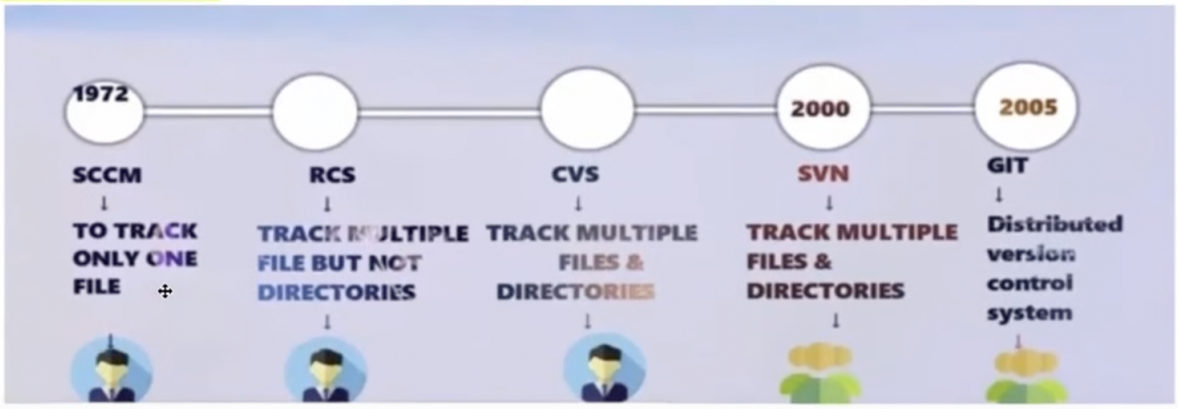
GIT

It is a version control system (VCS) or source code management (SCM).
It is used to track the changes in files.
It will maintain multiple versions of the same file.
It is platform independent.
It is free and open-source.
They can handle larger projects efficiently.
They save time and developers can fetch and create pull requests without switching.

VCS HISTORY



Revision Control System

- is an early version control system (VCS). It is a set of UNIX commands that allow users to develop and maintain program code or documents. With RCS, users can make their own revisions of a document, commit changes, and merge them.
- It will track only Multiple files but not Directories.
- Allowed for single user only.

Concurrent Versions System

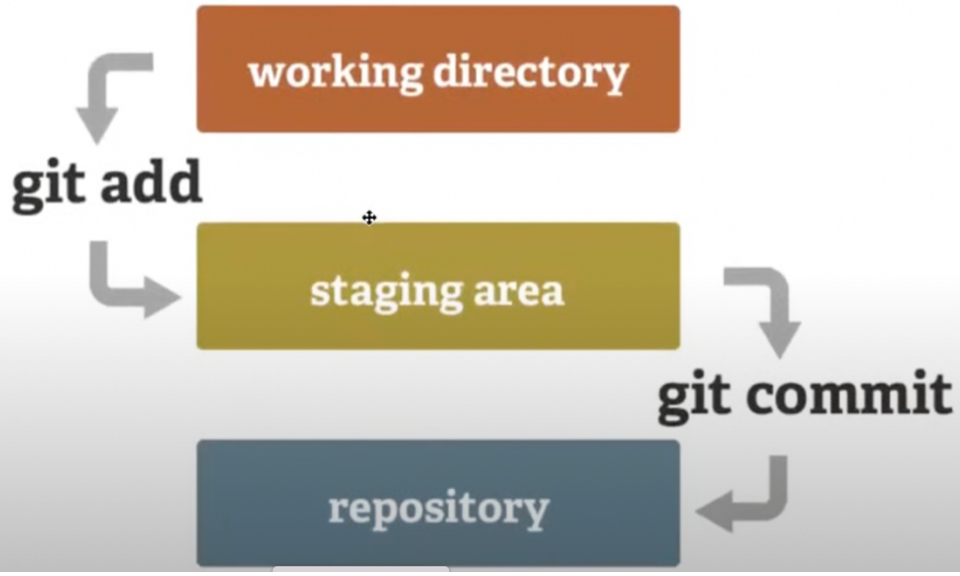
- CVS is a version control system, an important component of Source Configuration Management (SCM). Using it, you can record the history of sources files, and documents. It fills a similar role to the free software RCS, PRCS, and Aegis packages. CVS is a production quality system in wide use around the world, including many free software projects.
- Tracks Multiple files and Directories.
- Allowed for single user only.

Subversion

- SVN is an open-source centralized version control system that is available for everyone at zero cost. It is designed to handle minor to major projects with speed and efficiency. It is developed to coordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace.
- Allowed Multiple users.

FREE	:	GIT, SVN
PAID	:	BITBUCKET, P4, STASH

GIT STAGES



WORKING DIRECTORY

In this stage git is only aware of having files in the project.
It will not track these files until we commit those files.

