GIT

INTRODUCTION:-

Version control is a software tool which tracks when file is modified every time.

* Also called as revision control or source control
* Modification of a file/code is referred as version. Each version captures snapshot of the file

Ex: - If X developer is working on file/code. He always make changes, all those changes will tracked by version control. So whenever he wants to rollback to previous code in future, he can.

Advantages:-

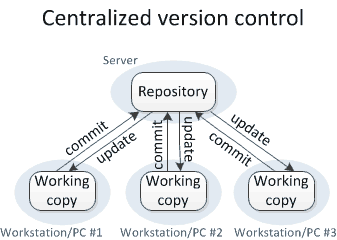
1. Allows developers to work simultaneously
2. Avoids overwriting each other changes
3. Maintains history of work
4. vcs allows to roll back earlier version of file when current file is not working

Types of vcs:-

1. centralized version control:-

There will be a centralized/remote server and the developers need to connect online while working

Ex: - svn, cvs

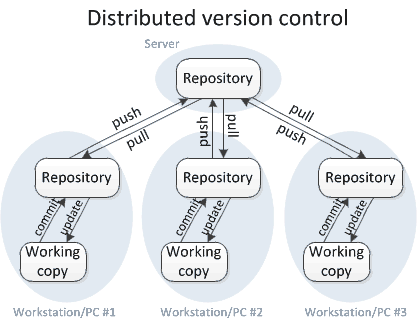


Drawbacks:-

1. Single point of failure means if central server goes offline then there is no collaboration between developers
2. We need to be always online
3. Distributed/Decentralized version control:-

Here we have remote repositories, developers will clone entire repo into his local system

Ex:-git, mercurial



Advantages:-

1. need not to be online every time
2. can restore from local repository if central repository went wrong
3. maintains repository in multiple locations and multiple copies

GIT INTRO:-

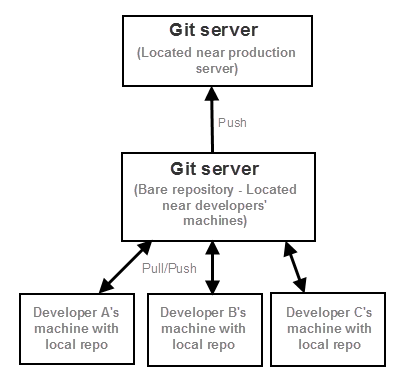
Git is a distributed version control system. It can track changes to a file and allows you to revert to any particular change.

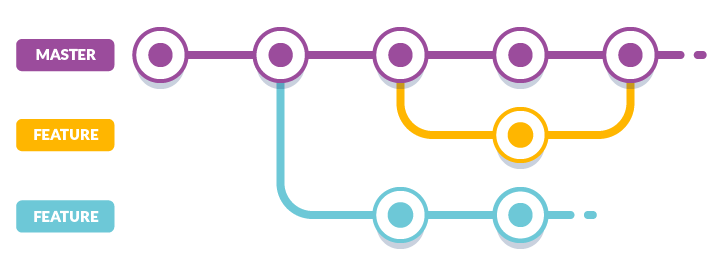
* git only tracks files and ignores empty directory

GITHUB is a website/remote repository where we push and fetch code. It also maintains backup

* Git can be used with either github/bitbucket there is no relation between git and github

GIT ARCHITECTURE:-





GIT WORKFLOW:-



For the first time when you initialize/clone git, a local repository (.git) will be created in work space, and master branch will be created by default. Now all files will be in workspace, we will add to index area (at any time we can revert from staging area to workspace) and we commit files from index area to local repo and finally push from local repo to remote repo

We will fetch the files from remote repo to local repo. Git merge will move your local repo files to workspace (now your remote repo files and workspace files are updated). If you want remote repo files to be updated in workspace we use git pull (fetch + merge)

By default there will be a master branch but we create a feature (new branch) in order not to disturb the master branch if at all we want to add some new features to the application, after we done successfully in the feature branch we merge feature branch with master branch

GIT CONFIG FILES:-

1. /etc/gitconfig------------
2. /.git/config --------------

GIT TERMINOLOGY:-

Clone = copies an existing git repository from remote location to local (laptop)

Commit=submitting files to the local repository

Version= modification of file is referred as version

Repository is a centralized location where data is stored and managed generally directory or cloud

