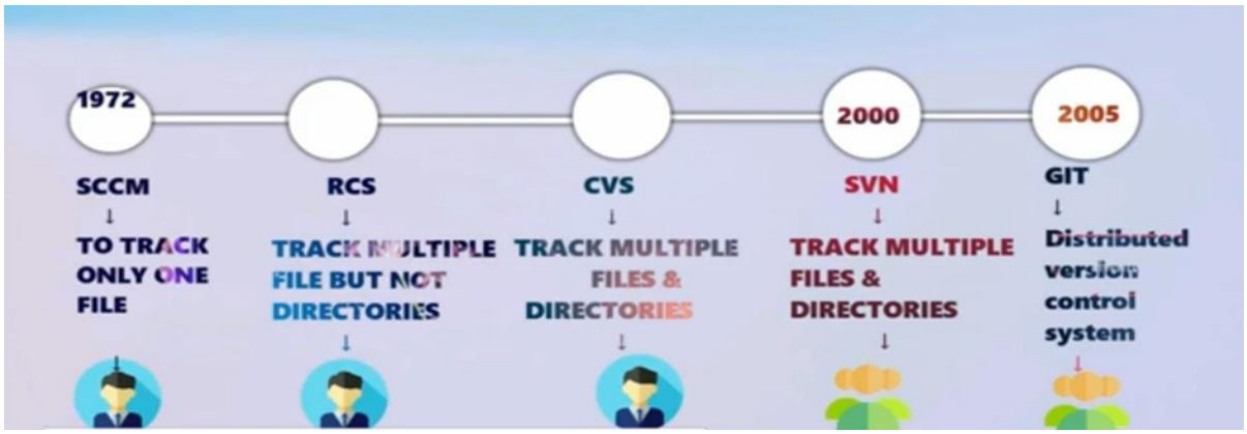
VCS HISTORY

GIT

- It is a version control system (VCS) or source code management (SCM).
- It is used for track the changes in files.
- It will maintain multiple versions of 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.

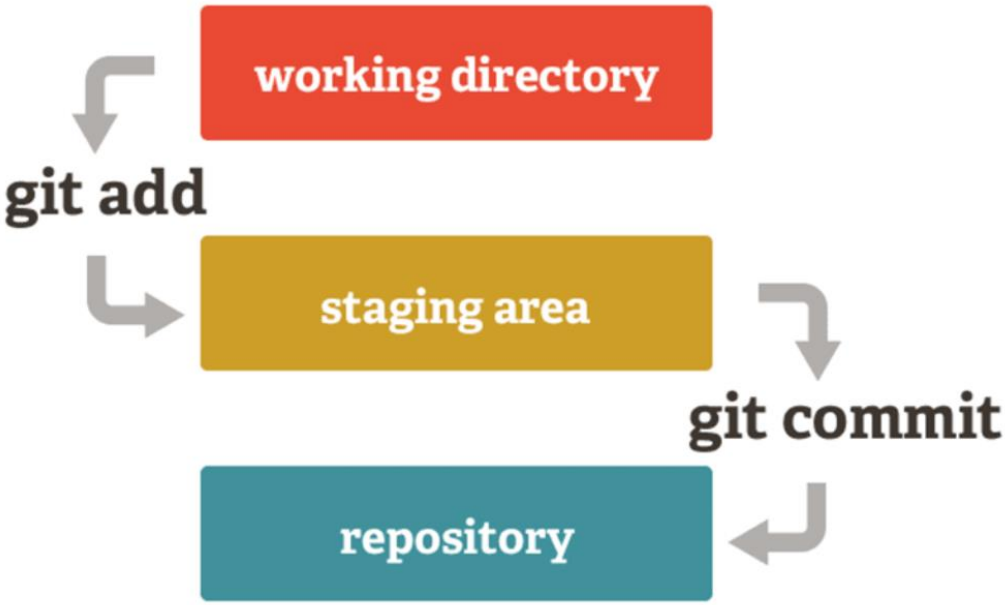
SUBVERSION

- 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 co-ordinate 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

1. **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).

2. **CENTRAL REPO:** The central repo means where you can **store all your source code** and make it available for the others to check it and change it simply its is what we called as **Git-Hub** simply By using Git-Hub anyone can see your code and share with them also.

3. **REMOTE:** Remote Repo means the repository which is located on a remote system. You can send your code to remote repo to anyone who is residing on remote location.

GIT INSTALLATION

- To install the git we use the below commands:
    - yum install git-y
    - git init.
- The git init command is used to **create a new blank repository**. It is used to make an existing project as a Git project.
- The git init command creates a **.git** subdirectory in the current working directory. This newly created subdirectory contains all of the necessary metadata.
- To check the git version we need to use below command: **git-version**

GIT ADD

- The git add command is used to add file contents to the Index his command updates the current content of the working tree to the staging area.

- The git add command can be run many times before making a commit. These all add operations can be put under one commit. The add command adds the files that are specified on command line.

GIT COMMIT

- Every commit contains the index data and the commit message. A commit command is used to **fetch updates from the staging area to the repository**. Commits are the snapshots of the project. Every commit is recorded in the master branch of the repository.

