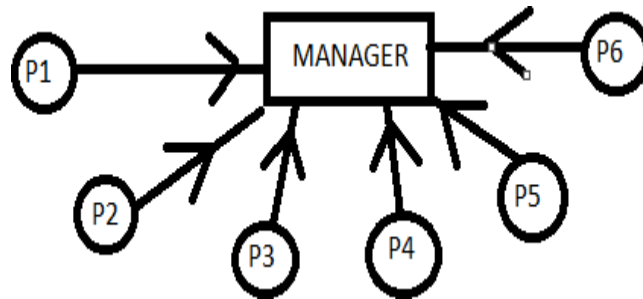# Introduction to Git

In previous time, when we work on a software development project we use to write different codes manually, merge that & compile.

It is very hard for a manager to remember who is giving which code.

So, to merge that codes we used a tool instead of person i.e. <u>Software Configuration Management or Source Code Management</u>. It is used to manage <u>versions of code, giving versions to codes to remember file</u>.

Types of source code management:

1-Centralised version control system.

2-Distributed Version control system.

Centralized Control Version System(CVCS):

Before git we use cvcs.
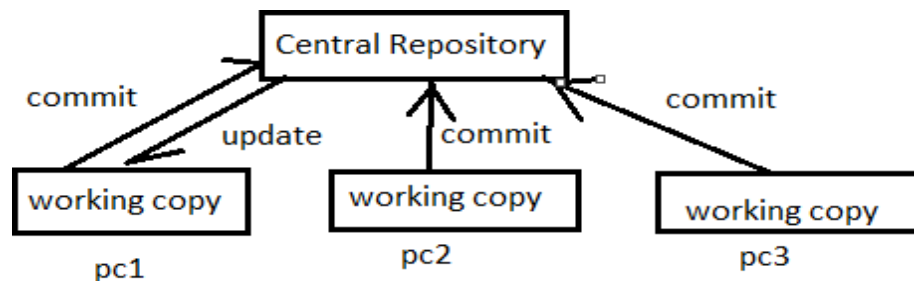
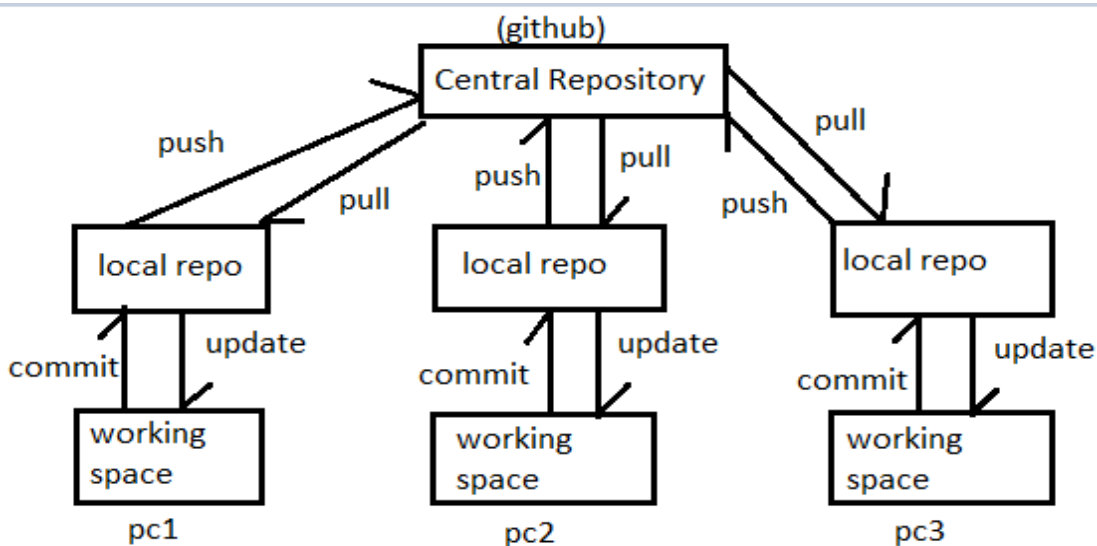Architecture of CVCS

SVN tool is used in cvcs.

Drawback of CVCS:

- ➢ It's not locally available, meaning you always need to be connected to a network to perform any action.
- ➢ Since everything is centralized, if central repository or server gets failed or crashed, you will loose entire data.

Distributed Control Version System(DVCS):

In DVCS, every contributor has a local copy or 'clone' of the central repository i.e. everyone maintain a local repository of their own which contains all the files & metadata present in main repository.

- ➢ Git is a DVCS.
- ➢ Git is manufactured by **Linus Torvald** in 2005.
- ➢ Before git linux uses bit keeper. It is a third party software. Linux uses bit keeper for versioning or version control.
- ➢ Linux stop using bit keeper because of some deal issue.
- ➢ Git is designed for linux kernel. It works on linux kernel.
- ➢ Git is a software or tool. It can be downloaded from internet.
- ➢ Github is different from git. It works on remote level.
- ➢ Git is a service work on local system level to manage versioning.

