Experiment 1

Aim: To study and implement version control using GIT

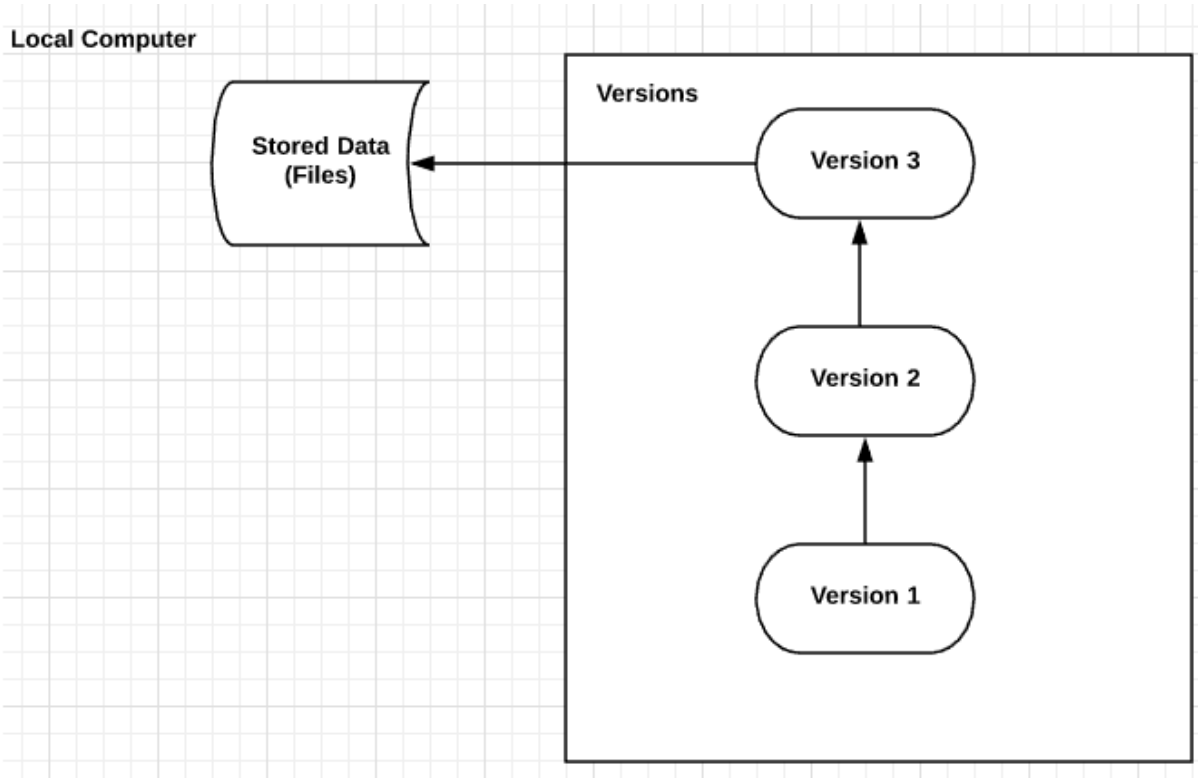
Theory:

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. Version control systems are also called as revision control systems. Revision control systems work as independent standalone applications. Applications like spreadsheets and word processors have control mechanisms. The unique features of version control system/ revision control system are as follows: Up to date history is available for the document and file types. It does not require any other repository systems. The repositories can be cloned as per the need and availability. This is extremely helpful in case of failure and accidental deletions. VCS includes tag system which helps in differentiating between alpha, beta, or various release versions for different documents.

