GIT

What is "version control"

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later

VCS enables us to control the whole project. If necessary, this allows us to revert one or more files any of their previous versions or the whole project to a previous version. We can also compare changes to one file between two versions in order to see exactly what was changed in each file, when it was changed and who made the change. We can also see why the change was made.

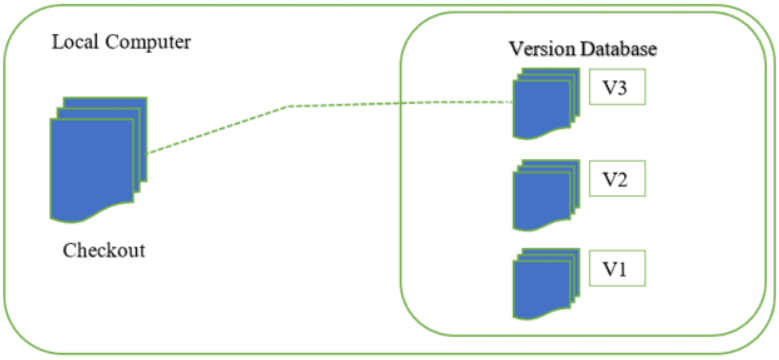
The types of **VCS** are:

Local Version Control System

Centralized Version Control System

Distributed Version Control System

Local Version Control System

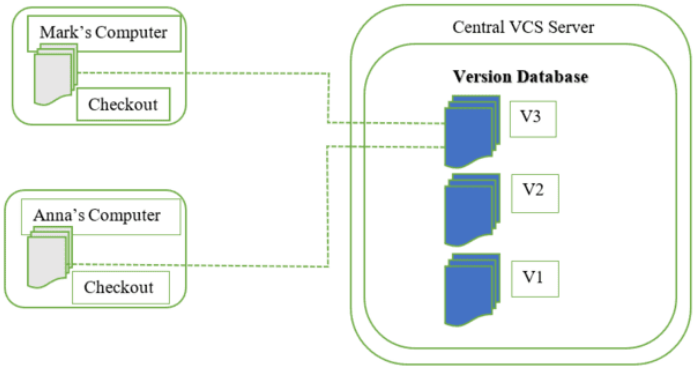


A local version control system is a local database located on your local computer, in which every file change is stored as a patch. Every patch set contains only the changes made to the file since its last version. In order to see what the file looked like at any given moment, it is necessary to add up all the relevant patches to the file in order until that given moment.

The main problem with this is that everything is stored locally. If anything were to happen to the local database, all the patches would be lost. If anything were to happen to a single version, all the changes made after that version would be lost.

Also, collaborating with other developers or a team is very hard or nearly impossible.

Centralized Version Control System



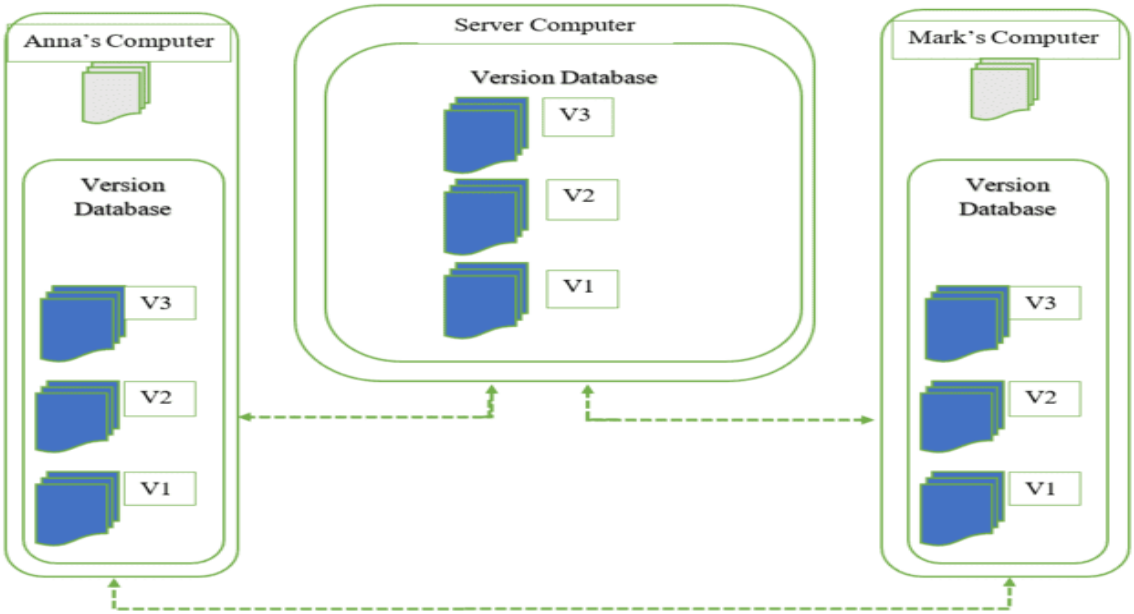
A centralized version control system has a single server that contains all the file versions. This enables multiple clients to simultaneously access files on the server, pull them to their local computer or push them onto the server from their local computer. This way, everyone usually knows what everyone else on the project is doing. Administrators have control over who can do what.

