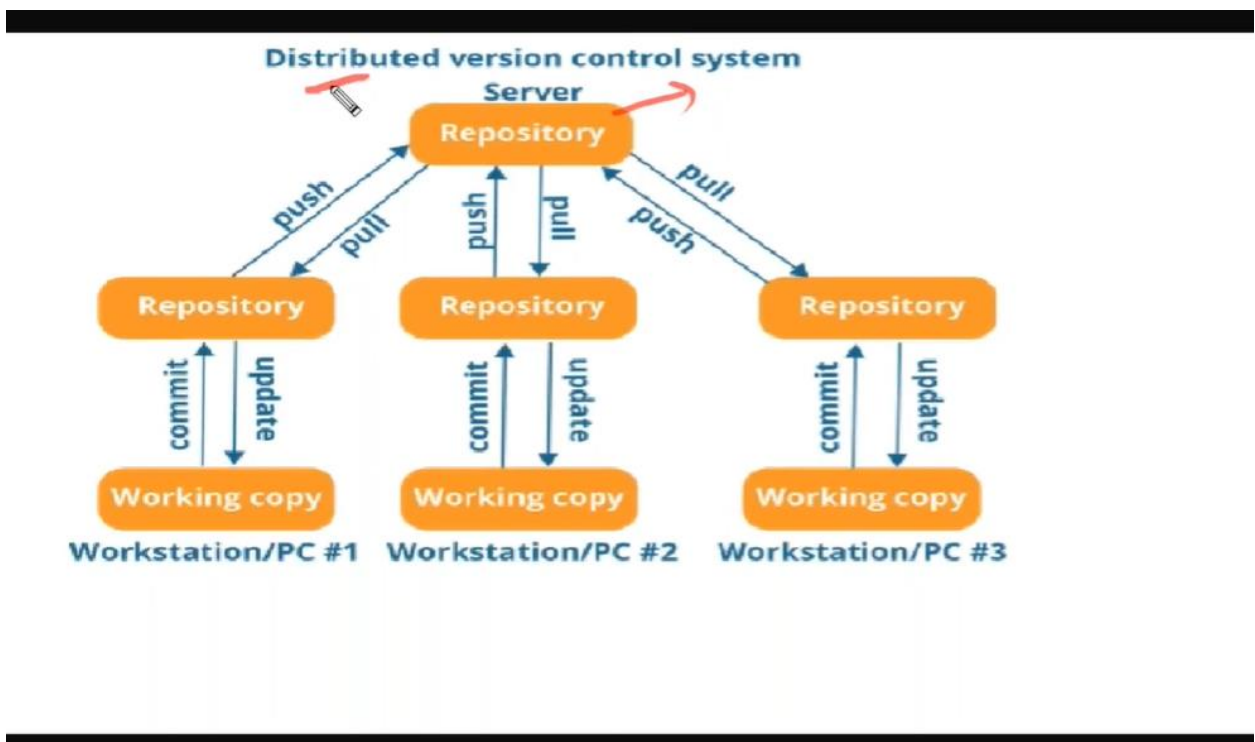


Version control system / Source code management system :- It is the system which track your changes, maintain proper log of your changes, which allow multiple developers to work simultaneously.

