

Objective

Implement a git like client system with limited capabilities. The program should be able

- to create/initialize a git repository, creating a .git folder with the necessary details.
- to add files to index and commit, maintaining the commit-ids so that retrieving them back could be done.
- to see the status, diff, checkout previous commits, as shown by the git utility.

Commands Implemented

➤ Init command

Command: ./vcs init

This is the first command that will be run. It is used to initialize a new repository(.vcs in our project). The repository created will store an image of the code base and its associated metadata. All the other commands can be run only when the repository gets created using the "init" command.

➤ Add command

Command: ./vcs add [filename/director name] [..] ..

This command is used to add files and directories to the staging area. Once the files are staged they can be committed by the commit command. Multiple files, Directories can be added to staging area the input format will be name of the files space separated.

➤ **Commit command**

Command: ./vcs commit [commit-message]

Upon running this command, all the changes that were made to the staging area will be saved in the local repository(.vcs in our project). Commits are created to capture the current state of a project. A permanent copy of the changes that were made will be saved in a folder inside this repository, and the user can rollback to this stage any time he wants. Before running the commit command, add command is used to promote changes to the project that will be then stored in a commit.

➤ **Status command**

Command: ./vcs status

This command shows the current status of the staging area and its difference with the working area/tree. It gives the following details for the working directory.

Staged Files

These are the files added to the staging area and ready to be committed.

Untracked Files

These are the files which are not being tracked by the vcs.

Modified Files

These are the files which are tracked and have been changed since the last add/commit.

Deleted Files

These are the files which were being tracked and now deleted.

➤ **Rollback Command**

***Command: ./vcs rollback
[commit_id_of_commit_user_wants_to_rollback]***

This command is used to rollback to the previous version. Users must give the commit id of the rollback they want to go. Before rolling back the current state of the working directory is committed.

➤ **Diff Command**

Command: ./vcs diff [filename/directory]

This command compares the file in the current working tree with the file present in the staging area and shows the difference between them. If the file is not present in the staging area then it attempts to find its difference with the last committed version, if the file is also not present over there then it classifies it as a new file. In case of directory, it shows difference in the count of files present in that directory.

➤ **Log command**

Command: ./vcs Log

Upon running this command, all the previous commits metadata will be displayed. For each of the commits its associated SHA, commit ID, hexcode and message will be displayed.

➤ **Retrieve command**

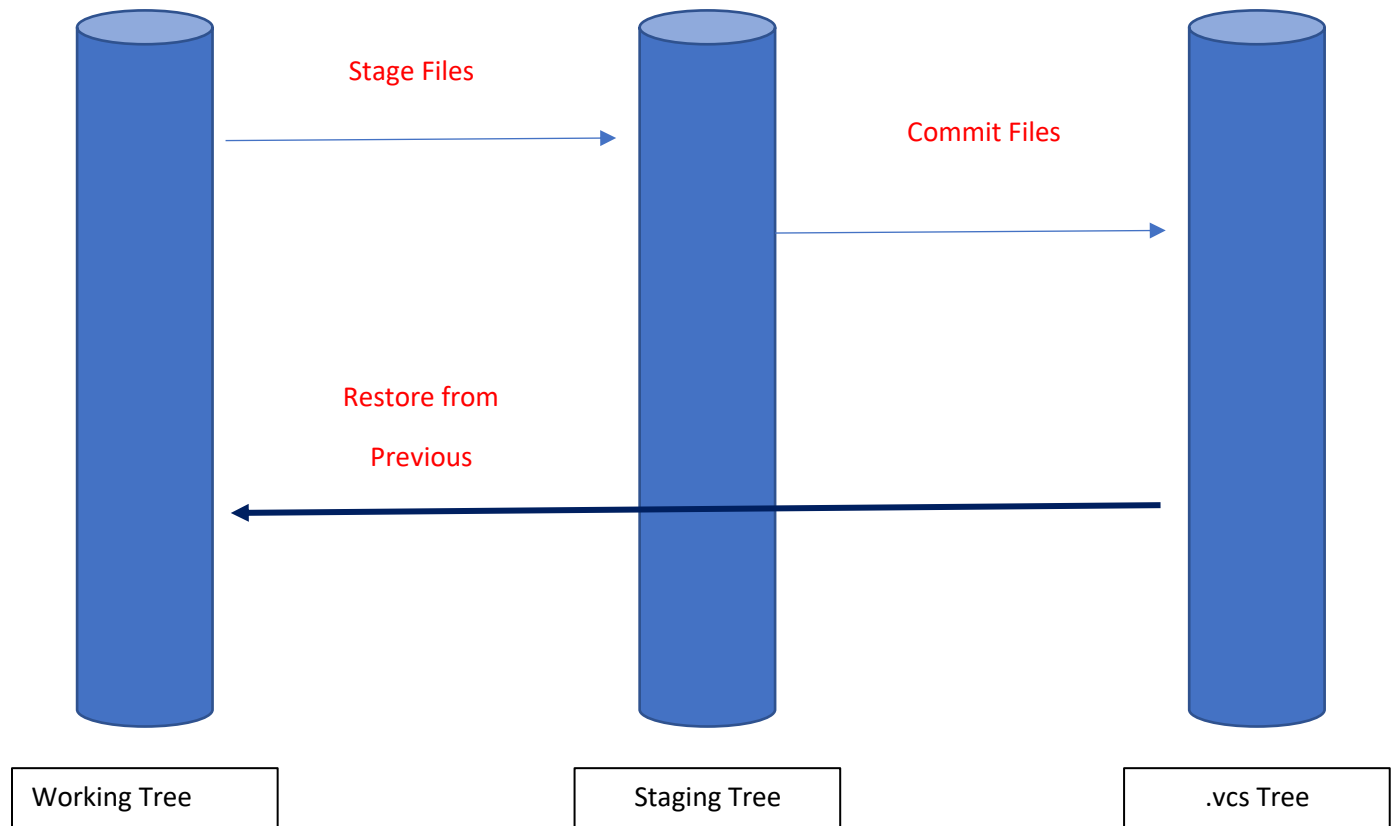
- ***Command: ./vcs retrieve -a version***

User can run this command to get the “Commit Id” of the particular version no.