GIT STATUS

- This is used to check the files like which are the staged files and which are the untagged files. Hidden files are shown by default but we can ignore those files.

COMMIT A FILE USING GIT

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" filename
4.To see details of that file	git log

- Now all those things will be done under root user
- If you want to 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 done anything.

Now create a file and commit that file and give git log you will see changes as you configure.

IGNORING CONTENT

It will be useful when you don't want to track some specific files then we use a file called .gitignore

create some text files and create a directory with "jpg" files.

- vi.gitignore
- *.txt

Now all the txt files will be ignored

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 pull 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 REPO'S (PRIVATE & PUBLIC)

GIT PUSH

- This is used to check the files like which are the staged files and which are the untagged files. Hidden files are shown by default but we can ignore those files. For that first we need to create a repo on GitHub and push the content.

GIT PULL

- It is exactly opposite to Git Push we can get all our files from git hub to git.
- git pull origin master.

GIT CLONIG

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 GITHUB.

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 STASH

Using the git stash command, developers can temporarily shave changes made in the working directory. It allows them to quickly switch contexts when they are not quite ready to commit changes. And it allows them to more easily switch between branches.

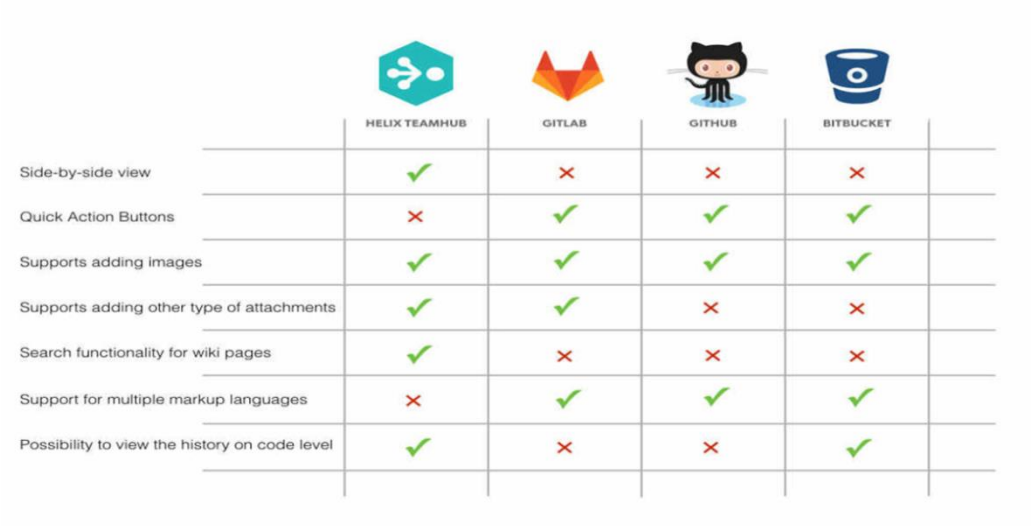
- To see modifications : git stash apply
- To see stash list : git stash list
- To delete stashes : git stash clear

ADVANTAGES:

Speed
Simplicity
Fully Distributed
Excellent support for parallel development, support for hundreds of parallel branches.
Integrity

DISADVANTAGES

Windows support issue.
Entire download of the project history may be impractical and consume more disk space if the project has long history.



INTERVIEW QUESTIONS

Q. What is a GitHub?

A. GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This will teach you GitHub essentials like repositories, branches, commits, and Pull Requests.

Q. How to use GitHub?

A. Step 0: Install git and create a GitHub account....

Step 1: Create a local git repository....

Step 2: Add a new file to the repo.....

Step 3: Add a file to the staging environment....

Step 4: Create a commit....

Step 5: Create a new branch

Step 6: Now push that branch to the repo

Q. Is GitHub open source?

A. Actually Git is not open source. Because it has Commercial offer known as "Git Enterprise Edition" Anyone can apply their ideas like open source on GitHub alike known as Gitlab which is a Ruby Application with its source code here

Q. What is a Branch in GitHub?

A. A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.

Q. What is a git Pull?

A. The git pull command is used to fetch and download content from a remote repository to local.

Q. What is meant by Git repository?

A. Git is a program that tracks changes made to files.... A Git repository is the. git/folder inside a project. This repository tracks all changes made to files in your project, building a history over time.

Q. What is meant by fork in GitHub?

A. Fork is a rough copy of the repository which allows you to test and debug the changes with changes and it does not affect the original code anymore.

Q. What is the difference between a fork and a branch in Git?

A. Branching and forking provide two ways of diverging from the main code line.... So, unlike a branch, a fork is independent from the original repository. If the original repository is deleted, the fork remains. If you fork a repository, you get that repository and all of its branches.

Q. What is Git Fetch?

A. git fetch is the command that tells your local git to retrieve the latest meta-data info from the original.

Q. What is .gitignore file?

A. if you want to ignore some specific siles you can mention where and those files won't be tracked anymore.

Q. What is the difference between Master and Main Branch?

A. when you start extracting files and implementing it automatically you will get the main branch there so there you have to do all the activities merge, pull ... and eventually it all goes to the master branch.

Q. Which generation is the Git comes under?

A. Git comes under 3 rd. generation of VCS.