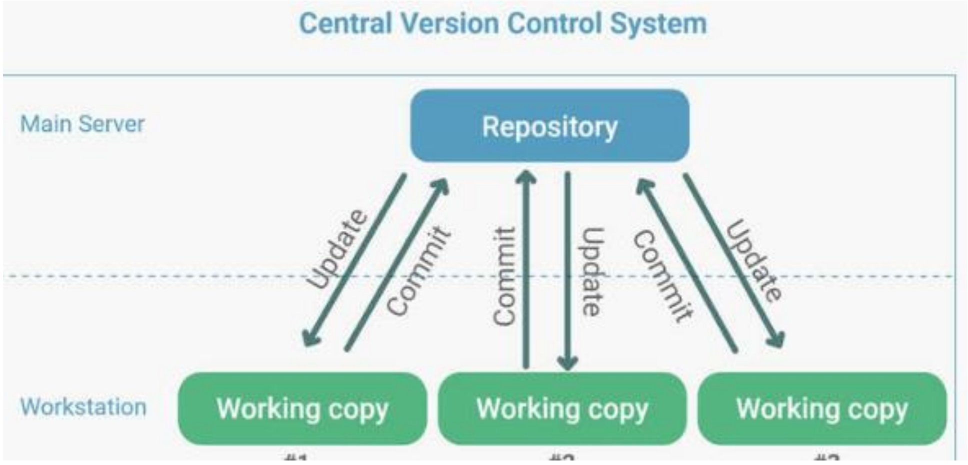

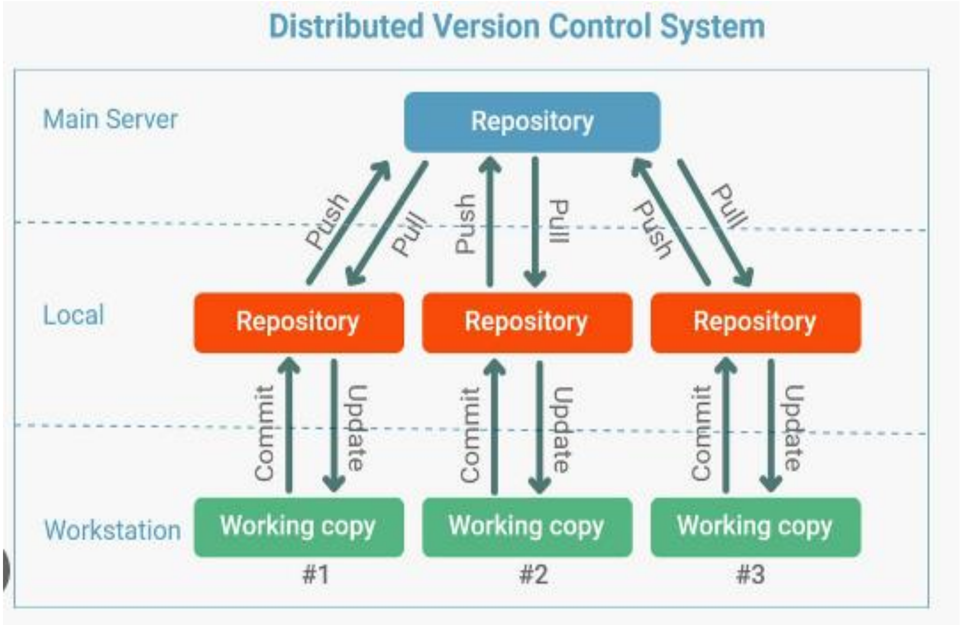

Git has three stages of workflow -1. **Working area 2. Staging area 3. Local Repository**.

We send data or code from working area to staging area by **add command** and staging area to Local repository by **commit command** and finally send data/code from Local repo to central repo by **push command**.

