**Git Features**

* Distributed - every dev has local copy of code
* Compatible - compatible with existing systems (SVN)
* Non-linear - non-linear development of software through branching
* Branching - we can create branches from master branch(MB has production quality code)
* Lightweight - compress data efficiently on client side
* Speed - fetching data from repos is faster than other vcs’s
* Reliable - even system crashes lost data can be recovered.

**Git Operations**

* Create Repos
  + Git init
* Make changes
  + Status - tells which files are added to index and ready to commit.
  + Add - add files to index
  + Commit – save changes in central repo
* Parallel development
  + Branch
  + Merge
  + Rebase
* Sync repos
  + Remote Add origin <link> - adds remote repo
  + Pull
  + Push
* Download/clone repos
  + Git clone

**Git Commands**

* git remote add origin <link> (connects local & remote repos)
* git add -A (adds all files to index)
* git status (check whether files are in index or not)
* git status -s (short desc of whether files are in index or not)
* git commit -m (commit files)
* git commit -a -m (add all files to index & commit files)
* git branch -d branch\_name (delete branch after merging)
* git checkout -b develop (create & switch to branch)
* git branch -D branch\_name (delete branch forcely)
* git log (shows commit history (commit id, author, date and message))
* git log –oneline (shows commit history in single line)
* git log –oneline --reverse (shows commit history in single line in reverse order)
* git pull <link> (pulls code from central repo)
* git branch branch\_name (creates branch)
* git checkout branch\_name (switch to branch)
* git merge branch\_name (merge branches)

**Note:** Perform merge in destination branch (master/main)

To delete branch checkout to other branches

**KUNAL KUSHWAHA**

**LINUX COMMANDS**

* ls (list files)
* touch file.txt (creates file)
* vi file.txt (write file)
* mkdir dir\_name (create dir)
* cd dir\_name (change dir)
* ls -a (show hidden files)
* cat file.txt (show data in file)
* vi file.txt (adds data from terminal)
* rm -rf file.txt (deletes file)
* mv file1.txt file2.txt (rename file)

**GIT COMMANDS**

* git config –global user.name “name”
* git config –global user.email [email@dom.com](mailto:email@dom.com)
* git config –global core.autocrlf true (windows-true) (max/linux-input)
* git restore --staged file.txt (removes file from staging before commit)
* git reset commit\_id (restores project to this commit)
* git reset HEAD~
* git ls-files (view files in staging area)
* git rm file1.txt (removes file from both staging & working area)
* .gitignore (mention folder/files to ignore inside this)
* git rm –cached -r file.txt (removes file from index)
* git show commit\_id (check commit info)
* git show HEAD (check last commit)
* git show HEAD~3 (checks 3rd  commit from last)
* git push -u origin branch\_name (sets upstream branch)
* git stash (push files into backstage for later use without committing, files must be added to index)
* git stash pop (pulls files in backstage into local repo)
* git stash clear (clear stash area)
* git stash show (show stash)
* git push origin branch\_name (push code to branch)
* git revert commit id (restores to previous version)