This allows for easy collaboration with other developers or a team.

The biggest issue with this structure is that everything is stored on the centralized server. If something were to happen to that server, nobody can save their versioned changes, pull files or collaborate at all. Similar to Local Version Control, if the central database became corrupted, and backups haven't been kept, you lose the entire history of the project except whatever single snapshots people happen to have on their local machines.

The most well-known examples of centralized version control systems are Microsoft Team Foundation Server (TFS) and SVN.

Distributed Version Control System



With distributed version control systems, clients don't just check out the latest snapshot of the files from the server, they fully mirror the repository, including its full history. Thus, everyone collaborating on a project owns a local copy of the whole project, i.e. owns their own local database with their own complete history. With this model, if the server becomes unavailable or dies, any of the client repositories can send a copy of the project's version to any other client or back onto the server when it becomes available. It is enough that one client contains a correct copy which can then easily be further distributed.

Git is the most well-known example of distributed version control systems.

GIT Version : 2.27.0.windows.1

==> Repositoty : used to store the data or files to share commonly.

==> Git called as central repository.which is on under remote servers. or git is called as version control tool.

==> Git provides the advantages of tracking the vesrions of files.

==> Git hub : which will host the git repositories

GIT is a software and it should be installed.

Git Notes April 2023

==> ** GIT Architecture :

In GIT Architecture we have 4 stages

- 1.Work Space
- 2.Staging Area
- 3.Local repo / git repo
- 4.Central repo

1.Work Space : it is a place where we edit, modify project related files
all the files in workspace are visible to all directories

2.Staging Area : on Git Add files are moved from work space to staging area where changes are saved

3.Local repo / git repo : on Git Commit, files will be added to local/git repo & then we track the file versions
Commit ID are created here

4.Central repo : on Git Push, files are moved to central repo.

==> To intialize git :

sudo yum install git
mkdir git
git init
ls -a

git config --global user.name "name"
git config --global user.mail "mailid"

create files - vi file1
git add file1
git status
git commit -m "mesage"

1.git init : command is used to initialize the respective directory as git repository
2.how to check a directory is git repository or not ??
if .git folder is present in the current directory, then it is a git repository
3.git status :it will show whether files are in workspace or staging are on in git repo
4.git commit -m "message" :this command commits the staged changes to the local repository.
5.git config --global user.name "name(bhavya)"
git config --global user.email "mailid (abc@gmail.com)"
6.git log : gives history of file or repo.
7.git checkout commitid - to go back to previous version .. or move to respective version /
switch to respective commit ID

Branching : Branching is a parallel development teams can work on same piece of code on different branches
parallel and later integrate by merging

Why we need branching ??
To develop new features we go for barching.

