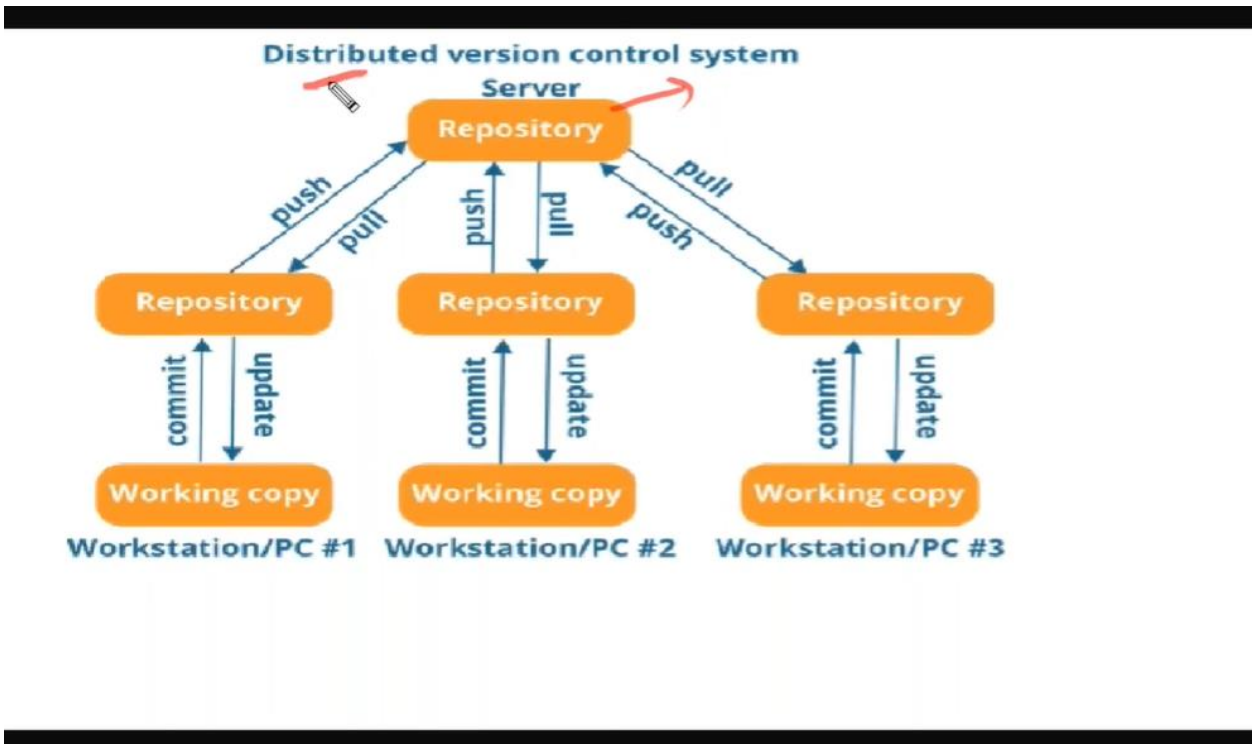
Version control system / Source code management system :- It is the system which track your changes, maintain proper log of your changes, which allow multiple developers to work simultaneously.

