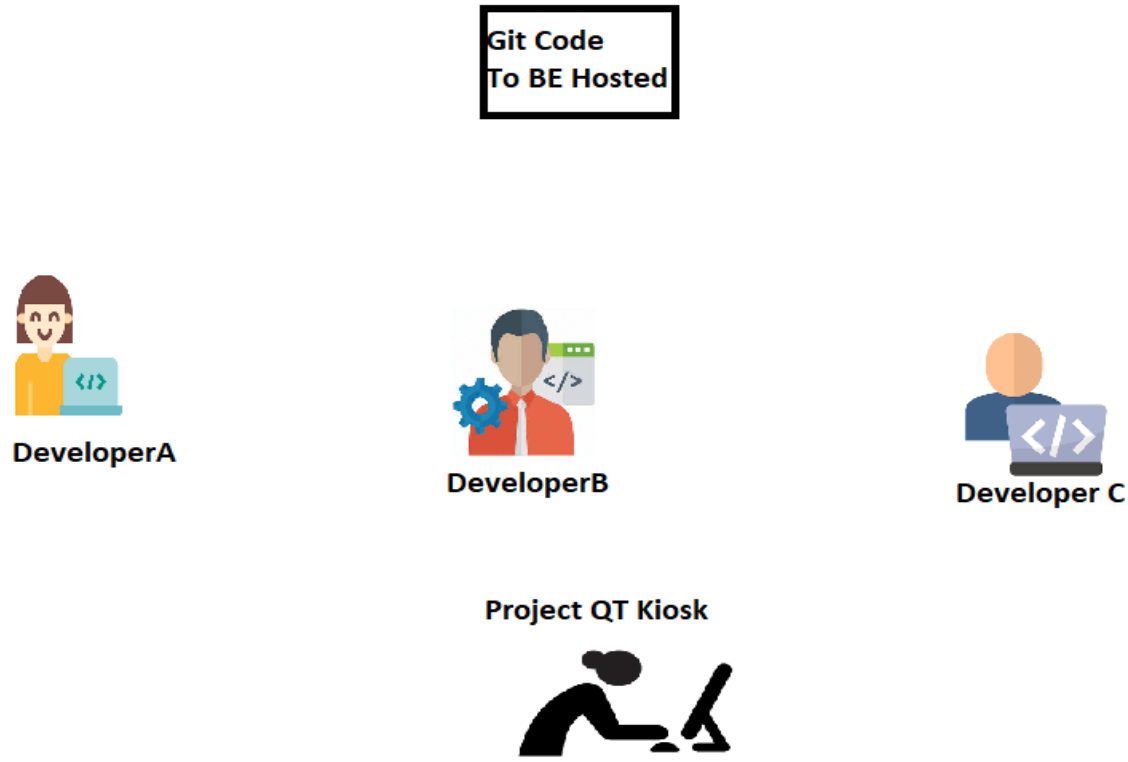
- In Git you will always have a branch and the default branch name is *master*
- In Enterprise work each branch represents a line of development
- Internally any branch in git will have link to *Latest version*
- Lets try to understand how git works using some internal commands (plumbing commands) such as git cat-file -p and git cat-file -t
- Git Commit => SHA1 Hash (changes, message, author, email)