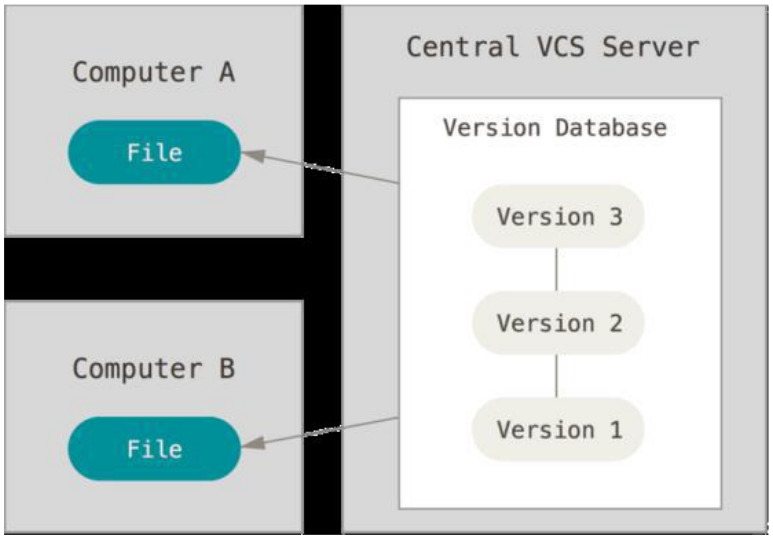
The various types of the version control systems are:

1. Local Version Control System
2. Centralized Version Control System
3. Distributed Version Control System

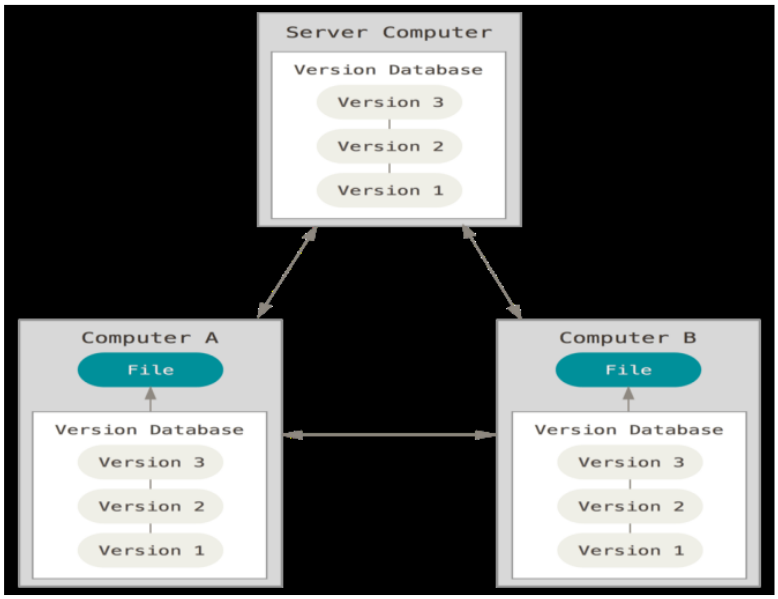
1. **Local version control system**: Local version control system maintains track of files within the local system. This approach is very common and simple. This type is also error prone which means the chances of accidentally writing to the wrong file is higher.



2. Centralized Version Control System: In this approach, all the changes in the files are tracked under the centralized server. The centralized server includes all the information of versioned files, and list of clients that check out files from that central place.