Architecture of DVCS

- In DVCS, we work on local system & save that work in local hard disk.
- Internet is not required to do any work because data will save in your system first.
- If repository(remote server) goes down or crashes, then no need to worry because your data will be saved in your local repository.
- Distributed means your code copy of same work is distributed at more than one place.
- Version control means, if you change your data very small then all teammates will know.

Difference between CVCS & DVCS:-

| CVCS	DVCS
In this, a client need to get local copy of source from server do the changes required & commit those changes to central source on server.	In DVCS, each client can have a local repository as well as have a complete history on it, client need to push the changes to branch which will be pushed then to server repository.
Cvcs system are easy to learn & setup.	Dvcs system are different for beginners. Multiple commands needs to be remembered.
Working on branches is difficult in cvcs. Developer often faces merge conflict.	Working on branches is easier in Dvcs. Developers faces less conflict.
Cvcs system don't provide offline access.	Dvcs systems are working fine on offline mode as a client copies the entire repository on their local machine.
Cvcs is slower as every command needs to communicate with server.	Dvcs is faster as mostly user deals with local copy without lifting server every time.
If cvcs server is down developer can't work.	If dvcs server is down, developer can work using their local copies.
Don't have personal working copy.	

Remember:

- Before CVCS, local version control system was used.

> ➢ It is used to save your codes on your local PC or Laptop, But if your PC or Laptop is crashed then all data will be lost.

Stages of Git & It's Terminology

Git workflow with Github:

| Stages	Description
1	Launch linux machine
2	Install git
3	Make a directory
4	Run git init(above directory converts into .git repository(local repo)
5	.git will divide into 3 region: ->workspace/working directory ->staging area ->local repo
6	Coding is done into workspace
7	Add coding file in staging area
8	Now commit code file from staging area to local repo branch
9	Push coding file to central repo(github)
10	Pulling code to other machine from github, you can see data in local repo/staging area/working directory.
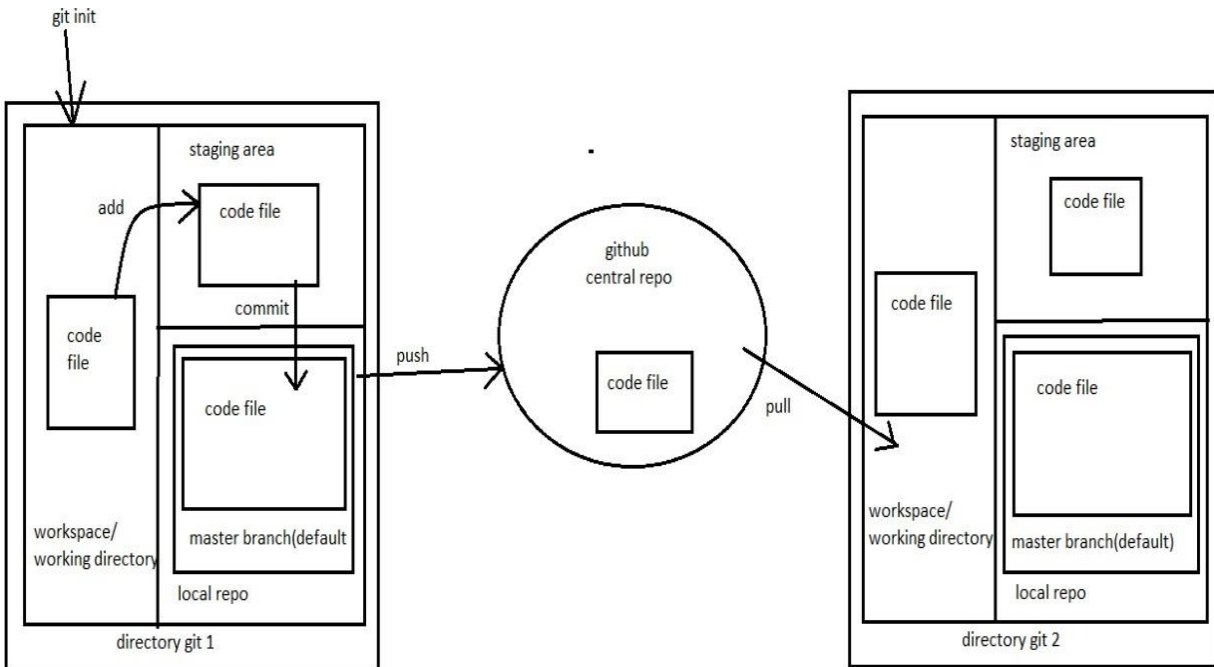
Working process of git

Repository:-

- ➢ It's a place where you have all your codes or kind of folder in server.
- ➢ It's a kind of folder related to one product.
- ➢ Changes are personal to that particular repository.

Server:- It stores all repository(local repo is also a server but it is private server). It contains metadata also.

