

**Every version we have to maintain.**

**1) Maintaining multiple versions manually is very complex activity.**

**2) Dev-A and Dev-B working on the code. At last we have to merge the code developed**

**by both developers and we have to deliver to the client. If both developers developed**

**a file named with Util.java, then one copy will overwrite with another copy, which**

**creates abnormal behaviour. We should not overwrite our code**.

**Every change should be tracked like**

**who did the change**

**when he did the change**

**which changes he did etc**

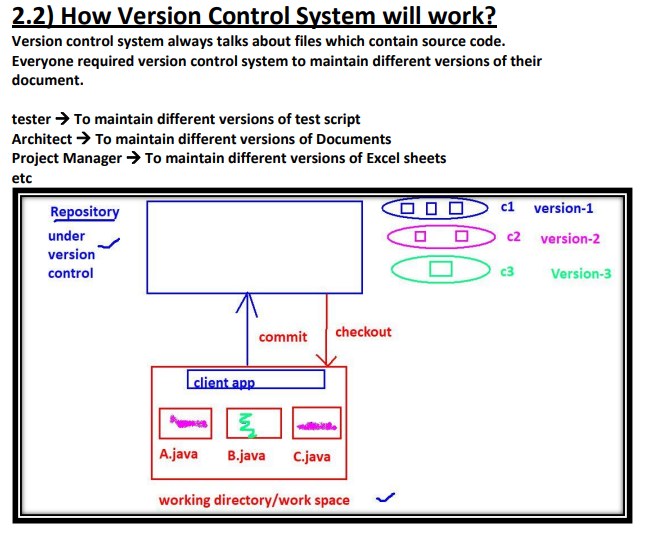
**and all changes should be maintained.**

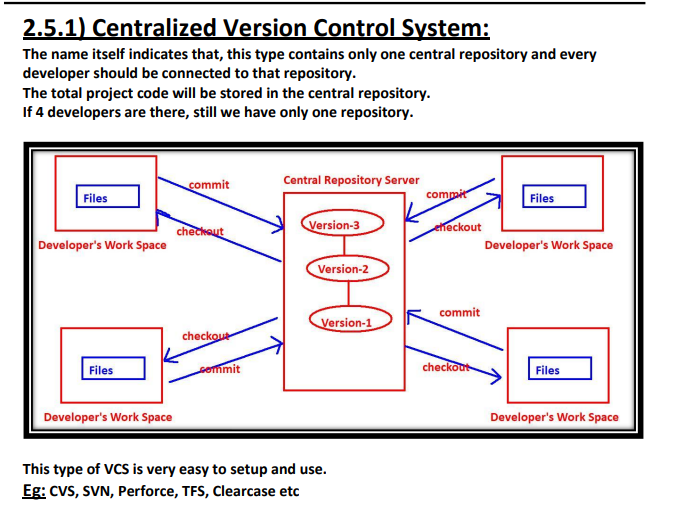
**3) Overwriting of the code should not be happend.**

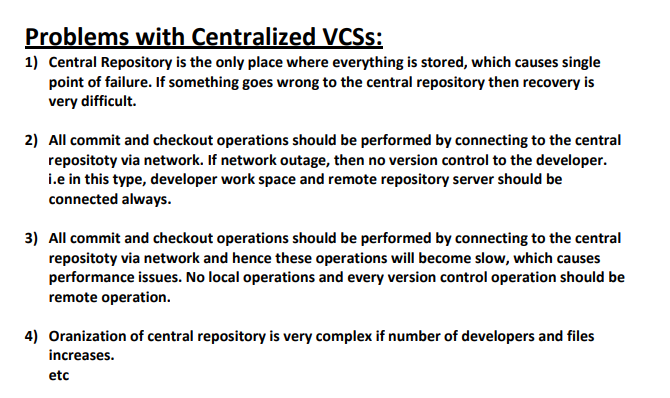
**4) Developers have to share their code to peer developers, so that multiple developers**