- ***Command: ./vcs retrieve SHA version***

User can run this command to get the “Commit SHA” of the particular version no.

Architecture



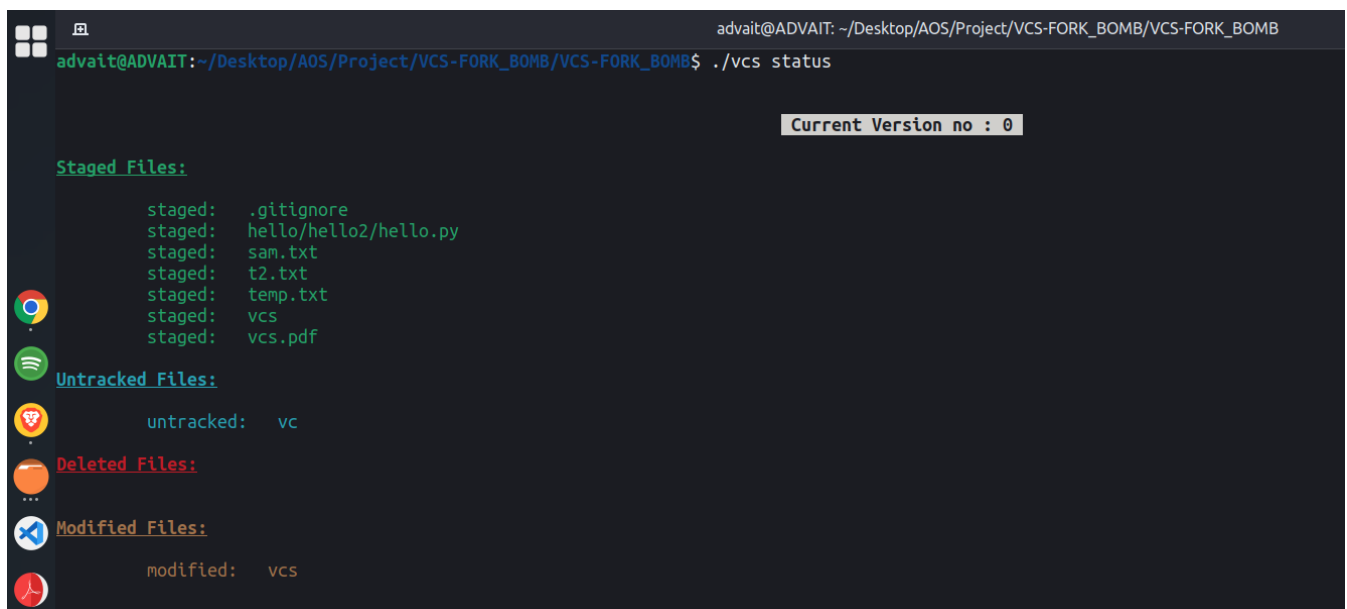
Working in Brief

`./vcs init`

This command will create a .vcs repository and the required structure. This folder will contain all the version control information.

`./vcs status`

The following command will show the status of the current working tree. Initially all will be untracked.



```
advait@ADVAIT: ~/Desktop/AOS/Project/VCS-FORK_BOMB/VCS-FORK_BOMB
advait@ADVAIT:~/Desktop/AOS/Project/VCS-FORK_BOMB/VCS-FORK_BOMB$ ./vcs status

Current Version no : 0

Staged Files:
    staged: .gitignore
    staged: hello/hello2/hello.py
    staged: sam.txt
    staged: t2.txt
    staged: temp.txt
    staged: vcs
    staged: vcs.pdf

Untracked Files:
    untracked: vc

Deleted Files:

Modified Files:
    modified: vcs
```

`./vcs add temp.txt`

This command will add the file/files in the current directory to the staging area.

```
./vcs commit "first commit"
```

The following command will commit all the files in the staging area.

```
./vcs diff temp.txt
```

The following command shows the difference for temp.txt. As described above.

```
./vcs log
```

The following command will log all the previous commit data.



```
advait@ADVAIT: ~/Desktop/AOS/Project/VCS-FORK_BOMB/VCS-FORK_BOMB$ ./vcs log

Current Version no : 3

Commit e42fe261ad7a44b5834b772fb8851980e307db9c (Version 0)
Date: Mon Nov 28 20:27:32 2022
Commit Id: 10de6db
First Commit

Commit b3bae35e3820b5274d5f3265ec0c12496b21acf6 (Version 1)
Date: Mon Nov 28 20:31:54 2022
Commit Id: ee70246
temp.txt file updated

Commit 8876cc9412e998e49381ae6c9234a0575d267f38 (Version 2)
Date: Mon Nov 28 20:33:50 2022
Commit Id: 94d0433
Some Corrections
```

```
./vcs rollback 10de6db
```

This will first commit the current staging area, and then rollback to the version corresponding to the commit id given.

```
./vcs retrieve -a 0
```

This command will return the "commit id" of the version no, if the version no is valid.

```
./vcs retrieve SHA 0
```

This command will return the "commit SHA" of the version no, if the version no is valid.