Working Directory:-

- ➢ Where you see files physically & do modification.
- ➢ At a time you can work on particular branch.
- ➢ In CVCS, developers generally makes modification & commit their changes directly to the repository. But git uses different strategy, it doesn't track each & every modified file. Whenever you do commit an operation, git looks for the files present in the staging area, only those files present in the staging area are considered for commit & not all the modified files.
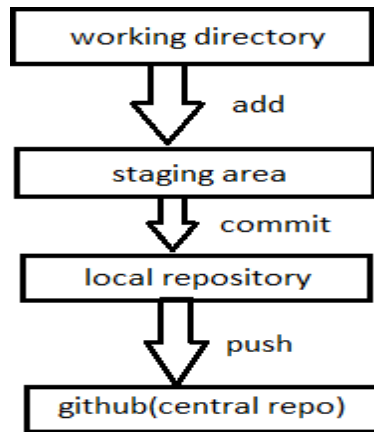
Summary of git workflow

Commit:-

- ➢ It means sending code file from staging area to local repository.
- ➢ It stores changes in local repository you will get an commit-ID.
- ➢ It's a 40 alpha-numeric characters.
- ➢ It uses SHA-1 checksum concept.
- ➢ Even if you change one dot(.) commit-ID will get changed.

SHA-1 Checksum Concept:-

- ➢ If you have a code of 1000 words in a file, so SHA-1 checksum will evaluate a binary value(like 234123411) & file will be sent to a person. Other side there will be SHA-1 checksum implemented & the output value for SHA-1 of that file is also same as sender value then it means there is no changes are made in that file, receiver gets that file as it is sent by sender.
- ➢ If any small changes is done like dot(.) then SHA-1 checksum value will be changed.
- ➢ It is used to know the status of code is changed or not in between sending & receiving.

Commit-ID/Version-ID/Version:-

- ➢ Reference to identity each change.

➢ To identify who changed the file.

Tags:- It assign a meaningful name with a specific version in the repository, once a tag is created for a particular save, even if you create anew commit, it will not be updated.

Snapshots(Increamental Backup):-

➢ Represents some data of a particular time.
➢ It is always incremental i.e. it stores the changed(appended data) only, not entire copy.
➢ It'll save only updated stuff previous code will not be saved with new code line as a new file, but when you retrieve all code line it will automatically merge old & new code line Y show all code line in a single file.

Push:- Push operation copies the changes from a local repository server to a remote or central repo. It is used to store changes permanently into the git repository.

Pull:- Pull operation copies the changes from a remote repository to a local machine.

Branch:-

➢ Product is same, so one repository but different task .
➢ Each task has one separate branch.
➢ Finally merges all branches(code).
➢ Useful when you want to work parallel.
➢ Can create one branch on the basis of another branch.
➢ Changes are personal to that particular branch.
➢ Default branch is 'Master'.
➢ File created in workspace will be visible in any of the branch workspace until you commit. Once you commit then that file belongs to that particular branch.

Advantages of Git

- ➢ Free & open source.
- ➢ <u>Fast & Small</u>:- As most of the operations are performed locally, therefore it's fast. Small in size too.
- ➢ <u>Security</u>:- Git uses a common cryptographic hash function called secure hash function(SHA-1) to name & identify objects within it's database.
- ➢ No need of powerful hardware.
- ➢ <u>Easier Branching</u>:- If we create a new branch, it'll copy all the codes to the new branch.

Types of Repositories:

I. Bare Repository(Central Repo)
- ✓ Store & share only.
- ✓ All central repositories are bare repo.

II. Non-Bare Repository(Local Repo)
- ✓ Where you can modify the files.
- ✓ All local repositories are non-bare repository.

How to use git & create github account?

<u>Configuring git</u>:-

- ➢ sudo su
- ➢ yum update –y
- ➢ yum install git
- ➢ git –version
- ➢ git config –global user.name "akas"
- ➢ git config –global user.email akas@gmail.com
- ➢ git config –list (to see created user in git)

```
[root@localhost abdlqdr]# git config --global user.name "Abdul"
[root@localhost abdlqdr]# git config --global user.email "abdul@gmail.com"
[root@localhost abdlqdr]# git config --list
user.name=Abdul
user.email=abdul@gmail.com
[root@localhost abdlqdr]#
```

<u>Creating github account</u>:-

➢ open www.github.com
➢ create an account by sign up.
➢ Click join free plan.
➢ Now click on complete setup.
➢ Now verify your email from gmail.
➢ Sign out then sign in again.

How to Commit, Push & Pull from Github?

➢ Login into linux machine.
➢ Create a directory & go inside it.

```
[root@localhost abdlqdr]# mkdir abdulgit
[root@localhost abdlqdr]# ls
abdulgit
[root@localhost abdlqdr]# cd abdulgit
[root@localhost abdulgit]#
```

➢ Run (git init) command inside directory to make it local repo.

```
[root@localhost abdulgit]# git init
Initialized empty Git repository in /home/abdlqdr/abdulgit/.git/
[root@localhost abdulgit]# ls -a
.  ..  .git
[root@localhost abdulgit]#
```

➢ Create a file-> touch myfile(put some data).

```
[root@localhost abdulgit]# touch codefile
[root@localhost abdulgit]# ls
codefile
[root@localhost abdulgit]#
```

➢ git status (to check what data is in local repo).

```
[root@localhost abdulgit]# git status
On branch master

No commits yet

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

        codefile

nothing added to commit but untracked files present (use "git add" to track)
[root@localhost abdulgit]#
```