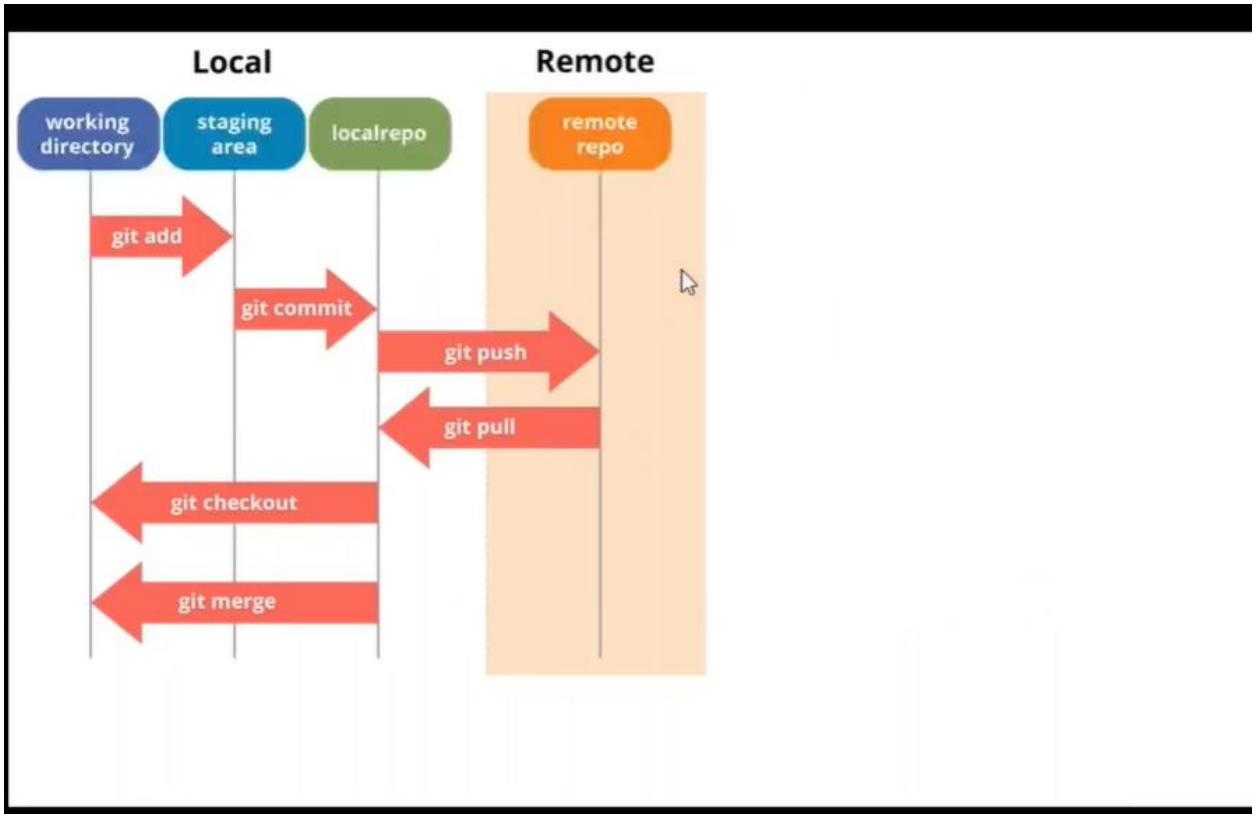
Types of Version control system:-

1. Local version control system
2. Centralized version control system (ex:- SVN)
3. Destributed version control system (ex:- git)

Architecture of Version control system :-



Architecture of git:-



Notes:-

1. Staging area is a virtual layer between working directory and local repo.
2. Working directory and local repo are physically same but logically different

Ways to create local repository :-

1. Get copy from remote repo
2. Manually create local repo

LABS :-

1. Install git
2. Go to specified folder location where you want to store the project
3. Initialize the git :- **git init**
   The above command will create .git folder inside your directory
4. Create some files (ex:- index1.html)
5. **git status**
   Note :- If status shows as untracked, this means files are in working directory but not added to git(i.e in staging area)
6. **git add file_name** (ex:- git add index1.html)

