The process of maintaining the different versions of the code. All team members upload the code into the remote version controlling server where the code get integrated into the single project

Next time when the team download the code they will be able to access the code created by all the team members

Version controlling system also preserves the older version of the code so the team members switch b/w any version based on their requirement

Version controlling system also keep a track of who is making what kind of changes. There are two types of version controlling f

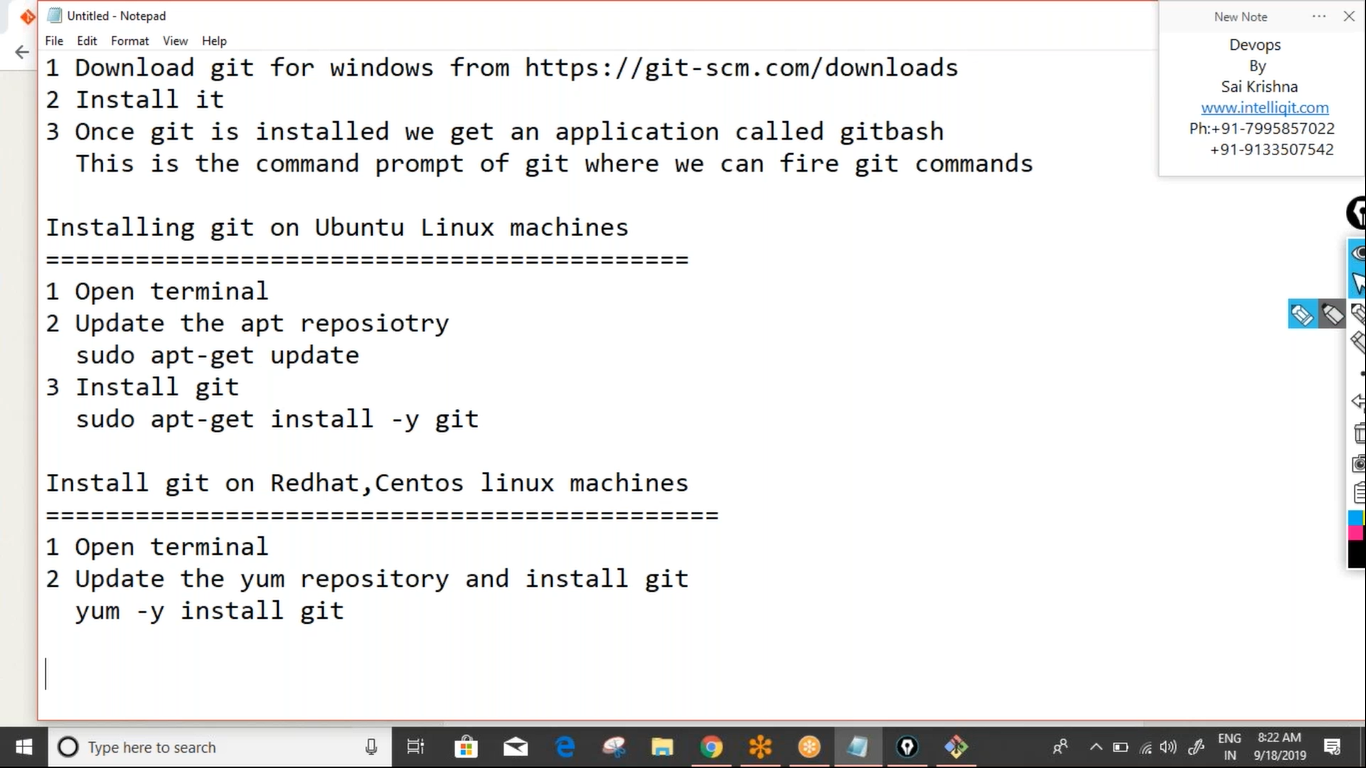
Centralized version controlling:

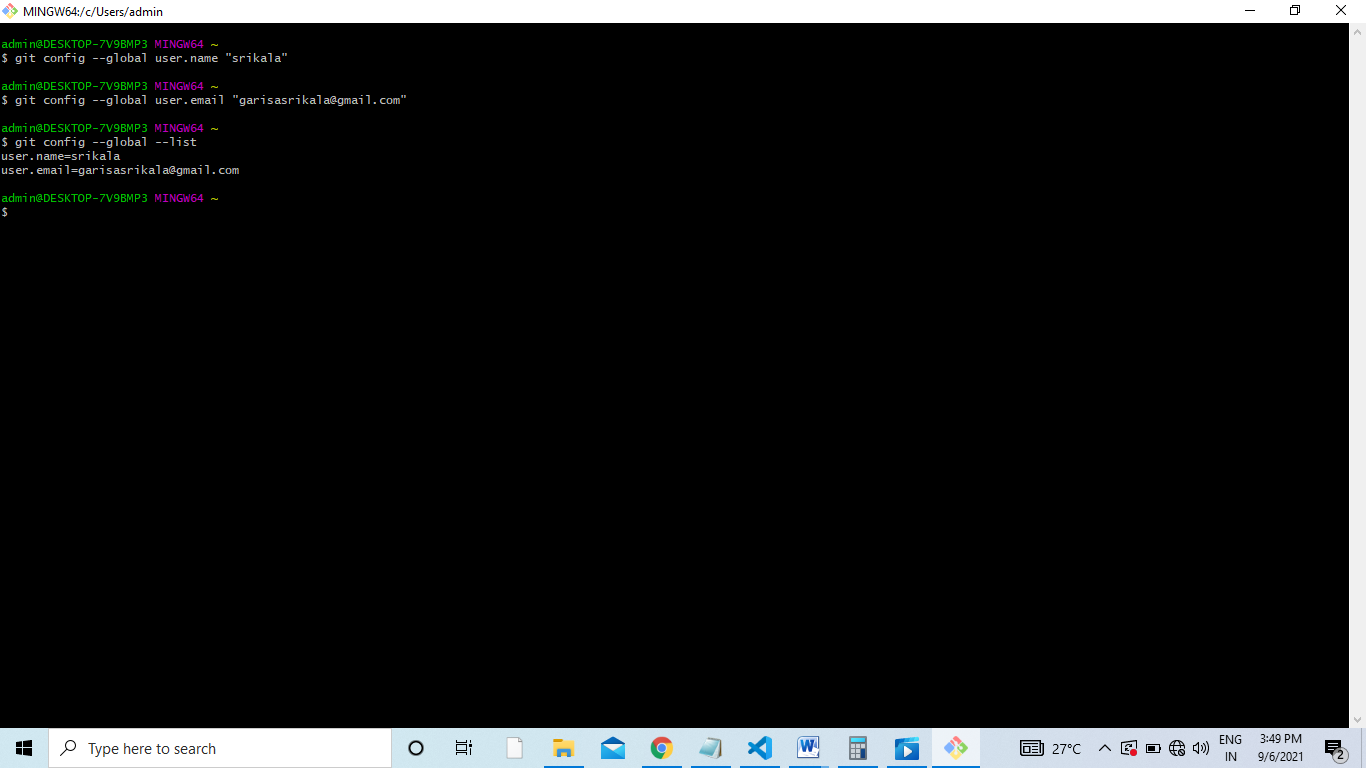
we have remote server where all the code is uploaded and version control is happen only on that remote server example is svn(sub version)

Distributed version controlling:

In this a local repository is installed on every developers machine there initially code is uploaded by the developer. Here version controlling happens only at level of individual developer later code will be uploaded into a remote server where version controlling where version controlling happens at level of team

In this we can create a bare repositories that is repositories at an individual team level or at sub team level and then main repository at the level of entire project





Git when working on the local machine uses three sections

1. Working directory or work space
2. Staging area
3. Local repository

Working dir is the physical folder where the developers create the code and initially all the file present here called the untracked files

Staging area is the intermediate buffer zone into which files are and these files are called the staged files

Local repository where the location where version controlling happens and the files present here are called the committed files

1. To initialize the current folder as the git repo use” git init” command
2. To send files from working dir to staging area “git add filename”

To send multiple files from working dir to staging area “git add filename1 filename2”

To send all files from working dir to staging area “git add .”. “.” represents the current current working directory

1. To send files back from staging rae to working directory

Git rm --chached filename or git reset filename

1. To send files from staging are to the local repository “git commit -m “some message””
2. To see the info about files present in working dir and staging area “git status”
3. To see the list of commits done in the local repository “git log” or “git log --oneline”

Note: git commit will move all the files from staging to local repository to move only specific file from staging to local repository git commit -m “message” filename

.gitignore:

This is a special configuration file where we can store private file info any file whose name is mentioned in .gitignore will no longer access by git

1. Create files touch f1 f2 f3
2. Create a file .gitignore and store the private files in it

Cat > .gitignore

F1

F2

F3

To come out of cat command cntl+d

Git branching:

This is the feature of git which allows developers to create separate branches. These branches are used by the developers for storing code related to different functionalities later these branches can be merged with the master branch. By default main branch called as master branch