- Git works like a content tracker and plumbing

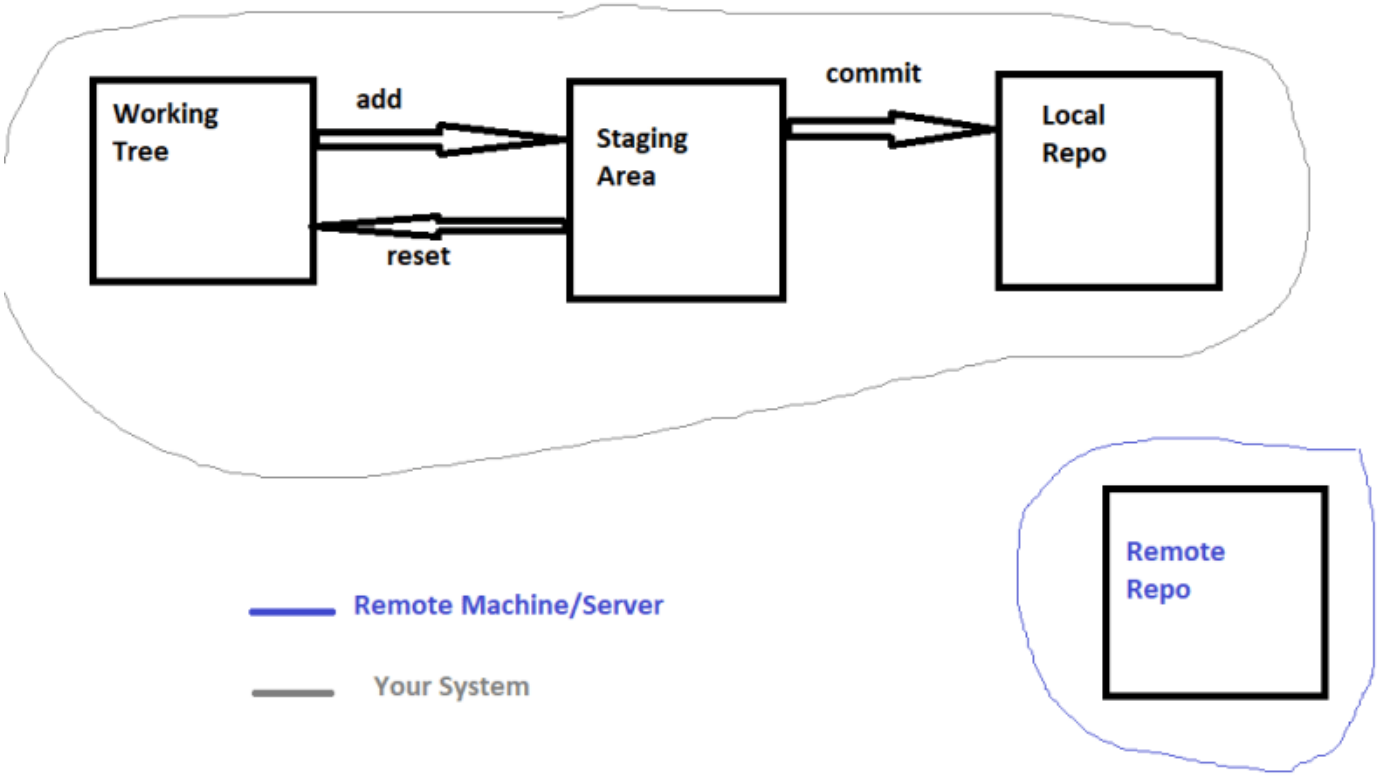



## Fourth Area of Git

- To Collaborate between developers, we need code to be hosted

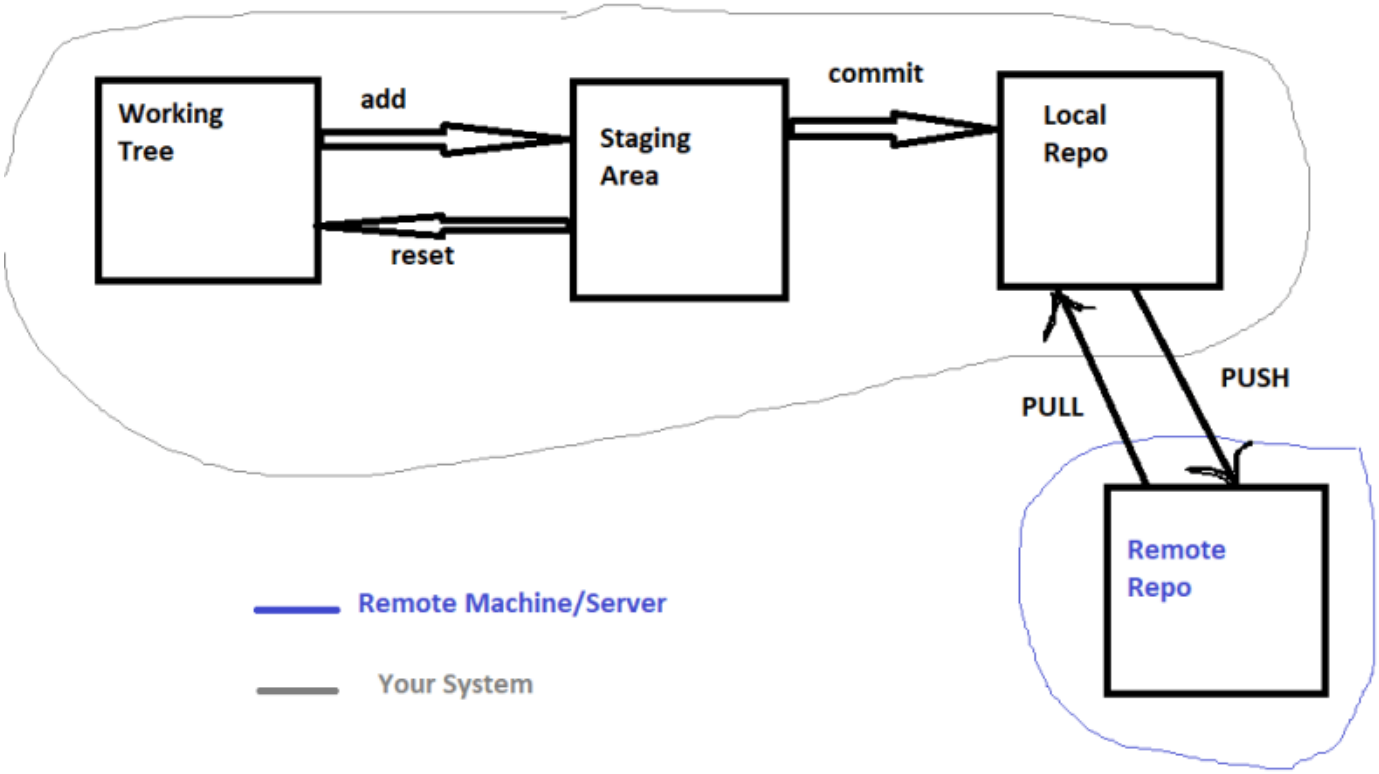


- Code can be hosted by any git server software/provider
  - Gitolite
  - GitHub
  - GitLab
  - BitBucket
  - Code Commit
  - Azure Source Repos
- Git to manage this will have a fourth area called as remote-repo




- Sending the commits to the remote repo is called as *push* and getting the commits from the remote repository is called as *pull*.
- If you don't have a remote repository yet, but you have local repo then we would add a remote repository to local
- If you have remote repository and you want that code to local system then we perform *clone*