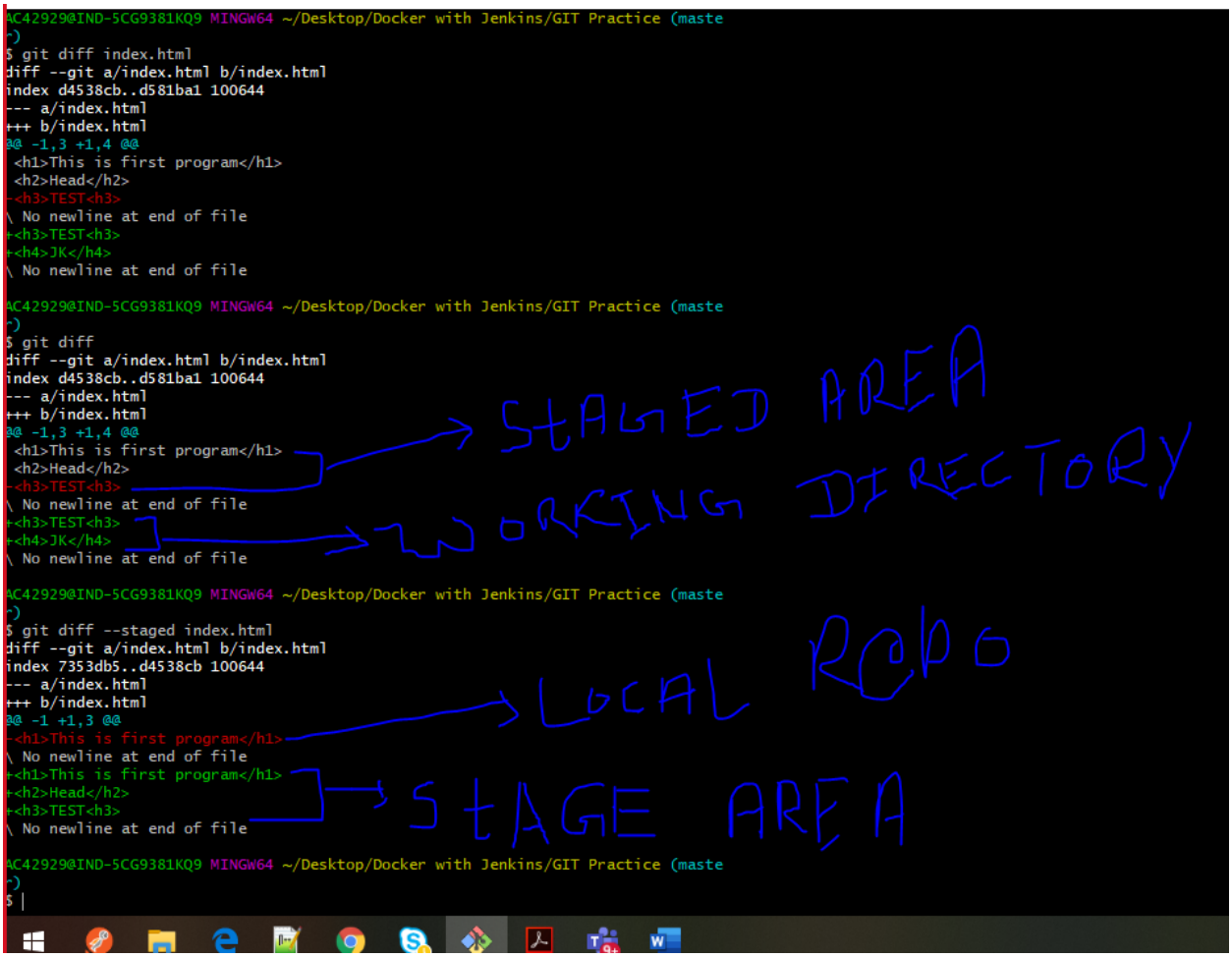
Note :- the above command will add your file to staging area

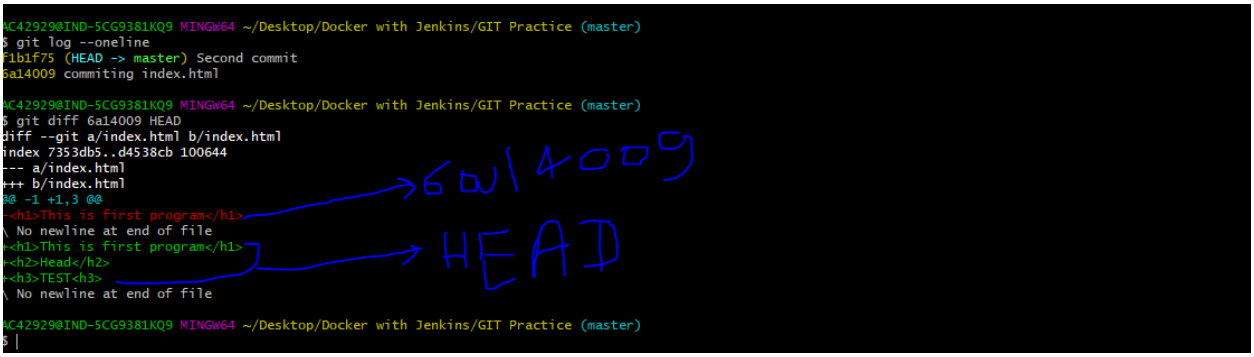
7. **git status**
   Note :- If status shows as Changes to be committed, this means files are in staging area but not committed to local repository.
8. **git commit -m "some commit message "** (i.e git commit -m "First Commit ")
   Note :- this command commit the file to our local repository
9. If it asks for credential, execute below two command otherwise you can ignore
   git config --global user.name "FIRST_NAME LAST_NAME"
   git config --global user.email "MY_NAME@example.com"
10. **git status**
    Note :- If status shows as nothing to commit, working tree clean, this means files are committed to local repository.
11. To see the files in local repository, execute below command
    git ls-files
12. To see the history of command already executed use below command
    history
13. To see the commit history, execute the below command
    git log
14. To see only commitId and message, execute below command
    git log --oneline
15. If you want to see any particular commit, execute below command
    git show commitId (ex:- git show 1def101)
16. Change some file contents which are already committed and execute **git status**
    Note :- It show status as Changes not staged for commit. Again execute git add file_name and **git commit -m "message"** or **git commit -a -m "message"**
17. To see the unstaged changes (you have modified the file but not added to git, it will compare working directory file with staged file) execute the below command
    git diff file_name (ex:- git diff file1.txt)
18. To see the staged changes (you have modified the file ,added to git, it will compare staged file with local repository) execute the below command
    git diff --staged file_name (ex:- git diff --staged file1.txt)

