GIT

Git is a decentralized or distributed version control system

Repository – collection of files managed by git

Commits – Git works by saving the current state of all its files it manages into snapshots called commits

A commit can contain one or many file changes

As changes are made, commits are saved onto the timeline known as branch.

Default branch – master

Github is a git repository hosting service.

Installation – [https://git-scm.com](https://git-scm.com/)

git config --global user.name "ashithas12"

git config --global user.email [ashithasuresh12@gmail.com](mailto:ashithasuresh12@gmail.com)

git init demo – initialise a project demo

Local Git States –

* Working directory –contains all files and folders for the application
* Staging area – is used to prepare for next commit
* Repository(.git folder) – all committed or saved changes to the git repository
* Remote – with all three states internally

ctrl+w to close the file in editor

exit notepad++ - alt+f4

git status

git add filename – moves the file to staging area

git status

git commit -m “First commit” (-m – message)

git status

git init . – initialise git repository inside demo folder

vi sample.md

git add . – add all files to the git staging area

git commit

git log – provides info on commit entry

git show – shows the last commit and a difference containing all the changes

to get out of show command press q

make changes to sample.md file and view the status

git status

touch new.file

git ls-files – lists files in the directory

rm new.file

git commit -am “Updating README.md” (-a parameter tells git to first add modified files to the git staging area)

git log

vi README.md

git add . – to add the files to staging area

git status

git reset HEAD README.md – to unstage the file

git checkout -- README.md - revert changes of the files

git status

git log

git help log

git log --oneline --graph --decorate --all

--oneline – provide log in a single line

--graph – provides an asterisk based graph denoting our branching hierarchy

--decorate – tell us which commits are a part of which branches

--all – provide history for all branches that are available in the repository

ALIAS in GIT

git config --global alias.hist “log --oneline --graph --decorate –all”

git config --global --list

git hist – now shows the log

git hist -- README.md – shows log pertaining only to README.md file

GIT RENAME

git mv example.txt demo.txt - rename file

commit the file after renaming

git rm demo.txt - remove file

git status

commit changes after deleting

when you rename the file without using git you need to give the below cmd to update the git status

git add -u – update the git status -only picks up the deletion update

git status

git add -A – to update all kind of modifications

EXCLUDING UNWANTED FILES

Open .gitignore file and add \*.log which ignores all .log files

git add .gitignore

git commit -m “adding ignore file”

git hist

git diff <commitidd> HEAD– to get difference between two commits, HEAD is the last commit

git diff – diff bet recently changed in the working directory to the HEAD

git help diff

BRANCHING AND MERGE TYPES

A branch is a timeline of commits. Once integrated with the main timeline, they are not needed anymore

Types-

* Fast forward- when no additional work has been detected on the parent branch(master), git will apply all commits from the other branch directly on to the parent branch
* Automatic- this happens when git detects non conflicting changes in the parent branch. It automatically resolves conflicts
* Manual- this happens when git is unable to automatically resolve any conflicts

SPECIAL MARKERS

HEAD – last commit of current branch

Its like pointers

BRANCHING

git status

git branch

git checkout -b updates – to create a new branch called updates

make changes to the file

git add .

git commit -m adding changes”

git diff updates master

git checkout master – to switch branches

git hist

git merge updates – merge the changes with master

git hist

git branch -d updates – delete updates branch

CONFLICT RESOLUTION

git checkout -b very-bad

git branch -a – shows all branches

make changes to the file

git commit -am “very bad update”

git hist

git checkout master

git branch -a

git merge very-bad

git mergetool

git commit -m “Resolving cinflict”

git status

GIT TAGS

git status

git hist

git tag mytag

git tag –list

git hist

git tag -d mytag – delete a tag

git tag -a v1.0 -m “Release 1.0” – a for annotated tag

git tag –list

git show v1.0

STASHING

Modify file

git stash

git stash list

git status

modify file

git commit -am “updating”

git status

git stash pop – apply and drop

git stash list

git reset <commitid from git hist> --soft (can be of three type-> --soft, --hard, --mixed)

git reflog – shows all the different actions taken while in this repository

GITHUB

Link Github to Git

git remote add origin <url of repo created>

git remote -v

Pushing changes to Github

Inside demo repo in git

git push -u origin master --tags

CLONE A REPOSITORY

Create a repo in github

Copy the https clone url from github

git clone url

or

git clone url website – naming the repo as website

SEED REPOSITORY

Initializr website to download files

cp -r folder/\* .

git add .

git commit -m “commit”

git push origin master

FETCH and PULL

Changes made in repository

To reflect changes in local repo use fetch/pull

git fetch

git status

git pull

git push

Updating repository

git remote set-url origin https url clone url

git remote -v

git remote show origin

push local branch to repo

git push -u origin branchname

git pull --all – will update all branches

git push origin :branchname – deletes the branch and pushes

GIT TAGS

git tag tagname branchname

git log --oneline

git tag -a v0.1 -m “Release 0.1” commitid

git show v0.1

git tag