Update Linux operating system in working area (Mumbai Ec2-user)

```
# yum update -y
```

```
# yum install git -y
```

```
# which git
```

```
User /bin /git
```

```
# git –version
```

```
2.40.1
# git config --global user.name "Ashok"
# git config --global user.email ashok.anupam465@gmail.com
# git config -list  (this command shows the all configurated list)
```

Now work inside the Mumbai machine, create Directory and make file inside localrepo

```
[Ec2-user] # mkdir mumbaigit
```

```
[Ec2-user] # cd mumbaigit
```

```
[Mumbaigit]# git init  ( init command turn Dir into local Repo)
```

```
[Mumbaigit]# cat >Mumbai1  (write inside the [Dir] local repo by cat > command)
```

```
# cat Mumbai 1  (to check the data/code what has been written in repo)
```

Put and write some code/data inside the file mumbai1, and come out by Ctrl+d

I Love India

```
# git status
```

Untracked files: Mumbai1 (it's in red color means not added yet staging aria)

```
[mumbaigit]# git add .  (Add command to add created file to staging aria)
```

```
# git status
```

New file: Mumbai1 (it's in green color means added staging area)

Now commit data from staging area to Local repo

```
[Mumbaigit]# git commit -m "first commit from Mumbai"
```

(**m**=message)

```
# git status
```

```
# git log
```

(to check what commits have been done, when and who did?)

You will see commit Id like 12345678KD458F4IW3E4 . Author, Mail id, Date, Time, message: first commit from
Mumbai

```
Mumbaigit]# git show <commit-id>
```

(**show command** the content of commit ID)

first commit from Mumbai

+ I Love India

If we run the git commit command again it will show nothing to commit, working tree clean means data has been committed.

If want to send this code to my central repository, I have to connect local repo to central repo first, for this action I have to create a new repository (any name) and paste the URL of git repo and execute command as given below

```
[Mumbaigit] # git remote add origin
 https://github.com/ashok1012/centralgit.git
```

Now local repo has been connected to central repo, for pushing data to central repo execute this command

```
[Mumbaigit] # git push -u origin master
```

(**push command** local repo to central repo)

It will ask for **username and password** of your git hub account, after filling this and you can see all committed data/code inside central repo.

Now create a machine in Singapore region and connect to git hub.

```
[ec2-user] # yum update -y
```

```
# yum install git -y
# git config -global username

# git config --global user.name "Alok"

# git config --global user.email alok.anupam26@gmail.com
```

User.name=Alok

User.email= alok.anupam26@gmail.com

```
[Ec2-user] # mkdir singaporegit
```

# cd singaporegit

```
[singaporgit]# git init

Initialized empty git repository in
Home/ec2user/singaporegit/.git/

[singaporgit]# ls -a
```

. .. .git

```
[singaporgit]# git remote add origin
 https://github.com/ashok1012/centralgit.git
```

Now local repo has been connected to central repo, for Pulling data to central repo, execute this command

`[singaporgit]# git pull -u origin master` (you can execute without **-u** as well)

Now you can see it has pulled all data/code from remote directory central repo, all details and commits has been done by other Mumbai machine.

`[singaporgit]# cat >mumbai1` (> used to write and overwrite code inside mumbai1) I love Bengaluru too **Ctrl+D**

If you want to add lines or something on this code inside the file use command **# cat >>file**

# git status

```
Modified: mumbai1
```

# git add .

# git status

Modified: mumbai1

```
# git commit -m “first commit from singapore”
```

```
# git log
```

Now it will show all messages **commits ids** and steps done by both Mumbai and Singapore machines

```
# git show 12345678KD458F4lW3E4
```

I Love India Old commit

I Love Bengaluru new commit

Push data/code into central git from local repo

```
# git push -u origin master
```
(you can use **-f** instead of -u for **force** push)

Now Enter username and password of git hub account, after that you will see all new and old **commits updates** in central git, click **mumbai1** file you will get code “I Love Bengaluru

GITIGNORE-This command is used to ignore some specific file which we don’t want to add & commit.

```
[mumbaigi1]# vi .gitignore


* CSS * used to ignore particular file

* java Esc+:wq

# git add .gitignore

# git commit -m “ignore file format”
```
can use single comma as well

```
# git status
```