- Default Remote-repository will have a name which is *origin*

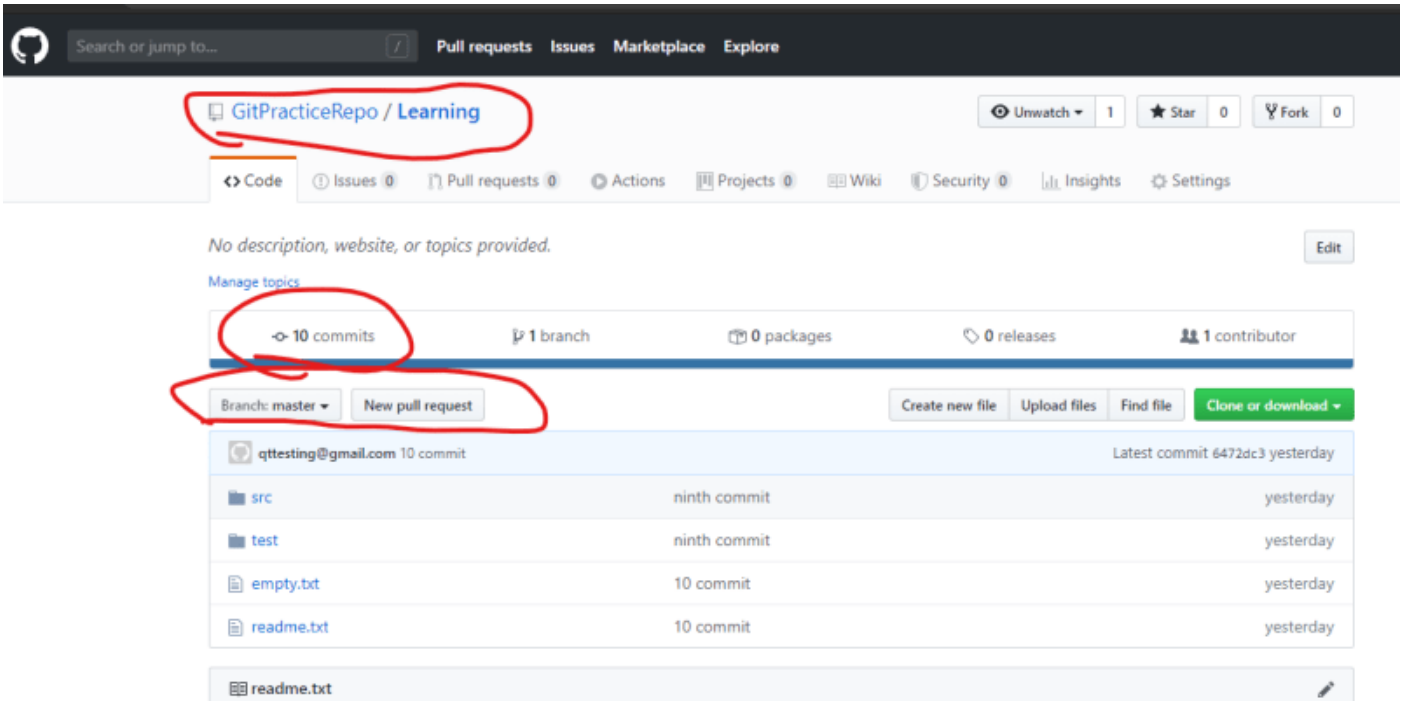



Scenario: You have a local repo and now you have remote repository created and want to send all the changes to remote repo.

- Connection between local and remote can be acheived by a command

```
git remote add --help # To know options
git remote add
```