STAGING AREA

The staging area is like a rough draft space, it's where you can git add the version of a file or multiple files that you want to save in your next commit.
In other words, in the next version of your project.

REPOSITORY

Repository in Git is considered as your project folder.
A repository has all the project-related data.
It contains the collection of the files and also history of changes made to those files.

TYPES OF REPOS

LOCAL REPO: The Local Repository is everything in your .git directory. Mainly what you will see in your Local Repository are all of your checkpoints or commits. It is the area that saves everything (so don't delete it).

CENTRAL REPO: It will be present in the Github where you can share all your files. You need to add and commit your files before you push into Github.

REMOTE REPO : It will be present on remote hosts where you can share all your files to remote machines.

GIT INSTALLATION

```
yum install git -y
git init .
```

STEPS TO COMMIT A FILE

1. Create a file : touch filename
2. Now add that file : git add . (Dot represents current directory)
3. commit the file with message : git commit -m "commit message you want" filenam
4. To see details of that file : git log

Now all those things will be done under root user
if you want to be done by another user or as under your name we need to configure it.

CONFIGURATION OF USER

if you want to give your username and E-mail id to those commits then,

```
git config user.name "username"
git config user.email "userxyz@gmail.com"
```

now give the git log command to see changes, it won't work because after configure we haven't

GIT-HUB

- Github is a web-based platform used for **version control**.
- It simplifies the process of working with other people and makes it easy to collaborate on projects.
- Team members can work on files and easily merge their changes in with the master branch of the project.

Now if you want to push your code to Github
git remote add origin url
git push -u origin branch-name
go to Github and check the files that you have pushed.

GIT MERGE

- If you want to merge branch-1 with branch-2 switch to branch-1 first and give command git merge branch-2
- now that command had merged the content of branch-1 to branch-2
- Whatever the content in branch-1 will be seen in branch-2 now.

GIT FORK

- A fork is **a rough copy of a repository**. Forking a repository allows you to freely test and debug with changes without affecting the original project

GIT BRANCHES

A branch represents an independent line of development.
The git branch command lets you create, list, rename, and delete branches.
The default branch name in Git is master.

- To see current branch : git branch

- To add new branch : git branch branch-name
- To switch branches : git checkout branch-name
- To create and switch at a time : git checkout -b branch-name
- To rename a branch : git branch -m old new
- To clone a specific branch : git clone -b branch-name repo-URL
- To delete a branch : git branch -d <branch>

The -d option will delete the branch only if it has already been pushed and merged with the remote branch. Use -D instead if you want to force the branch to be deleted, even if it hasn't been pushed or merged yet. The branch is now deleted locally.

Now all the things you have done is on your local system.
Now we will go to GIT HUB.

GIT COMMANDS:-

Getting & Creating Projects

Command	Description
git init	Initialize a local Git repository
git clone ssh://git@github.com/[username]/[repository-name].git	Create a local copy of a remote repository

Basic Snapshotting

Command	Description
git status	Check status
git add [file-name.txt]	Add a file to the staging area
git add -A	Add all new and changed files to the staging area
git commit -m "[commit message]"	Commit changes
git rm -r [file-name.txt]	Remove a file (or folder)

Branching & Merging

Command	Description
git branch	List branches (the asterisk denotes the current branch)
git branch -a	List all branches (local and remote)
git branch [branch name]	Create a new branch
git branch -d [branch name]	Delete a branch
git push origin --delete [branch name]	Delete a remote branch
git checkout -b [branch name]	Create a new branch and switch to it
git checkout -b [branch name] origin/[branch name]	Clone a remote branch and switch to it
git branch -m [old branch name] [new branch name]	Rename a local branch
git checkout [branch name]	Switch to a branch
git checkout -	Switch to the branch last checked out
git checkout -- [file-name.txt]	Discard changes to a file
git merge [branch name]	Merge a branch into the active branch
git merge [source branch] [target branch]	Merge a branch into a target branch
git stash	Stash changes in a dirty working directory
git stash clear	Remove all stashed entries

Sharing & Updating Projects

Command	Description
git push origin [branch name]	Push a branch to your remote repository

git push -u origin [branch name]	Push changes to remote repository (and remember the branch)
git push	Push changes to remote repository (remembered branch)
git push origin --delete [branch name]	Delete a remote branch
git pull	Update local repository to the newest commit
git pull origin [branch name]	Pull changes from remote repository
git remote add origin ssh://git@github.com/[username]/[repository-name].git	Add a remote repository
git remote set-url origin ssh://git@github.com/[username]/[repository-name].git	Set a repository's origin branch to SSH

Inspection & Comparison

Command	Description
git log	View changes
git log --summary	View changes (detailed)
git log --oneline	View changes (briefly)
git diff [source branch] [target branch]	Preview changes before merging