git branch --> to check number of branches. or list all branches
* --> indicates the current branch or indicates we are on that branch
git branch <branchname> --> to create a branch
git checkout <branch name> --> to switch to another branch
git branch -d branchname -> to delete the branch
git checkout -b branchname -> to create new branch

Merging :

git merge <branch name> --> to merge specified branch to checked out branch
merging is only 1 way -- from source to destination.

only 1 master per repository

==> **Merge Conflict** : occurs when same piece of code is been worked on different branches, and when we try to merge, the conflict occurs, we can contact particular developer to modify the changes

git cherry_pick commit_id --> to merge specific commit on branch(currently checkout branch)

Tagging : it is a name given to set of versions of files and directories, it is easy to remember the tag names, it indicates milestone of a project.

git tag tagname --> to create a tag name
git checkout tagname --> switch to tag
git tag --> to list all tags
git tag -d tagname -> to delete the tag

==> Difference between tag and branch

tag is a name given to set of files
branching is for parallel development

==> Assignment :
try to create master branch
in master branch create a file biggest3.c
in master branch create a file fact.c and give tag name
under master you create one more branch --> branch 1
in branch 1 create a file with code of reverse.c
again create a branch 2 in master branch
in branch 2 create a file fibbonacci.called
merge branch 1 and branch 2 in master branch
---
git revert - we can go back to the workspace after committing, but commit history will be stored
git revert HEAD~1

git reset - we can go back to the workspace after committing, but commit history is removed
git reset –hard HEAD~1
git reset –soft HEAD~1
git reset –mixed HEAD~1

git revert - committed changes are reverts, but commit history will be store
git reset - committed changes are reset, but commit history is removed
git restore -- to unstage a file

Rebase : similar to merge, rewrite the commit histroy
-> git rebase branch name
Rebase works in two modes
1.standard mode
2.Interactive mode

Difference b/w merge and rebase
Both merge and rebase perform same operation of integrating branches, but the difference is how they do it
merge :merge specified branch to checked out branch, creates new commitID indicating merge
merge conflict can be handled easily, as the commits are reachable

Rebase : rewrites the history by creating new commits for each commit in source branch
since commit histroy is rewritten, it will be difficult to understand the conflict in some cases as commits are no longer reachable

Squash : it is a technique to condense large number of commits to make into small number of meaningful commits
so that we can make git histroy clear.
--> git rebase -i HEAD~n

Stash : if i am working on one branch and i get critical work/bug to be fixed on another branch I don't want to commit changes in the current branch, so i will do git stash where the files will be stored in temporary area, and switch to other branch i will fix the issue and come back to previous branch to continue any work.

-> I need to do git stash pop, to get back files from temporary area

git stash
git stash pop

git conflict : same the definition of merge conflict.

Central Repo :

git clone --> will bring central repo to local work space for the first time

git pull --> it will compare if there are any changes, it will bring changes from central repo and merges to local repo automatically

git push --> it moves local changes from local repo to central repo

if you want consider current directory as central repository
git init --bare --> acts a central repo , we can only push and clone/pull changes to repository

git init -> act as local repo (non bare repository)

we have two types of repositories
1.bare repository - only we can pull and push the files, git operations cannot be performed
2.non bare repository - all the git operations are performed here,we can modify files push to central ,run all git commands

git fetch --> bring changes from central repo to separate branch (under FETCH_HEAD) without merging.

What is difference between pull an fetch
pull = fetch + merge

Assignment :
git amend : To modify your last commit. you can change your log message and files that appear in the commit.

the old commit is replaced by new commit. it means old commit will not be there.

--> it will help you edit the commit message

.gitignore : to ignore a file in the project

--> it will not allow the files to be commited. it will ignore those files.

---u

GIT Fork ??

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Branching Stratagey :

Explain branching Stratagey of your project or release activity or release architecture ??

