AOS Project – Writeup

Project: Version Control System

LINUXTERS

❖ Sumeet Agrawal
❖ Ashish Chauhan
❖ Gargi Dwivedi
❖ Rishu Toshniwal

ABSTRACT:

Version control systems, often shortened to VCS, are specialized software whose primary goal is to manage changes to codebases over time, a process called version control. The main aim of such a system is to be able to recall specific versions later.

While it is possible to version control any file, the most versatile file types to be version controlled are plain text files such as software source code and documentation.

Some of the most popular version control systems are Git, Mercurial, and Subversion.

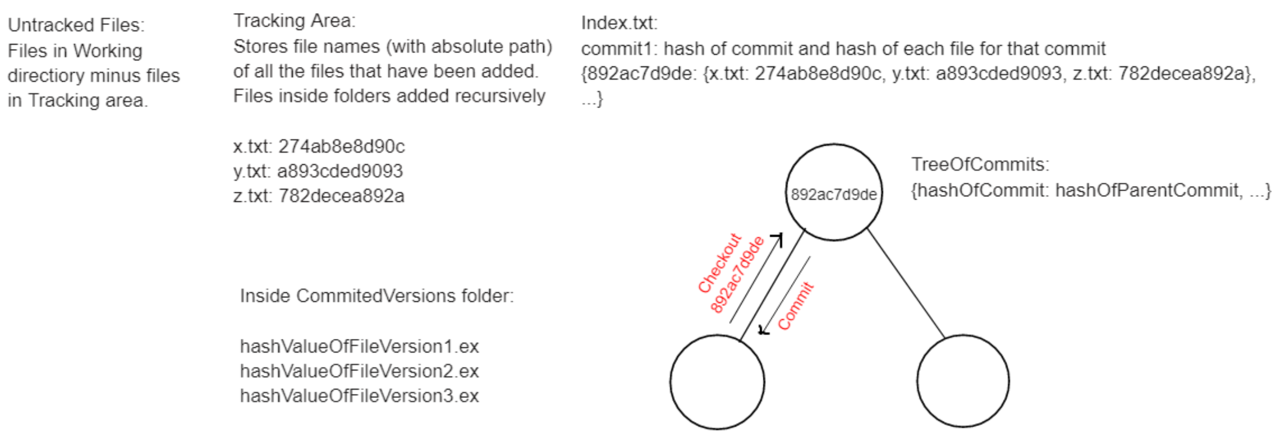
INTRODUCTION:

In this project we are demystifying git versioning system and create a mini version of git which can detect changes, store multiple copies of same files, track changes etc. The functionalities supported currently/ in production are:

1. git init - initializes the necessary files index, staging, tracking etc. in .git folder of root directory.
2. git add - adds the file in the tracking layer.
3. git status - provides information of new files, modified files from last git add and modification on existing files.
4. git commit - commits a checkpoint to the local staging directory.
5. git rollback - goes to the previous commit.
6. git log – provides log of all the commands executed
7. git retrieve – takes back to previous stated version
8. git push - pushes the last committed checkpointed stage to the remote repository.
9. git pull – pulls the latest version of repository.

IMPLEMENTATION:

1. Working Directory - Where the user modifies the files manually. All files present in this folder will be either in trackedFiles or UntrackedFiles.
2. Staging Area - Contains the fileNames (with path) and the corresponding hashcode of the last commit. For the first time this file will be blank.
3. Index File - Contains the history of each commit, files names and the related sha1 hash.
4. Log File - Contains hash value of the commit, commit message, time of the commits etc.
5. Repository - Repository where the files are stored in sha hash format. Multiple versions of same file can be present inside the folder.
6. Working Tree - Contains the map of each commit with other commits.
7. Below Is the HLD for the above implementation



Untracked Files:
Files in Working directiory minus files in Tracking area.

Tracking Area:
Stores file names (with absolute path) of all the files that have been added.
Files inside folders added recursively

x.txt: 274ab8e8d90c
y.txt: a893cded9093
z.txt: 782decea892a

Inside CommitedVersions folder:

hashValueOfFileVersion1.ex
hashValueOfFileVersion2.ex
hashValueOfFileVersion3.ex

Index.txt:
commit1: hash of commit and hash of each file for that commit
{892ac7d9de: {x.txt: 274ab8e8d90c, y.txt: a893cded9093, z.txt: 782decea892a}, ...}

TreeOfCommits:
{hashOfCommit: hashOfParentCommit, ...}

WORKING:

1. The framework starts with git init where the initial setup is done to track the root folder.
2. User adds the files or folders to track with git add file/folder or '.' for all files and folders from any path.
3. User checks the status of current checkpoint with last commit with git status.
4. User decides the current file/folder to checking to staging area of local repository and executes git commit -m "message" file/folder/blank.
5. At any point of time user can go back to the previous commit with git rollback or git checkout commit_hash command. This will overwrite the existing files. The files which are added in the current working directory but not present in last commit will be retained.
6. User can push the last commit to the remote file system like S3, Azure etc with git push command. Any files deleted in the commit will be deleted from the remote repo, similarly any files modified or added will reflect at the repo.

TIMELINE:

Milestones =>

- 17th Nov – High level design completed.
- 20th Nov – git Init/status/add/commit 80% completed.
- 24th Nov (Est) – complete backlog, git checkout/rollback
- 27th Nov (Est) – Multiple level Testing/Git push, Report buildup