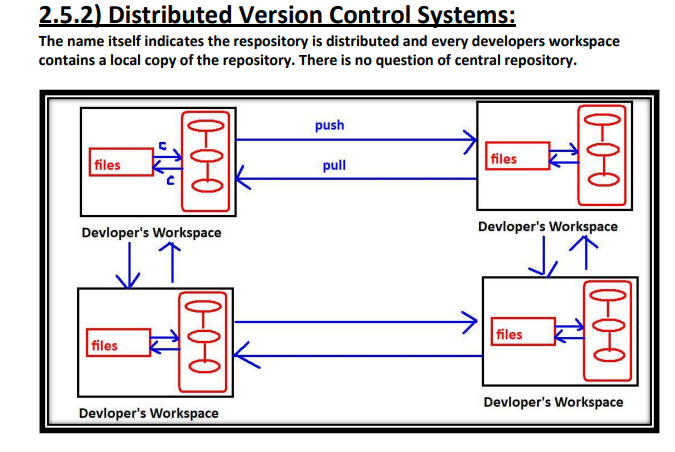
**will work in collaborative way.**

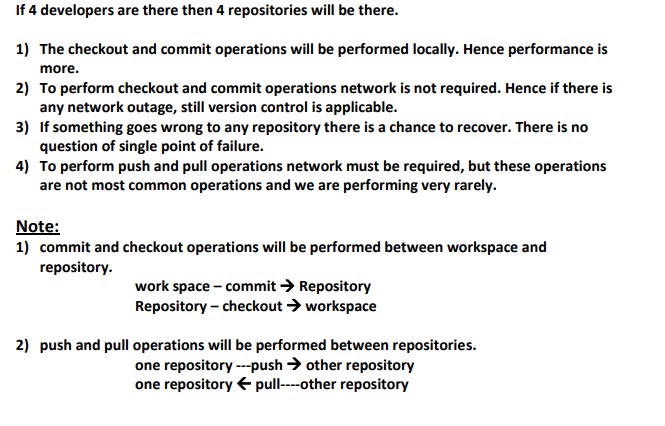
**5) Parallel development must be required**