Nothing to commit, working tree is clean, now create some files in different formats by using **touch command**

```
# touch file1.txt file4.java file3.css file5.java file2.txt

# ls
# git status
```

`File1.txt` only showing 2 untracked files rest three have been ignored

```
File2.txt
```

```
# git add .

# git status
```

Now both files have been added and showing us in green color after status command

```
File1.txt

File2.txt
```

```
# git commit -m "IGNORE JAVA CSS FILES"

# git log
```

`12345678KD458F4lW3E4` (HEAD -> master) So many commit Ids are showing
`12345678KD458F4lW3E4` (HEAD -> master)

`12345678KD458F4lW3E4` (HEAD -> master)

```
# git show 12345678KD458F4lW3E4

IGNORE JAVA CSS FILES
```

```
# touch ashok.java

# git status
```

Nothing to commit, working tree is clean, now create some files in different formats by using **touch command**

```
# touch Alok.txt

# git status
```

`Alok.txt` (Again it showed text file, not java file means ignored)

If I want to latest commit, last 2 commits, last-n commits and all commits in one line.

```
# git log -1

# git log -2
# git log -oneline
```

12345678KD458F4lW3E4 (HEAD -> **master)** message “1” So many commits are showing in one column

12345678KD458F4lW3E4 message “2”

12345678KD458F4lW3E4 message “3”

If I want to find specific commit, Acton and file use grep command with specific name rest will be ignored.

```
[mumbaigi1] # git log --grep “ignore”
 ignore=ignore css and java
files
```

GIT BRANCHES:

- • Each task has one separate branches, after done with code other branches merge with master.
-
- • This concept is useful for parallel development. Master branch is default branch
-
- • We make branches, one for little features and other one for longer running features.
-
- • It allows keeps the main master branch free from error.
-
- • Files created in workspace will be visible in any of the branch workspace, until you commit,
- once
- • you commit then those files belong to that particular branch
-

How to create Branches:

```
[ec2-user] # cd mumbaigit

[mumbaigit] # git log --oneline
```

# git branch

```
*master

# git branch branch1

# git branch

*master

Branch1

# git checkout branch1  (switch to branch branch1)

master

*Branch1

# git branch -d <branchname>  (to delete any branch)
```

Branches Working process:

```
# git checkout branch1

# cat >branch1file  (create branch1file and write anything inside by >)

Something is better that Nothing

# ls

mumbai1

# git checkout master

# ls

mumbai1 branch1
```

branch1file and code is **showing** inside master branch because it hasn't committed with any branch yet.

```
# git commit -m "branch1 first commit"

# git log  –oneline

Branch1 first commit

# git checkout master
```

# git log –oneline

branch1file & code will **not show** inside master branch because that file has been committed with Branch1.

How to Merge Branches: we use pulling mechanism, we can't merge branches of different repositories

# git checkout master

git merge branch**A** (to verify the merge)

Executed checkout command before merge command means, you wanted to merge any branch with master branch

# git log –oneleine

Now you can see **All commits** of both branches which have been merged together

ls

Now you can see **All files** of both branches which have been merged together.

git **push** origin **master** (to push central repo lit git hub)

Enter username & password you can see merged data in central repository on git hub.

GIT

GIT CONFLICT: When same files having different content in different branches, if you do merge ,conflict can occur. (Resolve conflict then add and commit)

# Cat >**anupfile**

hello anup ctrl+d

# git add .

# git commit -m "commit before conflict"

# git **checkout** branch1 switch to branch1

# Cat >**anupfile** create same file but write different code inside

hello Ashok ctrl+d

```
# git commit -m “commit from branch1”
```

```
# git checkout master switch to branch1
```

```
# git merge branch1
```

Merge failed: fix conflict, then commit result

```
# vi anupfile (update inside anupfile)
```

```
<<<<<<HEAD delete HEAD

Hello anup

============ delete =====
Hello Ashok

>>>>>> branch1 Esc+:wq
```