- **git command to compare files with different scenario**

  - To compare the working directory vs last staged area changes
    - $ git difftool
  - Compare working directory vs local git (last commit in current branch)
    - $ git difftool HEAD
  - Compare staged area vs local git repo last commit
    - $ git difftool --staged HEAD
  - For limiting file to choose one file out of many file (working area vs staging area)
    - $ git difftool - -filename

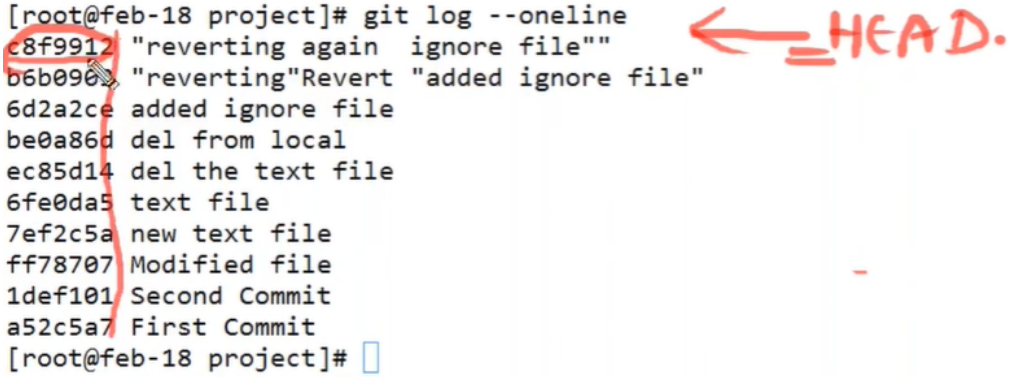
- To see the commit log
  - $ git log - -oneline
- To see differences between any of the two commits.
  - $ git difftool commitID1 HEAD (HEAD means last commit in current branch)
  - $ git difftool HEAD HEAD^ (HEAD vs HEAD^ means last commit in current branch vs 2nd last commit in current branch)
  - $ git difftool commitId1 commitId2 (compare commit with commitId1 vs commitId2)
- Compare from central(origin/master) to local git(master)
  - $ git difftool master origin/master





19. To remove the file from Working directory as well as from Local repository, execute below command
    git rm file_name (ex :- git rm file1.txt)
20. To remove the file from Local repository but not from Working directory, execute below command
    git rm --cached file_name (ex:- git rm --cached index.html)
    Note:- Here we have deleted file from local repo but its still available in working directory, so when you will do **git status** , it will show the status as Untracked files.
    To solve this create one file **.gitignore** and write the deleted file name or the file name which you want to ignore inside .gitignore

21. To revert the changes, execute below command
    git revert commitId
22. Git works with pointers, HEAD is the pointer in git which always points the top most commit in the log history



Now in above picture I want my HEAD should point 1def101 Second Commit, by doing so we will lose all the log history and all changes above that line. We can achieve this by doing reset at commit level. Execute the below command to do this

git reset --hard commitId (ex:- git reset --hard 1def101)

Note:- After executing above command by any means we can't revert the changes. It is called as destructive command. This command will not generate any commitId