➢ git add . (to add code file from working area to staging area).

```
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git status
On branch master

No commits yet

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

        new file:    codefile

[root@localhost abdulgit]#
```

➢ git commit –m "first commit from centos" (to commit & give a message).

```
[root@localhost abdulgit]# git commit -m "first commit from centos"
[master (root-commit) c4bc63c] first commit from centos
 1 file changed, 1 insertion(+)
 create mode 100644 codefile
[root@localhost abdulgit]#
```

➢ git status (to check status of a code file).

```
[root@localhost abdulgit]# git status
On branch master
nothing to commit, working tree clean
[root@localhost abdulgit]#
```

➢ git log (it tells which commit is done by which user).

```
[root@localhost abdulgit]# git log
commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472  (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos
[root@localhost abdulgit]#
```

➢ git show <commit-id> (to see code of a particular commit).

```
[root@localhost abdulgit]# git show  c4bc63c3fda2ad0c
commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472  (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos

diff --git a/codefile b/codefile
new file mode 100644
index 0000000..d4c9c76
--- /dev/null
+++ b/codefile
@@ -0,0 +1 @@
+shaher me jin ki sakhawat ka bada charcha hai,
[root@localhost abdulgit]#
```

➢ git remote add origin <url> (to add github(central repo) with our local
   repo).

```
[root@localhost abdulgit]# git remote add origin https://github.com/abdlqdr/gitlearning.git
[root@localhost abdulgit]#
```

➢ git push –u origin master (to push code file from local repo to github in
   master branch).

```
[root@localhost abdulgit]# git push -u origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 256 bytes | 64.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/abdlqdr/gitlearning.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
[root@localhost abdulgit]#
```

You can see your file in github(central repo).

Now updating content in the previous file & checking status,

```
[root@localhost abdulgit]# cat >>codefile
mere tute hue kashkol se dar jate hai.
[root@localhost abdulgit]# git status
On branch master
Your branch is up to date with 'origin/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:   codefile

no changes added to commit (use "git add" and/or "git commit -a")
[root@localhost abdulgit]#
```

Now adding file to staging & checking status,

```
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   codefile

[root@localhost abdulgit]#
```

Now committing file to local repo & checking status.

```
[root@localhost abdulgit]# git commit -m "modifcation in codefile"
[master 2874a1c] modifcation in codefile
 1 file changed, 1 insertion(+)
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[root@localhost abdulgit]#
```

Now looking total git commits.

```
[root@localhost abdulgit]# git log
commit 2874a1ce855e66deb66fc799d919b395d3bac201 (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 12:15:54 2020 +0530

    modifcation in codefile

commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472 (origin/master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 11:33:13 2020 +0530

    first commit from centos
[root@localhost abdulgit]#
```

Now seeing the updated content in file.

```
[root@localhost abdulgit]# git show  2874a1ce855e66de
commit 2874a1ce855e66deb66fc799d919b395d3bac201 (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Fri Sep 4 12:15:54 2020 +0530

    modifcation in codefile

diff --git a/codefile b/codefile
index d4c9c76..08aace8 100644
--- a/codefile
+++ b/codefile
@@ -1 +1,2 @@
 shaher me jin ki sakhawat ka bada charcha hai,
+mere tute hue kashkol se dar jate hai.
[root@localhost abdulgit]#
```

Now pushing modification code file to github.

```
[root@localhost abdulgit]# git push -u origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Branch 'master' set up to track remote branch 'master' from 'origin'.
Everything up-to-date
[root@localhost abdulgit]#
```

Pulling Mechanism

Login into linux machine.

Create a directory, go inside it & run 'git init' command to create local repo.

```
root@ubuntuserver:/home/abdlqdr# mkdir quadirgit
root@ubuntuserver:/home/abdlqdr# ls
quadirgit
```

Git remote add origin <github url> (to connect github with local repo).

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git remote add origin https://github.com/abdlqdr/gitlea
rning.git
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

git pull –u origin master/git pull origin master –allow

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git pull origin master --allow
From https://github.com/abdlqdr/gitlearning
 * branch             master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 2 ++
 codefile  | 2 ++
 2 files changed, 4 insertions(+)
 create mode 100644 README.md
 create mode 100644 codefile
```

git log (to show all commit)

add some code to codefile

```
root@ubuntuserver:/home/abdlqdr/quadirgit# cat >>codefile
khalike qauno makan daure fugha kam kar de
```

git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# 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:   codefile

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

git add . then git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   codefile
```

git commit –m "codefile modified" /then git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "codefile modified"
[master 2e899f1] codefile modified
 1 file changed, 1 insertion(+)
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
```

git push origin master

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git push origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 1.14 KiB | 61.00 KiB/s, done.
Total 11 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/abdlqdr/gitlearning.git
   5c4a4b2..2e899f1  master -> master
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

git ignore Command

- ➢ To ignore some files while committing.
- ➢ Create one hidden file ".gitignore" & enter file format which you want to ignore.