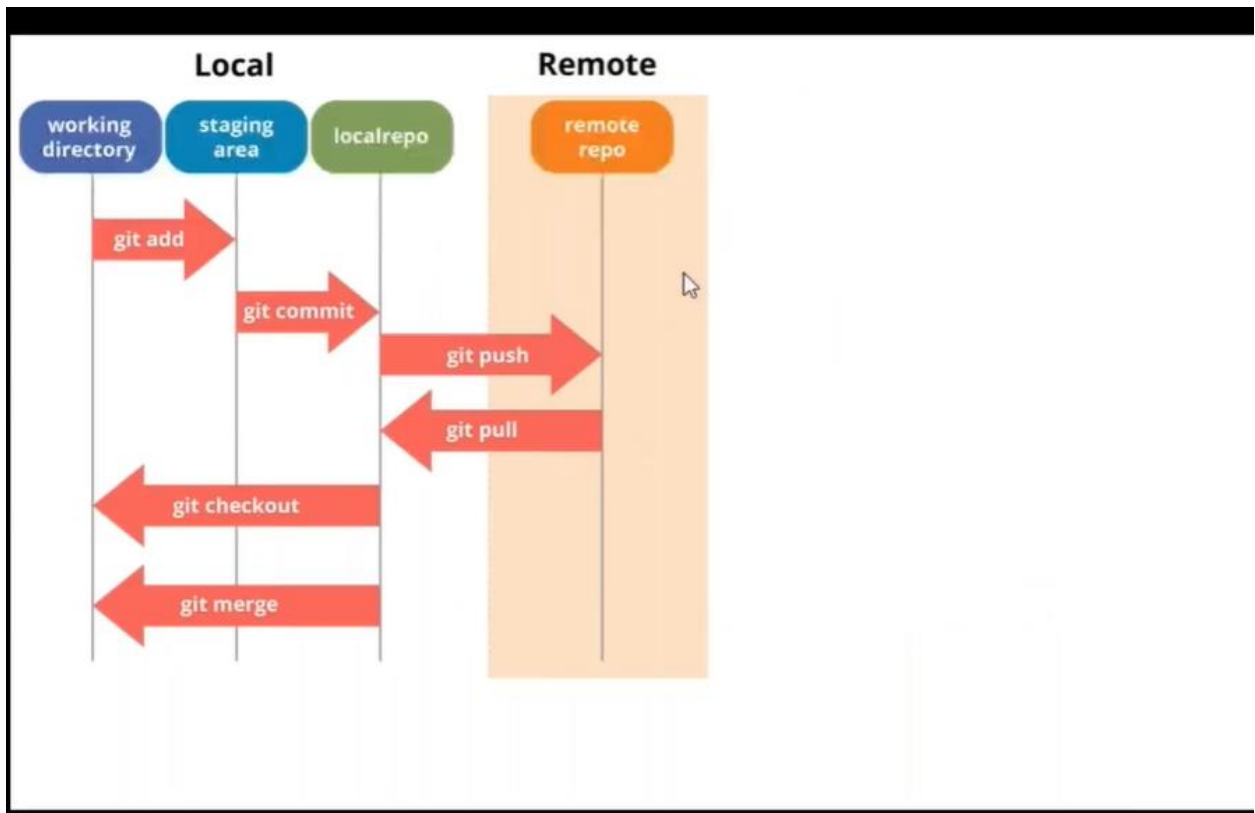
Types of Version control system:-

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 diff tool commitID1 HEAD (HEAD means last commit in current branch)
 - \$ git diff tool HEAD HEAD^ (HEAD vs HEAD^ means last commit in current branch vs 2nd last commit in current branch)
 - \$ git diff tool commitId1 commitId2 (compare commit with commitId1 vs commitId2)
- Compare from central(origin/master) to local git(master)
 - \$ git diff tool master origin/master

```

AC42929@IND-5CG9381KQ9 MINGW64 ~/Desktop/Docker with Jenkins/GIT Practice (maste
-)
$ git diff index.html
diff --git a/index.html b/index.html
index d4538cb..d581ba1 100644
--- a/index.html
+++ b/index.html
@@ -1,3 +1,4 @@
<h1>This is first program</h1>
<h2>Head</h2>
-<h3>TEST<h3>
\ No newline at end of file
+<h3>TEST<h3>
+<h4>JK</h4>
\ No newline at end of file

AC42929@IND-5CG9381KQ9 MINGW64 ~/Desktop/Docker with Jenkins/GIT Practice (maste
-)
$ git diff
diff --git a/index.html b/index.html
index d4538cb..d581ba1 100644
--- a/index.html
+++ b/index.html
@@ -1,3 +1,4 @@
<h1>This is first program</h1>
<h2>Head</h2>
-<h3>TEST<h3>
\ No newline at end of file
+<h3>TEST<h3>
+<h4>JK</h4>
\ No newline at end of file

AC42929@IND-5CG9381KQ9 MINGW64 ~/Desktop/Docker with Jenkins/GIT Practice (maste
-)
$ git diff --staged index.html
diff --git a/index.html b/index.html
index 7353db5..d4538cb 100644
--- a/index.html
+++ b/index.html
@@ -1 +1,3 @@
-<h1>This is first program</h1>
\ No newline at end of file
+<h1>This is first program</h1>
+<h2>Head</h2>
+<h3>TEST<h3>
\ No newline at end of file

```

STAGED AREA
WORKING DIRECTORY
LOCAL REPO
STAGE AREA

```

MC42929@IND-5CG9381KQ9 MINGW64 ~/Desktop/Docker with Jenkins/GIT Practice (master)
$ git log --oneline
1b1f75 (HEAD -> master) Second commit
6a14009 committing index.html

MC42929@IND-5CG9381KQ9 MINGW64 ~/Desktop/Docker with Jenkins/GIT Practice (master)
$ git diff 6a14009 HEAD
diff --git a/index.html b/index.html
index 7353db5..d4538cb 100644
--- a/index.html
+++ b/index.html
@@ -1,3 @@
-<h1>This is first program</h1>
\ No newline at end of file
-<h1>This is first program</h1>
-<h2>Head</h2>
-<h3>TEST</h3>
\ No newline at end of file

```

6a14009
HEAD

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

```

[root@feb-18 project]# git log --oneline
c8f9912 "reverting again ignore file"
b6b0962 "reverting"Revert "added ignore file"
6d2a2ce added ignore file
be0a86d del from local
ec85d14 del the text file
6fe0da5 text file
7ef2c5a new text file
ff78707 Modified file
1def101 Second Commit
a52c5a7 First Commit
[root@feb-18 project]#

```

← HEAD.

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