Bare repository= where repo is available in the local system

Source code repository= a repository where source code is maintained. Ex:- github

GIT COMMANDS:-

1. init:-

This command actually initiates the git repository. (.git) hidden directory is created

#git init

1. clone:-

Copies entire remote repository (github) into local system [download entire project into a directory]

#git clone git\_hub\_link

If you want to clone particular branch of a repositry

#git clone git\_hub\_link -b Branch\_Name

1. workspace (or) working tree:-

The place where we generally work like modify add delete files

* this is the area which holds actual files ex: - git directory 🡪/git

There are 4 states of a file

1. untracked----when new file created(in workspace) it’s not tracked by git
2. tracked/staged file----when a file added to index area
3. committed file---when you commit file to local repo
4. dirty/modified file----the file has been staged and later if you modify the same file in working tree, then that file is called dirty file
5. staging area (or) index area:-

This is the place which holds all the changes before committing into local repo.

These changes are temporary, at any point you can commit or revert changes

1. local repo:-

It is a directory in local system generally hidden directory (.git)

1. remote repo:-

Remote repo is located remotely

Ex: - github/bitbucket/gitblit

1. add:-

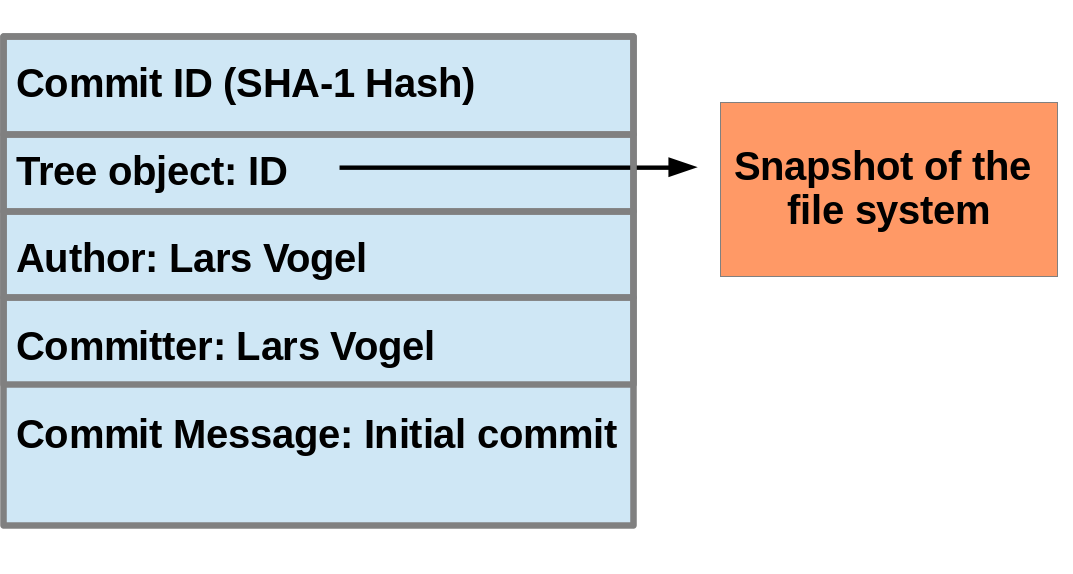
This command moves your modified file from workspace to staging area

#git add filename (or) #git add .

Where . is current directory

1. commit:-

It allows you to move your files from staging area to local repository and the changes will be saved safely in local repo



#git commit –m “message” (or) #git commit –m “msg” filename

1. commit –am:-

Commit –am = add+ commit

It does add and commit action once

#git commit –am “message”

1. push:-

Push command allows you to move files from local repo to remote repo (github)

* If local repo doesn’t have any changes present in remote repo then push operation will be failed

#git push –u origin master

Where you want to push= (origin in this case) to which branch you want to push= (master branch in this case)

\*Before you do push operation you always need to pull the remote repo, so your local repo will be updated then (files-in-remote repo=files-in-local repo)

1. fetch:-

git fetch downloads new files from remote repository to local repository. Opposite of push.

#git fetch origin

1. merge:-

All the updated files from local repo will be moved to local repo.