```
[root@localhost abdulgit]# vi .gitignore
[root@localhost abdulgit]# ls
codefile  file1  README.md  ubuntufile
[root@localhost abdulgit]# ls -a
.  ..  codefile  file1  .git  .gitignore  README.md  ubuntufile
```

```
*.css
*.py
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
".gitignore" 2L, 11C
```

➢ Run git add .ignore & create some file using touch command.

```
[root@localhost abdulgit]# git add .gitignore
[root@localhost abdulgit]# touch file1.java file2.css file3.txt file4.py
[root@localhost abdulgit]# ls
codefile  file1  file1.java  file2.css  file3.txt  file4.py  README.md  ubuntufile
```

➢ Run git status, you will see it is showing only 2 files other 2 are ignored.

```
[root@localhost abdulgit]# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore

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

        file1.java
        file3.txt
```

➢ Now, you can commit these files as you need.

```
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   file1.java
        new file:   file3.txt

[root@localhost abdulgit]# git commit -m " java & text files"
[master 383e6f7]  java & text files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.java
 create mode 100644 file3.txt
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

➢ git log -1 (to show last & recent commit).

```
[root@localhost abdulgit]# git log -1
commit 383e6f7def8620a65942e44578ba5e2f2e8bdd4c  (HEAD -> master)
Author: Abdul <abdul@gmail.com>
Date:   Sat Sep 5 21:49:44 2020 +0530

    java & text files
```

➢ Git log -2 (to show last 2 commit)

```
[root@localhost abdulgit]# git log -2
commit 383e6f7def8620a65942e44578ba5e2f2e8bdd4c  (HEAD -> master
Author: Abdul <abdul@gmail.com>
Date:   Sat Sep 5 21:49:44 2020 +0530

    java & text files

commit 6fb73bad27528cbd2cdd0a35425a603d191a1c4a
Author: Abdul <abdul@gmail.com>
Date:   Sat Sep 5 21:48:40 2020 +0530

    java & text files
```

➢ git log –oneline (to display all commits in one line).

```
[root@localhost abdulgit]# git log --oneline
383e6f7 (HEAD -> master)  java & text files
6fb73ba  java & text files
c4a1e2e (origin/master) update by 5/sep
2e899f1 codefile modified
11f6f58 erge branch 'master' of https://github.com/abdlqdr/gitlearning
5c4a4b2 added read me| added string 'Hello'
8a95ee7 first commit from ubuntu
6fda145 first commit from ubuntu
2874a1c modifcation in codefile
c4bc63c first commit from centos
```

➢ git log –grep "commit" (it's used to search a commit by a word).

```
[root@localhost abdulgit]# git log --grep "commit"
commit 8a95ee794356db9be55c6d91b8b41cddc137bbaf
Author: Quadir <quadir@gmail.com>
Date:    Sat Sep 5 13:55:49 2020 +0000

    first commit from ubuntu

commit 6fda1457bb669221b17519a84fac530e81642de8
Author: Quadir <quadir@gmail.com>
Date:    Fri Sep 4 19:06:53 2020 +0000

    first commit from ubuntu

commit c4bc63c3fda2ad0cc6f3a0bc36b33112abdc6472
Author: Abdul <abdul@gmail.com>
Date:    Fri Sep 4 11:33:13 2020 +0530

    first commit from centos
```
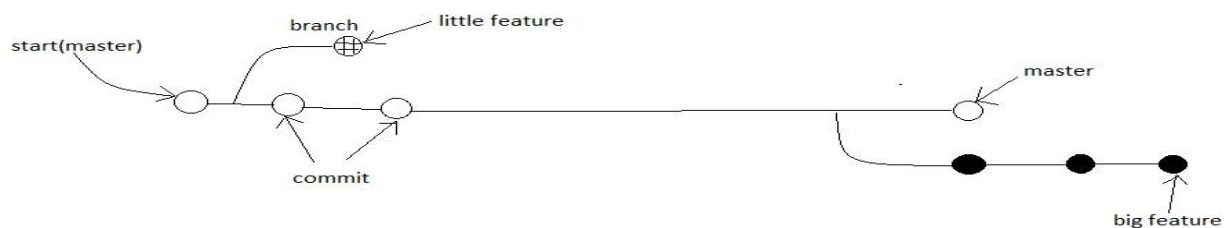
How to create Branch?

Branch structure

The above diagram visualizes a repository with two isolated lines of development.
One for a little feature, by developing them is branches. It's not possible to work

on both of them in parallel but it also keeps the main master branch free from error.

Properties of Branches:-

➢ Each task has one separate branch.

➢ After done with coding, merge other branches with master.

➢ This concept is useful for parallel development.

➢ You can create any number of branches.

➢ Changes are personal to that particular branch .

➢ Default branch is 'master'.