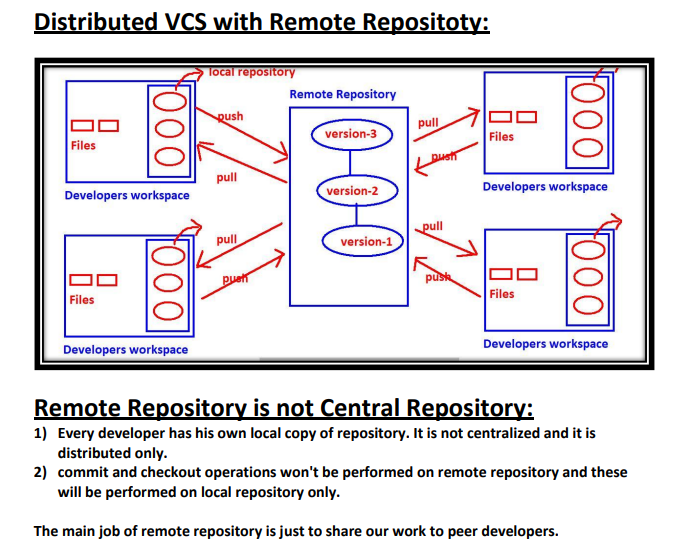












**High availability, Speed and there is no single point of failure are main reasons for**

**popularity of this model.**

**Eg: Git, Mercurial, Fossil**

**3.1) What is GIT?:**

**✽ Git is Distributed Version Control System Tool.**

**✽ Git is not acronym and hence no expansion.But most of the people abbreviated as**

**✽ "Global Information Tracker".**

**✽ GIT is developed by Linus Torvalds(Finnish software engineer), who also developed**

**Linux Kenel.**

**✽ Most of the companies like Microsoft,Facebook,Yahoo,LinkedIn,Intel using Git as**

**Version Control System Tool.**

**3.2) Features of GIT:**

**Git is very popular because of the following features:**

**1) Distributed**

**Git is developed based on Distributed Version Control System Architecture.**

**Because of Distributed Architecture it has several advantages:**

**A) Every Developer has his own local repository. All the operations can be performed**

**locally.Hence local repo and remote repo need not be connected always.**

**B) All operations will be performed locally, and hence peformance is high when compared**

**with other VCSs. i.e it is very speed**

**C) Most of operations are local. Hence we can work offline most of the times.**

**D) There is no single point failure as Every Developer has his own local repository.**

**E) It enables parellel development & automatic-backups**

**2) Staging Area:**

**It is also known as index area.**

**There is logical layer/virtual layer in git between working directory and local repository.**

**Working Directory  Staging Area  Local Repository**

**We cannot commit the files of working directory directly. First we have to add to the**

**staging area and then we have to commit.**

**This staging area is helpful to double check/cross-check our changes before commit.**

**This type of layer is not available in other Version Control Systsem Tools like CVS, SVN etc**

**Git stores files in repository in some hash form, which saves space.**

**GIT will uses internally snapshot mechanism for this. All these conversions and taking**

**snapshots of our data will be happened in staging area before commit.**

**Eg: If a sample repository takes around 12 GB space in SVN where as in GIT it takes hardly**

**420 MB.**

**Hence Staging Area is the most important Strength of GIT.3) Branching and Merging:**

**We can create and work on multiple branches simultaneously and all these are branches**

**are isolated from each other. It enables multiple work flows.**

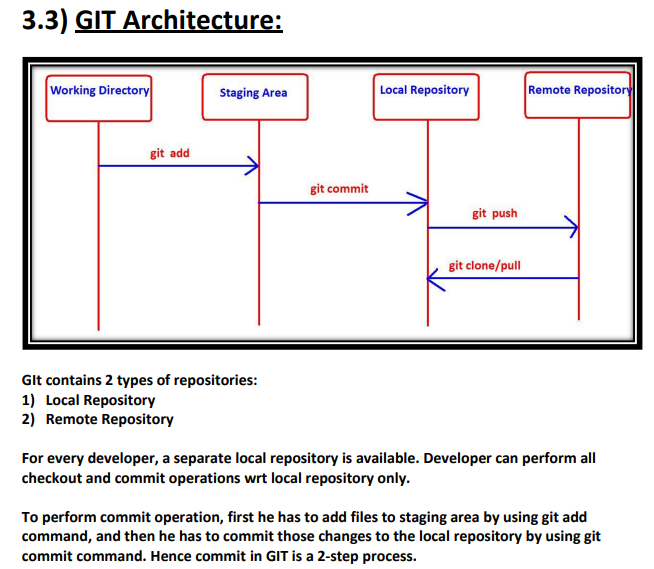
**We can merge multiple braches into a single brach. We can commit branch wise also.**