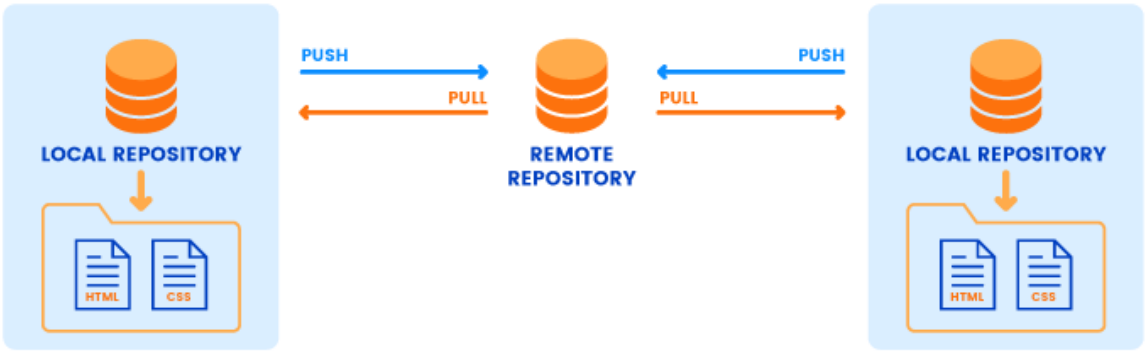


3. Distributed Version Control System: Distributed version control systems come into picture to overcome the drawback of centralized version control system. The clients completely clone the repository including its full history. If any server dies, any of the client repositories can be copied on to the server which helps restore the server.



Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows. Git is a distributed version control system (DVCS). "Distributed" means that all developers within a team have a complete version of the project. A version control system is simply software that lets you effectively manage application versions. Thanks to Git, you'll be able to do the following:

- Keep track of all files in a project
- Record any changes to project files
- Restore previous versions of files
- Compare and analyse code
- Merge code from different computers and different team members.



The commonly used git commands are listed as follows:

Git: configurations

```
$ git config --global user.name "FirstName LastName"
$ git config --global user.email "your-email@email-provider.com"
$ git config --global color.ui true
$ git config --list
```

Git: starting a repository

```
$ git init
$ git status
```

Git: staging files

```
$ git add <file-name>
$ git add <file-name> <another-file-name> <yet-another-file-name>
$ git add .
$ git add --all
$ git add -A
$ git rm --cached <file-name>
$ git reset <file-name>
```

Git: committing to a repository

```
$ git commit -m "Add three files"
$ git reset --soft HEAD^
$ git commit --amend -m <enter your message>
```

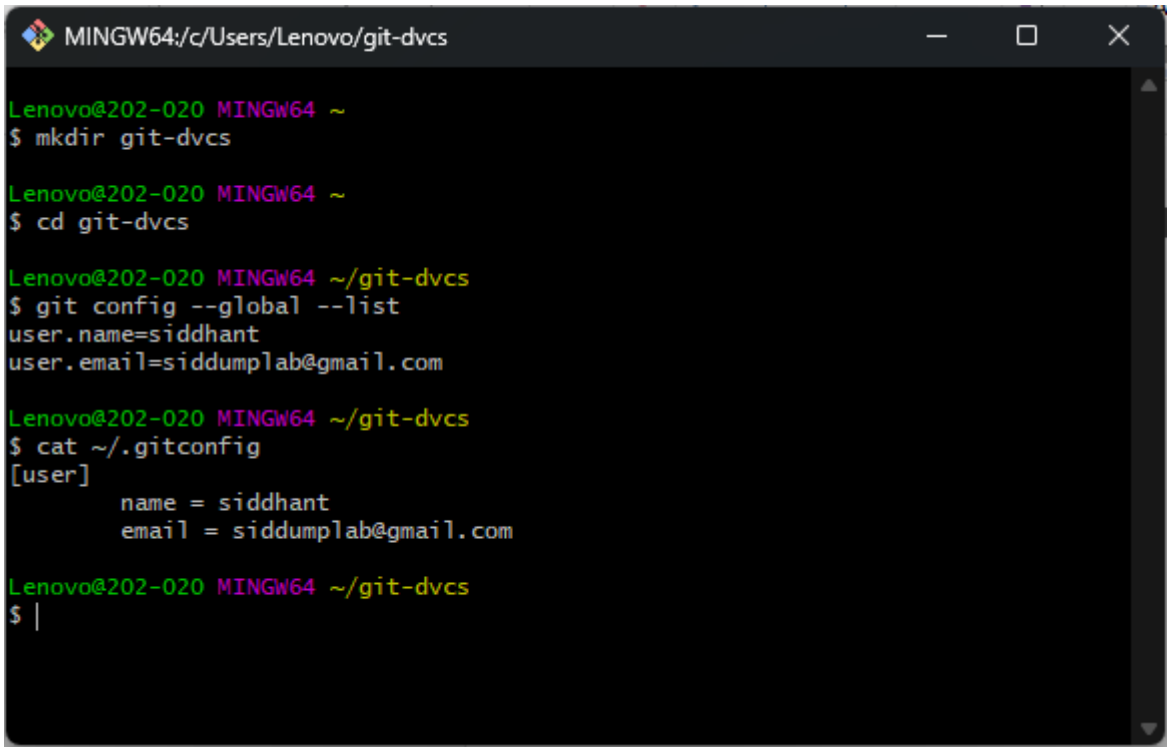
Git: pulling and pushing from and to repositories