➢ File created in workspace will be visible in any of the branch workspace until you commit once you commit, then that file belongs to that particular branch.

```
[root@localhost abdulgit]# git branch
  branch1
* master
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# cat >secondfile
hi
[root@localhost abdulgit]# ls
branchfile1  file1        file2.css  file4.py   secondfile
codefile     file1.java  file3.txt  README.md  ubuntufile
[root@localhost abdulgit]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
[root@localhost abdulgit]# ls
codefile  file1  file1.java  file2.css  file3.txt  file4.py  README.md  secondfile  ubuntufile
[root@localhost abdulgit]# git branch
  branch1
* master
```

```
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "2nd commit from branch1"
[branch1 81e4b59] 2nd commit from branch1
 1 file changed, 1 insertion(+)
 create mode 100644 secondfile
[root@localhost abdulgit]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
[root@localhost abdulgit]# ls
codefile  file1  file1.java  file2.css  file3.txt  file4.py  README.md  ubuntufile
```

➢ When created new branch, data of existing branch is copied to new branch.

```
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# ls
branchfile1    file1           file2.css    file4.py       secondfile
codefile       file1.java   file3.txt    README.md    ubuntufile
[root@localhost abdulgit]# git branch branchA
[root@localhost abdulgit]# git branch
* branch1
  branchA
  master
[root@localhost abdulgit]# git checkout branchA
Switched to branch 'branchA'
[root@localhost abdulgit]# ls
branchfile1    file1           file2.css    file4.py       secondfile
codefile       file1.java   file3.txt    README.md    ubuntufile
[root@localhost abdulgit]# git branch
  branch1
* branchA
  master
```

Commands of Branch:-

➢ To see list of available branches (git branch).

```
[root@localhost abdulgit]# git branch
  branch1
* master
```

➢ To create new branch (git branch <branch name>).

```
[root@localhost abdulgit]# git branch branch1
[root@localhost abdulgit]# git branch
  branch1
* master
```

➢ To switch branch. (git checkout <name of switching branch>). (*) shows your current branch.

```
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# git branch
* branch1
  master
```

➢ To delete branch (git branch –d/-D <branch name>). [-D is used to delete forcefully].

```
[root@localhost abdulgit]# git branch
  branch1
* master
[root@localhost abdulgit]# git branch -d branch1
Deleted branch branch1 (was 383e6f7).
[root@localhost abdulgit]# git branch
* master
```

Branch merge:-

You can't merge branches of different repositories.

We use pulling mechanism to merge branches.

To merge branches (git merge <branch name>). Then run git log –oneline.

```
[root@localhost abdulgit]# git merge branch1
Updating 383e6f7..c2e2ba9
Fast-forward
 branchfile1 | 3 +++
 secondfile  | 1 +
 2 files changed, 4 insertions(+)
 create mode 100644 branchfile1
 create mode 100644 secondfile
[root@localhost abdulgit]# git log --oneline
c2e2ba9 (HEAD -> master, branch1) branchfile1 updated
81e4b59 2nd commit from branch1
10dcfae branch1 commit
383e6f7  java & text files
6fb73ba  java & text files
c4a1e2e (origin/master) update by 5/sep
2e899f1 codefile modified
11f6f58 erge branch 'master' of https://github.com/abdlqdr/gitlearning
5c4a4b2 added read me| added string 'Hello'
8a95ee7 first commit from ubuntu
6fda145 first commit from ubuntu
2874a1c modifcation in codefile
c4bc63c first commit from centos
```

Push the commits to github(central repo). (git push origin master)

```
[root@localhost abdulgit]# git push origin master
Username for 'https://github.com': abdlqdr
Password for 'https://abdlqdr@github.com':
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 2 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), 1.24 KiB | 105.00 KiB/s, done.
Total 15 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/abdlqdr/gitlearning.git
   c4a1e2e..c2e2ba9  master -> master
```

Check github to see the changes.

branchfile1	branchfile1 updated	14 minutes ago
codefile	update bv 5/sep	16 hours ago

git conflict:-

➢ When same file having different content in different branch if you do branch merge, conflict occurs (resolve conflict then add & commit).

➢ Conflict occurs when you merge branch.

Ex 1:

Branch:Master   branch:Branch1

File name:abdl   File name:abdl

Content:My name is Akas.   Content: My name is Akas. I

                                                            live in India.

➢ When you merge above branch then there is no conflict because there is same line available in both file. It'll know second line to merge.

Ex 2:

Branch:Master   branch:Branch1

File name:ak   File name:ak

Content:My name is akas.   Content: I live in India.

➢ In above example, when we merge branches git will confuse which data to add above or below. When data is completely different then conflict occurs.

```
[root@localhost abdulgit]# cat >abdl
hello abdl
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "first commit conflict
[master 8f4d34e] first commit conflict
 1 file changed, 1 insertion(+)
 create mode 100644 abdl
[root@localhost abdulgit]# git checkout branch1
Switched to branch 'branch1'
[root@localhost abdulgit]# cat >abdl
hi abdl
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "conflict branch1"
[branch1 82cba1c] conflict branch1
 1 file changed, 1 insertion(+)
 create mode 100644 abdl
[root@localhost abdulgit]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

➢ To correct conflict, In master branch open file in vi editor(vi abdl) & edit as you want & save this file, git understand that & merge file.