**4. Moving files in GIT is very easy as GIT automatically tracks the moves. Whereas in other**

**VCS we need to create a new file & then delete the old one.**

**5. Freeware and Open Source**

**6. It provides support for multiple platforms.**

****

**commit is applicable only for staging area files but not for working directory files.**

**If the developer wants to share his work to the peer developers then he has to push his**

**local repository to the remote repository by using git push command.**

**Remote repository contains total project code, which can be accessible by all developers.**

**New developer can get local repository by cloning remote repository.For this we have to**

**use git clone command.**

**A developer can get updates from the remote repository to the local repository by using**

**git pull command.**

**git add 🡺 To add files from working directory to staging area.**

**git commit 🡺 To commit changes from staging area to local repository.**

**git push 🡺To move files from local repository to remote repository.**

**git clone 🡺To create a new local repository from the remote repository.**

**git pull 🡺 To get updated files from remote repository to local repository.**

**1)Untracked:**

**The files which are newly created in working directory and git does not aware of these**

**files are said to be in untracked state.**

**2)Staged:**

**✽ The files which are added to staging area are said to be in staged state.**

**✽ These files are ready for commit.**

**3)In Repository/ Committed:**

**Any file which is committed is said to be In Repository/Committed State.**

**4)Modified:**

**Any file which is already tracked by git, but it is modified in working directory is said to**

**be in Modified State.**

**1)git init**

**Once we creates workspace, if we want version control, then we require a local**

**repository. To create that local repository we have to use git init command.**

**$ git init**

**Initialized empty Git repository in D:/gitprojects/project1/.git/**

**.git is an empty repository, which is hidden directory.**

**2)git status:**

**It shows the current status of all files in each area, like which files are untracked, which**

**are modified, which are staged etc.**

**$ 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: a.txt**

**modified: b.txt**

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

**Note: We can get concise information by using -s option**

**3)git add:**

**To add files from working directory to staging area for tracking/commiting purpose, we**

**have to use git add command.**

**i) To add all files present in current working directory**

**git add .**

**ii) To add one or more specified files**

**git add a.txt**

**git add a.txt b.txt**

**iii) Even we can use pattern also**

**git add \*.txt**

**git add \*.java**

**4)git commit:**

**If we want to commit staged changes, then we have to use git commit command.**

**For every commit, a unique commit id will be generated. It is of 40-length hexadecimal**

**string.**

**$ echo -n "df4bb05e36e672698251e05e09d92ba45ea1fc47" | wc -c**

**40**

**The first 7 characters also unique, by using that also we can identify commit.**

**This unique id is considered as hash, which is generated based on content of files.**

**The advantages of this hash are**

**1) Data inside our local repository is more secure.**

**2) git requires less space to store contents of files.**

**(If SVN repository required 12GB, but for same content git requires 420MB)**

**For every commit, git records author name,mail id, timestamp and commit message.**

**We can add files to staging area and we can commit changes by using a single command**

**git commit -a -m "commit message"**

**-a means adding files to staging area**

**-m means commit message**

**But this command will work only for tracked files but not for new files.**

**git commit -a -m "commit message"**

**git commit -am "commit message" Valid**

**git commit -ma "commit message" won't work, because order is important.**

**5)git log:**

**It shows history of all commits.**

**It provides commit id, author name,maild , timestamp and commit message.**

**lenovo@DESKTOP-ECE8V3R MINGW64 /d/gitprojects/project1 (master)**

**$ git log**

**6)git config:**

**We can use this command to configure git like user name, mail id etc**

**git config --global user.email "durgasoftonline@gmail.com"**

**git config --global user.name "Durga"**

**\*\*\*Note:**

**global means these configurations are applicable for all repositories created by git. If we**

**are not using global then it is applicable only for current repository.**

**$ git config --list**

**To list out all git configurations**

**$ git config user.name**

**To display user name**

**commit 9a33a5b2e0d1c90eff544a3710b599be3c22665e**

**Author: Durga <durgasoftonline@gmail.com>**

**Date: Thu May 14 22:16:59 2020 +0530**

**Added two files a.txt and b.txt**

**There are multiple options available with git log command.**

**git log –help**

**Git Log Command**

**8.2) How to see Log Information of a Particular File:**

git log <filename>

git log file1.txt

$ git log file1.txt

**commit 4b77312160c82d76395558da415a96b2a8b36072**

**Author: Durga <durgaadvjava@gmail.com>**

**Date: Thu May 7 21:05:25 2020 +0530**

**This is second commit related to file1.txt**

**commit d49f79120beecb2ea9e34b8398b4ee78bf662bf4**

**Author: Durga <durgaadvjava@gmail.com>**

**Date: Thu May 7 20:52:12 2020 +0530**

**This is my first commit**

**Note: There are multiple options are availble for git log command to see the history.**

**git log --help**