1. To see the list of local branches “git branch”
2. To see the list of local remote branches ”git branch -a”
3. To create branch “git branch name”
4. To move into a branch “git checkout branch\_name”
5. To create branch also move into it “git checkout -b branch\_name”
6. To delete branch that is merged “git branch -d branch\_name” also called the soft delete
7. To delete branch that is not merged “git branch -D branch\_name” also called the hard delete

Note: whenever branch is created whatever is the commit history of the master gets copies into that new branch

Note: Irrespective of where the file created or modified git only considers the branch that is committed and that file will belong to that branch only

Working on git remote repository:

1. Open github.com ------🡪 sigup for free
2. Sing into a new account
3. Click on new icon(+) on the top right cirner
4. Click on new repository
5. Enter some repository name ---? Select public or private
6. Click on create repository
7. Go to “push an existing repository from command line” copy and paste the first command this will create the link b/w local and remote

“Git remote add origin link”

Copy the second command and paste it in git bash

“git push –u origin master”

Enter username and password of git

This process of uploading code into remote github called the checkin

Downloading code from github:

To download from git repository we can use the fallowing three ways

1. Git clone
2. Git fetch
3. Git pull

Got clone:

This will download the entire code from repository into the local machine irrespective of whether that code is already present or not

Git clone remote\_git\_url

Git fetch:

This will download only the modified file and place them on the separate branch called as remote branch we can go into this remote branch and see if the modifications are acceptable if they are then can be merged with master branch

Git fetch

To see the name of the remote branch

Git branch -a

Git checkout remote/origin/master

Git checkout master------if changes are ok going into master

Git merge remote/origin/master—merging two branches

Git pull:

This is used when there are modifications b/w the code present on the local repository and remote repository. It will download only the modified files but directly merge with the master branch

Open git bash and use command “git pull”

Git merging:

Merging of branches always based on the time stamp of the commit

1. Create few commits on the master
2. Create new branch test and add commits in this
3. Check the commit history head is pointing to d that is test commit
4. Move to master and create few more commits
5. At this point the head is pointing to f that is master commit
6. Merge test with master the head is pointing to f that is master branch

Rebasing:

This is also called as fast forward merge the commits coming from a branch will be projected as the top most commits from the master branch

1. Check the commit history
2. To rebase test with master move test and rebase, move to master and merge

Git switch test

Git rebase master

Git switch master

Git merge test

Now the head will be pointing to test that is d

Git cherry pick:

This is the feature of git which allows us to selectively pick up inly few commits from a branch and add them to the master branch. The disadvantage of both merge and rebase is that all the commits present on a branch get added to the master branch. When the developer knows that there are only few commits where working version of code is present and only those commits should be moved into the master branch then we can use cherry picking

1. Create few commits to master
2. Check the commit history
3. Create new branch test and make commits init
4. Check the commit history
5. To cherry pick only few commits to master

Git checkout master

Git cherry-pick commitid commitid2

Git reset:

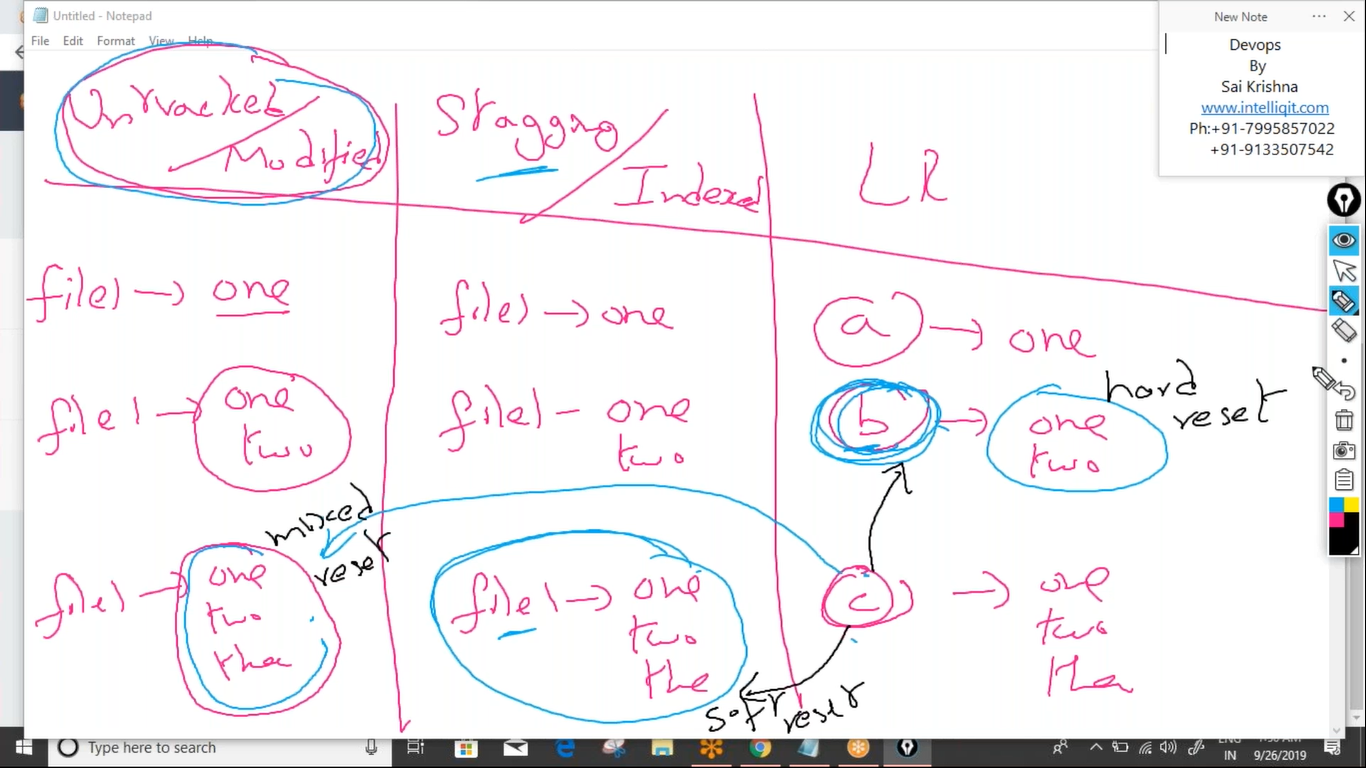
This is classified into three types

1. Soft reset
2. Mixed reset
3. Hard reset

Soft reset: in the case of soft reset git will move the control one step bck from the current commit that is we will see the files moving back into the staging area. Command “Git reset --soft commitid”

Mixed reset: in this case git will move two steps back that is the files will move to the untracked or modified section. Command “Git reset --mixed commitid”

Hard reset: in this case it will move three steps back that is it will move to the level of the previous commit. Command “Git reset --hard commitid”



Git stash:

This is a feature of git which allows the developers to leave their unfinished work in the stash section so that further commands of git cannot access those files later we can unstash them and git can start accessing it

1. To stash all the files in the staging area

“git stash”

1. To stash the files in staging are and untracked section

“git stash -u”

1. To stash the files in staging are and untracked section and .gitignore

“git stash -a”

1. To see the list of all statshes

Git stash list

1. To unstash a latest stash

Git stash pop

To unstash a older stash

Git stash pop stash@{stash\_no}

Git stash –a

.git ignore is a special configuration file which is used for hiding private files but .gitignore itself is accessible by git to make .gitignore also inaccessible by git we can use git stash -a

Git amend:

This command is used for modifying an existing commit when we don’t want to create a commit for every minor change we can use this git amend command

1. Create few files and send them to staging area and commit them
2. Check the commit history
3. Create new files and add them to b commit only

Git add

Git commit -m -amend “b”

Rearranging commit order:

The commit history of git can be re arranged so that head can point to any commit that we want can be done using git rebase command. The very first commit called as initial commit that cannot be rearranged

1. Git rebase-i HEAD~3

3 represents the we want to modify the top 3 commits

This command open the top 3 commits in vi editor where we can simply rearrange the commit order

And merging of multiple commits into older commit is called as squashing. This is useful when we want to reduce the commit history before we can push into the remote github

1. Git rebase-i HEAD~3

3 represents we want to modify the top 3 commits

This command open the top 3 commits in vi editor where we can “pick” words for commits that we want to merge and replace it with “squash”

Tagging in git:

This is a feature of git which allows us to place bookmarks on important commits. This is generally done to identify commits that are related to software releases. Tagging classified into two type

Light weight tag: this carries simply name of the tag

Annotated tag: this contains detailed information about why the tag was created when it was created

1. To create light weight tag:

Git tag tag\_name

1. To create annoted tag

Git tag -a tag\_name -m “some message”

Tags are created for top most commit. If want to create tag for older commit

Git tag -a tag\_name -m “some message” older commit\_id

1. To see list of all tags .

git tag

1. To delete tag on local machine.

Git tag -d tagname

1. To push all tags into github

Git push --tags

1. To delete tag from remote github

Git push origin :tagname

1. Git show tag\_name.Sso we can know which is annoted and light weight tag