(Or)

Merge current branch with another branch {all your files, commit-ids in your current branch will be merged into destination branch and gives a new commit-id}

#git merge branch\_you\_wanted\_to\_merge

1. pull:-

Pull = Fetch + Merge

Git pull download new files from remote repo and integrate into current working directory

#git pull origin master (or) #git pull

1. log:-

To check files in local repo and shows sha1 or revision number, author, date, commit messages

#git log

1. status:-

Shows files in working space and staging area. Untracked files, staged files

#git status

1. diff :-

git diff shows the changes of same file in local repo and working tree

&

Also shows merge conflicts

#git diff

1. HEAD:-

HEAD is a file in local repo (.git), all files in staging area are committed to HEAD. The last commit you made is pointed to HEAD.

Head points to the master branch by default. If you switch to new branch, Head will point to new branch

1. diff HEAD-

If you committed a file and later modified the same file, now you want to check the difference between them ie.., git diff HEAD

#git diff HEAD

1. merge conflicts

If git couldn’t merge source branch with destination branch. Then it’s a merge conflict

Master branch

#vi f1----> this is mani

#git add f1

#git commit –m “f1”

Second branch

#vi f1----> this is main

#git add f1

#git commit –m “f1”

Imagine if you are in second branch now you want to merge with master branch

*#git merge master*

o/p:- *Auto-merging f7*

*CONFLICT (content): Merge conflict in f7*

*Automatic merge failed; fix conflicts and then commit the result.*

***\*merger conflict will be taken care by developers***

1. config:-

Configuring name, email-id, color, editor in git globally

#git config --global user.name “xxx”

#git config --global user.email “xxx@gmial.com”

#git config --global color.ui true

1. remote:-

Instead of typing remote repo link every time. We will give simple name for that

#git remote add urchoicereponame remoterepolink

To see remote repos

#git remote -v

1. checkout:-

If you want to undo any changes to a file then use this command

#git checkout –filename (or) git checkout first-commit-id

1. branch:-
   * By default master branch will be created in git.
   * The new branch created is called feature branch
   * When you create a new branch git copies entire master branch into new branch
   * The purpose of creating a new branch is to edit
   * When you want to develop a new feature without disturbing the main working application then you create a new branch and modify. If it’s working fine then you will merge current branch with master branch

To create new branch

#git branch new\_branchname

To switch from one branch to other

#git checkout branch\_name

To create and switch branch in one move

#git checkout –b branch\_name

To see all branches

#git branch

To see all branches in server (remote repo)

#git branch -r

To delete a branch

#git branch -D branch\_name

1. rm:-

Removes the file from working space and staging area

#git rm filename (or) #git rm filename –f

\* even if you remove a file you need to add it to staging area and then local repo

1. reset head:-

If you added a file to index area, reset command moves back to workspace. Unstages file

#git reset head filename (or)

#git reset HEAD filename (or)

#git rm --cached filename (or)

#git reset filename

reset soft:-

If you add a file from staging area to local repo, and wants to uncommit the file

#git reset --soft commit-id

ex:- 4th file-------#(jhbdcb78y87dsbd)----latest commit

3rd file----------#(uhduawiuwe87783) -----third commit

2nd file-----------#(jkdshbidbbads7632) ------second commit

1st file-------------#(f383738be8d8ddd8)------first commit

If you want to move 4th file to staging area then you have to give 3rd commit-id

reset mixed:-

If you want to undo files from local repo to workspace in one step (commit🡪staged🡪workspace)

#git reset --mixed previous-commit-id

1. tags:-

Referring/attaching a name to the commit id is called tag. Since we can’t remember commit-id every time

#git tag -a tag\_name -m “message” commit-id

#git tag tag\_number commit\_id

Display all tags

#git tag

Delete tag

#git tag -d tagname

1. statsh:-

It temporarily allows you to save uncommitted files (both staged and unstaged files) for later use. Stash is local to git repo and can’t be pushed to remote repo

* It takes the changes in working and staging area, places them in temporary memory .nges in
* For example, we are working on branch1, there are changes in working and staging area, suddenly we got a mail to fix a issue in production. we want to create a new branch and fix a issue in that. In working and staging area branch those of our core but

#git stash

#git stash save------it saves changes in staging and working area

#git stash clear------to remove changes from stash

#git stash list-----to list files in stash

#git stash pop-----moves your files from stash to working tree

1. clean:-

Removes untracked files

#git clean -f

1. cherry-pick:-

It is picking particular commit-id from other branch to your branch

#git cherry-pick commit-id

1. hooks
2. bare init
3. git rebase:-
4. git reflog:-
5. git modules:-