23. Normally for different requirement or for different releases we maintain different branches. Default branch is master. To check the list of branch, execute the below command
    git branch
24. To create a new branch execute the below command
    git checkout -b new_branch_name old_branch_name
25. To merge the code from one branch to other branch execute the below command
    git merge source_branch_name destination_branch_name
    ex:- git merge NEW_BRANCH master
    here from NEW_BRANCH we are merging to master
    source_branch_name :- from where you want to merge
    destination_branch_name :- on which branch you want to merge
26. **Resolve Merge Conflicts :-**
    Suppose we have two branch master and B1
    We have created index4.html in master branch
    <h1>This is index4 in master</h1>
    We have added this file and committed to local repo.
    Again we switched to branch B1
    We have created index4.html in B1 branch
    <h1>This is index4 in B1</h1>
    We have added this file and committed to local repo.
    Again we switched to branch master and we tried to merge B1 into master, In such cases it shows conflict because both have different contents at same line

```
AC42929@IND-5CG9381KQ9 MINGW64 ~/Desktop/Docker with Jenkins/GIT Practice (maste
r)
$ git merge B1
CONFLICT (add/add): Merge conflict in index4.html
Auto-merging index4.html
Automatic merge failed; fix conflicts and then commit the result.
```

    We we open the file index4.html from master it will look like this,

```
<<<<<<< HEAD
<h1>This is index4 in master</h1>
=======
<h1>This is index4 in B1</h1>
>>>>>>> B1
```

    Head means master branch, so above line is from master branch and below line is from B1 branch. Now you can resolve this conflict and commit the file. After committing again if you try to merge, it will show Already up to date.
27. **git stash :-**
    Suppose we are in master branch, modifying index1.html and index2.html simultaneously. Suddenly requirement came so that we have to commit the files, but our files is not complete so we can't commit to local repo and also we don't want to lose these files

because again we have to complete this. In such cases satsh concept comes into picture. Stash command move the file from working directory to some virtual/temporary shelves . After stash, your working directory will be cleaned, so you can commit the files and do whatever you want. Again after completing your work if you unstash the changes then your stashed changes will be available in your working directory.
Command :-
git stash
git stash list
git show stash_number (ex:- git show stash@{0})
git stash pop stash_number (ex:- git stash pop stash@{0}) :- this is the command for unstash the changes
git stash clear: - command to clear and delete all the stash
git stash drop stash_number :- to delete particulat stash
git stash -p: - to do partial stash

28. Suppose we are working in master branch, currently we have two commit in this CommitA, CommitB, at this point we have cretated another branch B2 from this so it will already have CommitA and CommitB , on top of that we have again Committed CommitD in B2 branch. Again Switched to master branch and done one Commit called CommitC.
    At this point following are the commits for different branch:-
    master :- CommitA, CommitB, CommitC <- HEAD
    B2 :- CommitA, CommitB, CommitD <- HEAD
    Now if we merge master to B2, commit history will look like this and Head will point to CommitC
    B2:- CommitA, CommitB, CommitD , CommitC<- HEAD
    But we don't want this, We want CommitC before CommitD i.e
    B2:- CommitA, CommitB, CommitC , CommitD<- HEAD
    We can achieve by using concept called Rebase.
    git rebase branch_name (suppose you are in branch B2 and want to rebase from master the the command should be git rebase master)
    Note:- Rebase can be done only from the parent branch
29. Push your project to git:-
    A. Login to git
    B. Create a repository (i.e newrepo1)
    C. Get the link of remote repo (i.e https://github.com/jitendrakr93/newrepo1.git)
    D. Now in command prompt execute below command
       git remote add origin git_link (i.e git remote add origin https://github.com/jitendrakr93/newrepo1.git)
    E. **git push origin branch_name** (i.e git push origin master)
    F. execute same push command for all other branches
30. To delete the branch from local execute the below command
    git branch -d branch_name
31. To delete the branch from remote execute the below command
    git push origin --delete branch_name
32. To create local repo by getting repo from remote, execute below command
    git clone repo_url
33. To update local repo with cloned remote repo execute below command

git pull origin branch_name

34. To Check the changes local repo with cloned remote repo execute below command
    git fetch origin branch_name
    Note :- pull command download file from remote and merge it to local repo. While fetch command only show the differences but will not download