You can change data according to yourself which you exactly needed before conflict do changes in file git will understand the change and execute data accordingly.

```
# git status
```

```
# git add .
```

```
# git commit -m “Resolve conflict”
```

```
# git log  --oneline
```

```
12h3a8g90  (HEAD ❼   master) Resolve conflict
```

GIT BRANCH STASH: If your code is in progress and suddenly need changes through client escalation you have to **keep aside** current code and have to work on new features for some hours.
You can’t commit your parallel code so you need some temporary storage to store partial changes and later on commit it. To stash an item only applied for **modifies files** not new files.

```
# git checkout master
```

```
# cat >anupamfile1
# git commit -m "anupamfile1 commit"

# vi anupamfile1
```

Boss asks to do some other work

```
# git stash

# Cat anupamfile1  (anupamfile1 empty, data stashed ,now you can do new work)

# Git stash list

Stash (0) : WIP on master 1372ee7 .anupamfile1

# vi anupamfile1

My super anupam code-2  Esc+:wq

# cat anupamfile1

My super anupamfil2 code-2

# git stash

# git stash list

Stash (0)

Stash (1)

# cat anupamfile1  (anupamfil1 empty, data/code has been stashed)
```

Now going to do old pending work

```
# git stash apply stash@{1}

# cat anupamfile1

My super anupam code-2

# git add.

# git commit -m "anupamfile1 commit done"

# git stash apply stash@{0}
```

Auto merging anupamfile1; CONFLICT:Merge **conflict** in **# Vi anupamfile1 (update inside anupamfile1)**
<<<<< update stream

My super anupam-1 final code would be my super anupam-2

=================== delete ========

My super anupamcode-2

>>>> stashed changes Esc+:wq

# git add .

# git commit -m "anupamfile1 commit done2"

# git status (empty)

# git log --oneline

anupamfile commit

anupamfile xommit done

anupamfile commit done2

# git stash list

Stash (0)

Stash (1) (still available in stash list delete it by **# git stash clear**, recheck by **stash list** command)

GIT RESET: It is a powerful command that is used to undo local changes to the state of a git repository.

It used to undo the add . command.

# cat >testfile

Hello Ashok

# git add .

# git status

New file:testfile (now realized did mistake in data wanted to change)

- • **To reset from staging aria**

# git reset testfile

# git reset . (removed data from staging aria)

# git status

testfile

# git add .

# git status

testfile

- • To reset from staging area

# git reset –hard

# git status

```
One branch master nothing to commit: working tree clean
```

GIT REVERT:

Revert command helps you to undo the existing commit, it doesn’t delete any data instead rather git creates a new commit with the included previous files reverted to the previous stat.
So, history moves forward while the state of your file moves backward.

# cat >checkfile

I LOVE MY INDIA

# git add.

# git commit -m “checkfile1 commit” after commit I realized did wrong commit

# git log –oneline

Now you can see so many commits copy previous commit id just before the mistake and paste on revert command

```
# git revert 12h3a8g90
```

Wrong commit **undo** state moves to backward also write a message in this commit “please ignore previous commit”

How to remove untracked files

```
# git clean -n dry run
```

```
# git clean -f forcefully
```

Git Tags: Tag operation allows giving meaningful name to a specific version in the repository.

To Apply Tag # git tag -a<tag-name> -m “message” commit-id

```
# git tag -a Ashok -m "love you India" 12h3a8g90g6k
```

To see tag **# git tag**

# git show tag-name to see **particular commit** content by using Tag

To delete a tag **# git tag -d** tag-name

Git Hub Clone: go to existing repo in git Hub copy the URL of central repo and paste with run command of Linux machine.

```
# git clone <URL git hub repo>
```

```
[ec2-user] # git clone
 https://github.com/ashok1012/centralgit.git
```

It creates a **local repo** automatically inside Linux machine with the same name of git hub account.

```
# ls
```

```
Mumbai1git Anupamfile centralgit node1git
```

Both repositories can be connected together easily by master branch