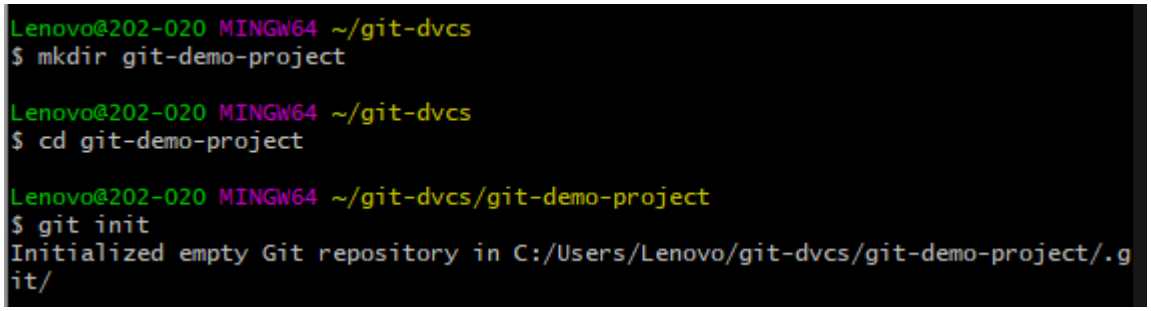
```
$ git remote add origin <link>
$ git push -u origin master
$ git clone <clone>
$ git pull
```

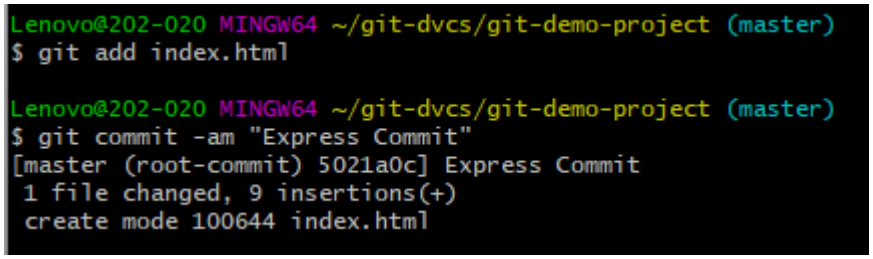
Git: branching

```
$ git branch
$ git branch <branch-name>
$ git checkout <branch-name>
$ git merge <branch-name>
$ git checkout -b <branch-name>
```

Implementation:







```
Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        teststatus

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git checkout -- teststatus
error: pathspec 'teststatus' did not match any file(s) known to git

Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git checkout -- index.html
```

```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        teststatus

nothing added to commit but untracked files present (use "git add" to track)

Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git add teststatus

Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   teststatus
```

```
Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git commit -am "Express Commit"
[master b832d40] Express Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 teststatus
```

```
Lenovo@202-020 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git log
commit b832d40533b93a6eda48eb1382b274a7c67a7b2e (HEAD -> master)
Author: siddhant <siddumplab@gmail.com>
Date:   Fri Jan 19 16:20:23 2024 +0530

    Express Commit

commit 5021a0cc894063d98af69bb5add57e01109e5376
Author: siddhant <siddumplab@gmail.com>
Date:   Fri Jan 19 15:55:37 2024 +0530

    Express Commit
```

Conclusion: Thus, we have successfully studied and implemented version control using GIT.