->Branches can be created for multiple reasons, here we create branches for releases ,development will be going on the dev branch. once the code is ready on the dev brach for the first release we create relase 1 branch and we make a release from the release 1 branch and this branch acts as a maintainance branch for the 1st relase
that means what ever the issues related to 1st release will be fixed on release 1 branch.and parallely development
will be going on the dev branch for the 2nd release once the code is ready for the 2nd release on the dev branch

before we create release 2 branch we merge release 1 branch to dev branch then we create branch for 2nd release
from the dev branch.what ever the issues that we have seen in previous release should not be visible in the next
release.

1.Git Diff:
The git diff command shows the differences between the files in two commits or between your current repository and a previous commit.

2.Git bisect:
The git bisect command is used to discover the commit that has introduced a bug in the code. It helps track down the commit where the code works and the commit where it does not, hence, tracking down the commit that introduced the bug into the code

3.Git insta web :
Instaweb is a script used to set up a temporary instance of GitWeb on a web server for browsing local repositories.

4.Git drop : if wew want to delete commits we use this git drop

5.Amazon EMR : Amazon Elastic Map Reduce
Amazon EMR (previously called Amazon Elastic Map Reduce) is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark , on AWS to process and analyze vast amounts of data.

6.Types of AMI :
AMI is divided into two categories:
1. EBS - backed Instances.
2. Instance Store - backed Instances.

8.What is difference between git and other repository

git
*git is a distributed version control system, that means whole repository will be present in the local workspace.
*if you want to go to previous version of the code, it will be available in the local workspace.
*in git we can work offline(local workspace).
*git has many advanced features like fetch, revert, rebase..etc

other(svn):
*centralized version control systems, only the latest version of code will be there in the local workspace.
*if you want the previous version of the code, it needs to be checked out from the central repo.
*we need to interact with central repo frequently.
*we dont have direct commands to all these features

==> **Git Show** :
git-show is a command line utility that is used to view expanded details on Git objects such as blobs, trees, tags, and commits. git-show has specific behavior per object type. ...
Blobs show the direct content of the blob. Commits show a commit log message and a diff output of the changes in the commit.

-> Git show is similar to git log but it also shows which line and what has been modified

---

==> **Git Hook** :
Git hook are scripts that run automatically every time a particluar event occur in a git repository
==> Git blame :
it will show who has modified the code(eachline of the code)

==> What are the advantages of using GIT?
a) Data redundancy and replication

b) High availability
c) Only one. Its directory per repository
d) Superior disk utilization and network performance
e) Collaboration friendly
f) Any sort of projects can use GIT

==> What language is used in GIT?

GIT is fast, and 'C' language makes this possible by reducing the overhead of runtimes associated with higher languages.

==> What is GIT stash drop?

When you are done with the stashed item or want to remove it from the list,run the git 'stash drop' command.

It will remove the last added stash item by default, and it can also remove a specific item if you include as an argument

---

==> How will you know in GIT if a branch has been already merged into master?

- ❖ Git branch—merged lists the branches that have been merged into the current branch
- ❖ Git branch—-no merged lists the branches that have not been merged

---

==> What is the function of 'git config'?

The 'git config' command is a convenient way to set configuration options for your Git installation.
Behaviour of a repository, user info, preferences etc. can be defined through this command.

==> What does commit object contain?

a) A set of files, representing the state of a project at a given point of time
b) Reference to parent commit objects
c) A SHAI name, a 40-character string that uniquely identifies the commit object.

==> Git Remote ?

The git remote command lets you create, view, and delete connections to other repositories

---

==> What is GIT version control?

With the help of GIT version control, you can track the history of a collection of files and includes the functionality
to revert the collection of files to another version. Each version captures a snapshot of the file system at a
certain point of time. A collection of files and their complete history are stored in a repository

---

==> What is the function of 'git rm'?

To remove the file from the staging area and also off your disk 'git rm' is used.

Create git hub account ,clone push and pull the files

Interview Questions ??

1.What is difference between Merge and Rebase ??
2.Branching Strategy ??
3.Difference between fetch and pull ??
4.merge conflict ??
5.Git Stash ??