```
[root@localhost abdulgit]# git merge branch1
Auto-merging abdl
CONFLICT (add/add): Merge conflict in abdl
Automatic merge failed; fix conflicts and then commit the result.
[root@localhost abdulgit]# vi abdl
```

➢ After that you have to add & commit that merged file in master branch.

```
[root@localhost abdulgit]# git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both added:        abdl

no changes added to commit (use "git add" and/or "git commit -a")
[root@localhost abdulgit]# git add .
[root@localhost abdulgit]# git commit -m "conflict resolved"
[master 471fd6a] conflict resolved
[root@localhost abdulgit]# git log --oneline
471fd6a (HEAD -> master) conflict resolved
82cba1c (branch1) conflict branch1
8f4d34e first commit conflict
c2e2ba9 (origin/master) branchfile1 updated
```

git Stashing:-

> Suppose, you are implementing a new feature for your product. Your code is in progress & suddenly a customer escalation comes because of this, you have to keep aside your new feature work for few hours. You can't commit your partial code & also can't throw away your changes. So, you need some temporary storage, when you can store your partial changes & later on commit it.
> To stash an item (only applies to modified files not new files).
> git stash

```
root@ubuntuserver:/home/abdlqdr/quadirgit# touch demofile
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "demofile"
[master d14a9b3] demofile
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demofile
root@ubuntuserver:/home/abdlqdr/quadirgit# vi demofile
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash
Saved working directory and index state WIP on master: d14a9b3 demofile
```

> To see stashed item list
> git stash list

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash list
stash@{0}: WIP on master: d14a9b3 demofile
stash@{1}: WIP on master: 24f8dda practice conflict resolved
```

> To apply stashed items(working file will move to workspace).
> git stash apply stash stash@{1}

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash apply stash@{1}
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:   demofile

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

➢ Then you can add & commit

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "second code done"
[master 2d3e81c] second code done
 1 file changed, 2 insertions(+), 1 deletion(-)
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
root@ubuntuserver:/home/abdlqdr/quadirgit# git log --oneline
2d3e81c (HEAD -> master) second code done
293ec7f first code done
d14a9b3 demofile
24f8dda (origin/master) practice conflict resolved
```

➢ To clear the stash items
    git stash clear

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash clear
root@ubuntuserver:/home/abdlqdr/quadirgit# git stash list
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

➢ When you get back your data from stash to workspace. Then that
    data will be also there in stashing area.

git reset:-

git reset is a powerful command undo local changes to the state of a git
repository.

To reset staging area.

git reset <file name>

```
root@ubuntuserver:/home/abdlqdr/quadirgit# ls
abdl          codefile   file1       file3.txt  secondfile  ubuntufile
branchfile1   demofile   file1.java  README.md  testfile    xyz
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   testfile

root@ubuntuserver:/home/abdlqdr/quadirgit# git reset testfile
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        testfile

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

git reset .

```
root@ubuntuserver:/home/abdlqdr/quadirgit# cat >testfile
my name is khan.
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   testfile

root@ubuntuserver:/home/abdlqdr/quadirgit# git reset .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        testfile

nothing added to commit but untracked files present (use "git add" to track)
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

To reset the changes from both staging area & working directory at a time.
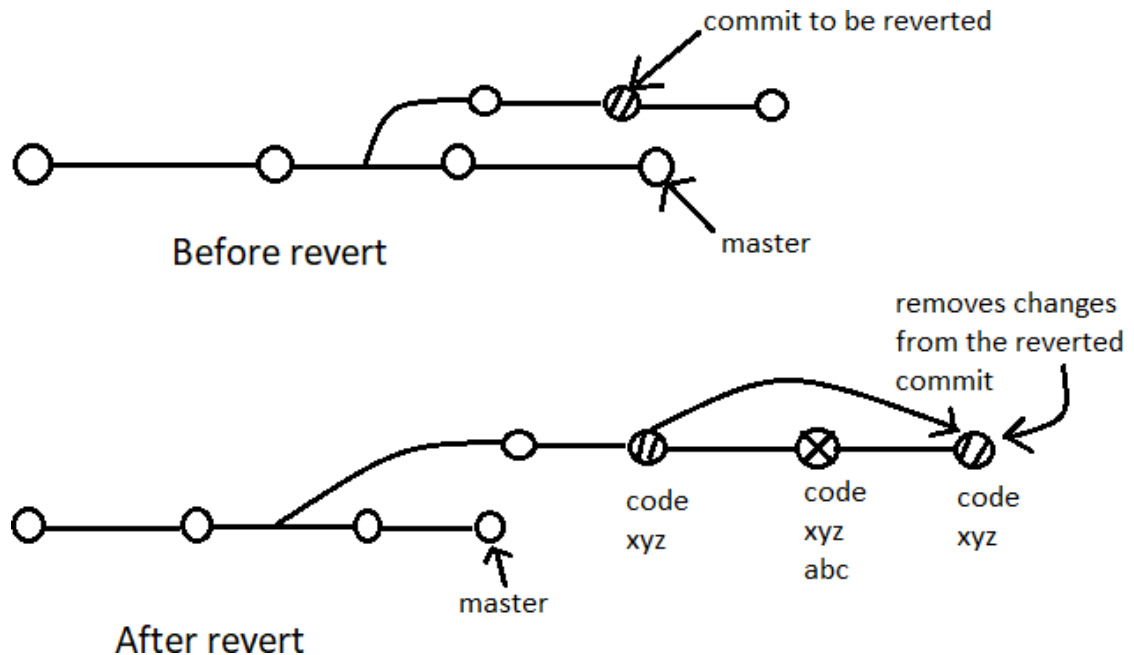
git reset –hard

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   testfile

root@ubuntuserver:/home/abdlqdr/quadirgit# git reset --hard
HEAD is now at 2d3e81c second code done
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
root@ubuntuserver:/home/abdlqdr/quadirgit#
```