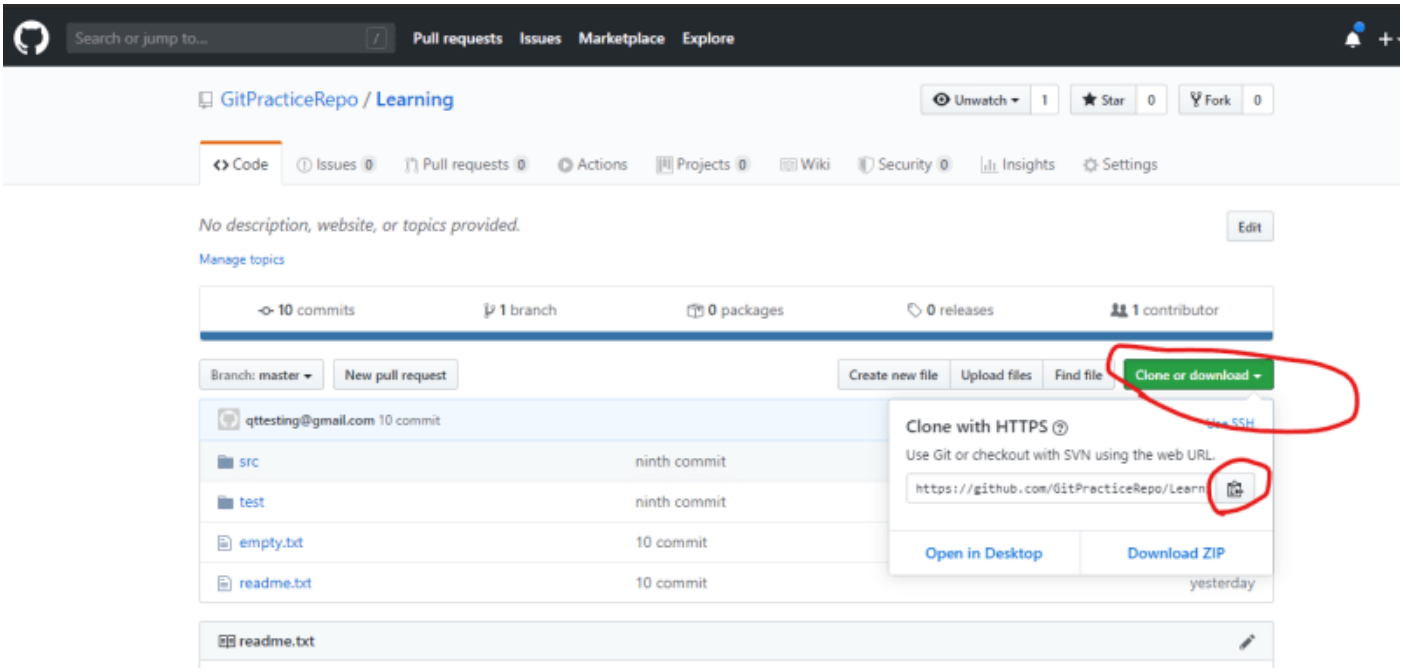
```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git remote add origin https://github.com/GitPracticeRepo/Learning.git

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git push --help

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git push -u origin master
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Delta compression using up to 4 threads
Compressing objects: 100% (26/26), done.
Writing objects: 100% (37/37), 3.16 KiB | 323.00 KiB/s, done.
Total 37 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/GitPracticeRepo/Learning.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```



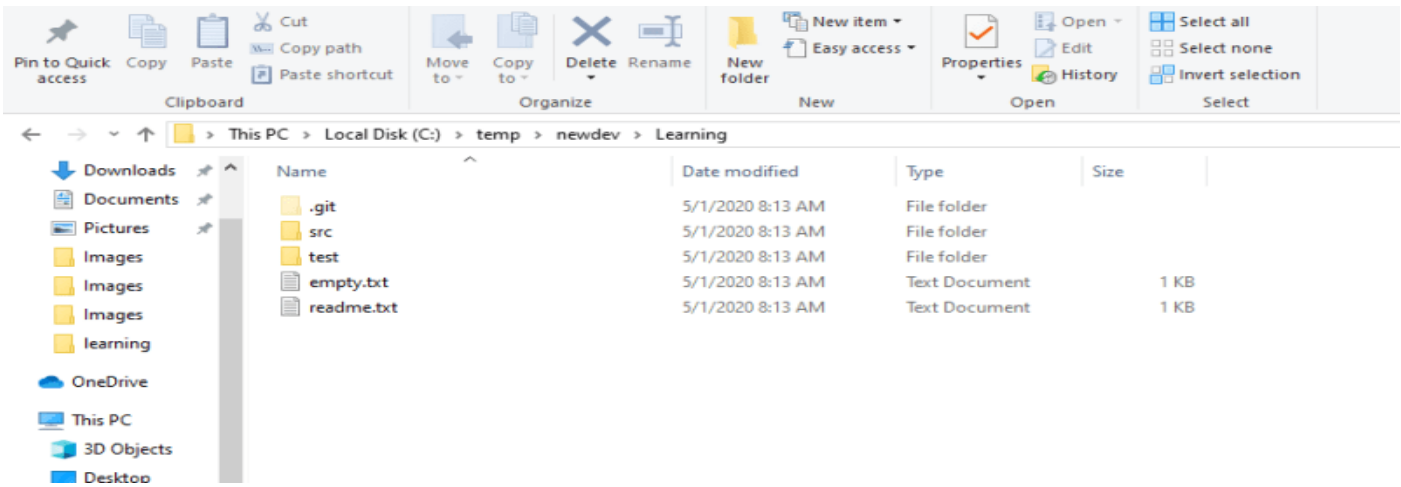

Scenario: A New Developer wants to get the code from remote repository

- In this case as mentioned earlier, we need to use clone

```
git clone <remote-url>
```




```
PS C:\temp\newdev> git clone https://github.com/GitPracticeRepo/Learning.git
Cloning into 'Learning'...
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 37 (delta 5), reused 37 (delta 5), pack-reused 0
Unpacking objects: 100% (37/37), done.
PS C:\temp\newdev> ls