git revert:-

- ➤ Revert command helps you undo an existing commit.
- ➤ It doesn't delete any data in the process instead rather git creates new commit with the included files reverted to their previous state. So, your version control history moves forward while the state of your file moves backward.

Before revert

After revert

➢ When you revert a commit, a commit id is assigned to reverted commit.

Command for git revert:-

git status

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
nothing to commit, working tree clean
```

cat >newfile

```
root@ubuntuserver:/home/abdlqdr/quadirgit# cat >newfile
guftgu tune sikhai hai ke mai gunga tha
```

git add .

git commit –m "code"

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git add .
root@ubuntuserver:/home/abdlqdr/quadirgit# git commit -m "checking revert"
[master ff72eb1] checking revert
 1 file changed, 1 insertion(+)
 create mode 100644 newfile
```

git log –oneline

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git log --oneline
ff72eb1 (HEAD -> master) checking revert
f9b75e4 (origin/master) code file complete
f939e20 code file stashing test
2d3e81c second code done
293ec7f first code done
```

git revert <commit id>

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git revert  c720124c89cf496a
[master 2ed04fc] Revert "checking revert update" plese consider this code.
 1 file changed, 1 deletion(-)
```

How to remove untracked files?

git clean –n (dry run)

git clean –f (forcefully)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# touch file1 file2 file3 file4
root@ubuntuserver:/home/abdlqdr/quadirgit# ls
abdl          codefile   file1       file2   file3.txt   newfile     secondfile   xyz
branchfile1   demofile   file1.java  file3   file4       README.md   ubuntufile
root@ubuntuserver:/home/abdlqdr/quadirgit# git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2
        file3
        file4

nothing added to commit but untracked files present (use "git add" to track)
root@ubuntuserver:/home/abdlqdr/quadirgit# git clean -n
Would remove file2
Would remove file3
Would remove file4
root@ubuntuserver:/home/abdlqdr/quadirgit# git clean -f
Removing file2
Removing file3
Removing file4
root@ubuntuserver:/home/abdlqdr/quadirgit# ls
abdl          codefile   file1       file3.txt   README.md     ubuntufile
branchfile1   demofile   file1.java  newfile     secondfile    xyz
```

Tag in git:-

Tag operation allows you giving meaningful names to a specific version in the repository.

To apply tag.

git tag –a <tag name> -m <message> <commit id>

git tag (to see tags)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag -a important -m "imp commit" c720124
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag
important
```

git show <tag name> (to see particular commit content by using tag)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git show important
tag important
Tagger: Quadir <quadir@gmail.com>
Date:   Mon Sep 7 07:25:23 2020 +0000

imp commit

commit c720124c89cf496aa94b4a48fdce0886c363bba3 (tag: important)
Author: Quadir <quadir@gmail.com>
Date:   Mon Sep 7 06:46:01 2020 +0000

    checking revert update

diff --git a/newfile b/newfile
index c84dba5..0f3dcc0 100644
--- a/newfile
+++ b/newfile
@@ -1 +1,2 @@
 guftgu tune sikhai hai ke mai gunga tha
+jab mai bolu to baton me asar bhi dena.
```

git tag –d <tag name> (to delete tag)

```
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag -d important
Deleted tag 'important' (was 747fa38)
root@ubuntuserver:/home/abdlqdr/quadirgit# git tag
root@ubuntuserver:/home/abdlqdr/quadirgit# git log -oneline
```

Github Clone:-

➢ Open github website.
➢ Login & choose existing repository, copy github url.

➢ Now, go to your linux machine, & run command

git clone <url of github repo> (as you can see gitlearning repo folder)

```
[root@localhost abdlqdr]# git clone https://github.com/abdlqdr/gitlearning.git
Cloning into 'gitlearning'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 100% (83/83), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 83 (delta 27), reused 76 (delta 23), pack-reused 0
Unpacking objects: 100% (83/83), done.
[root@localhost abdlqdr]# ls
abdulgit   gitlearning
[root@localhost abdlqdr]#
```

➢ It creates a local repo automatically in linux machine with the same name
as in github account. As you can see above.

```
[root@localhost abdlqdr]# cd gitlearning
[root@localhost gitlearning]# ls
abdl          codefile   file1       file3.txt   README.md    ubuntufile
branchfile1   demofile   file1.java  newfile     secondfile   xyz
[root@localhost gitlearning]#
```