    Directory: C:\temp\newdev


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/1/2020   8:13 AM                Learning


PS C:\temp\newdev>
```



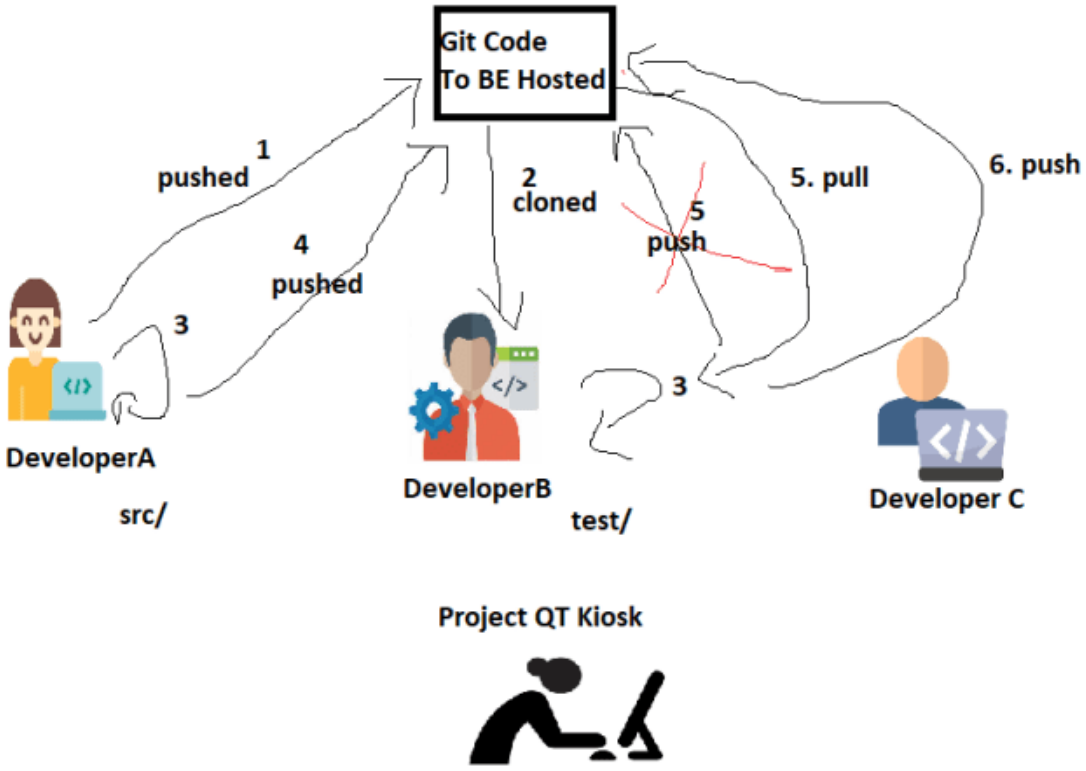
Scenario: Move all of the code from one repository to other repository

- It is all about adding a new repository and pushing the changes from any developers machine

```
git remote add <newremote> <newremoteurl>
git push -u <newremote> <branch-name>
```

```
QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git remote --help

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$ git remote
bitbucket
origin

QT@DESKTOP-HGH07L2 MINGW64 /d/learning (master)
$
```

Scenario: Multi Developer UseCase



1. Developer who is pushing the code to remote repository needs to be on latest version, so you have to do a pull before you push your changes.
    - Best Practice: Get Periodic Updates from remote using pull operations
    - Steps

```
PS C:\temp\newdev\Learning> git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/GitPracticeRepo/Learning
   6472dc3..4d55eb2  master     -> origin/master
Merge made by the 'recursive' strategy.
 src/main.py | 1 -
 1 file changed, 1 deletion(-)
PS C:\temp\newdev\Learning> git push origin master
Enumerating objects: 12, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 634 bytes | 317.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/GitPracticeRepo/Learning.git
   4d55eb2..40090df  master -> master
PS C:\temp\newdev\Learning>
```