V13

Git References

In order to remember the git commit id’s we can use these references

Instead of commit id’s we can use these names directly

Commit id’s are very larger we can’t remember them so we use references

References are pointers to commit id’s

Most recent commit id is head or master

.git/refs directory -- > In these all the references will be stored

What is master

Master is nothing but the name of the branch

It is the reference or the pointer to the last commit

Wherever we require last is required we can easily use master

Master is a file and it contains commit id

git show -- > Following command gives complete information about this commit \

git show first 7 characters of the commit id

git show master is also same as above command as we are searching with commit id previously and here, we are using master which is reference to commit id

master ~1 -- > Commit before recent commit -- > It means commit before head

What is Head? -- > It is a reference to master

Reference which is pointing to symbolic reference

Ex: Head because, master itself is reference to a pointer and Head is a reference to master so we can consider Head as symbolic reference

Detached Head -- > It means head wont point to latest commit called master and head may point to another commit, it is called as Detached head

Head information is available in .git directory

Go to .git directory and use ls command

V14

There are 2 uses of this command

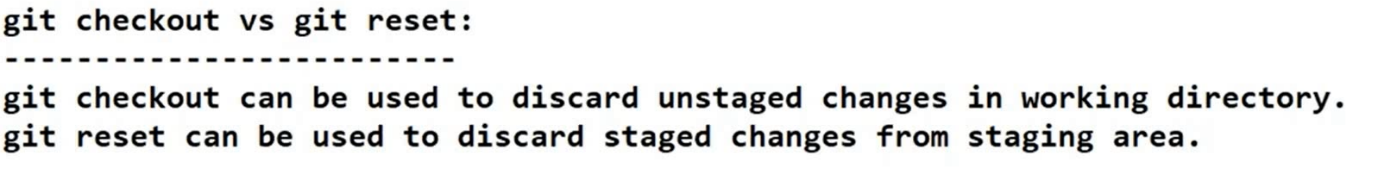
Usage 1:

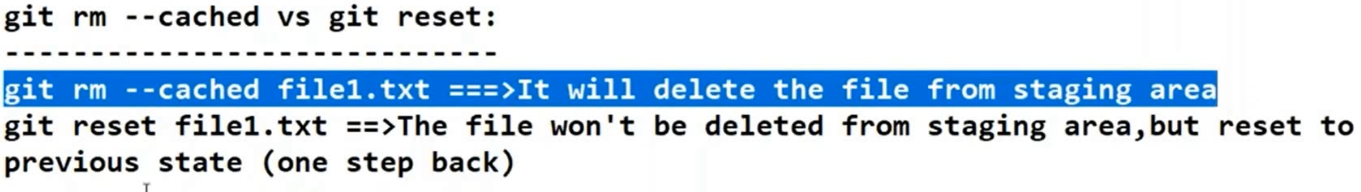
Git Reset command -- > To remove the changes from the staging area

For example, I have a file a.txt -- > I have added and didn’t commit the changes to staging

Now, I want to discard the changes from stating area

Then we need to go for git reset a.txt





V15

Usage 2:

If we need to discard the commits from local repository then we need to go for git reset

Git reset <mode> <commit id>

Here whatever commit id we specify that will become the head and above all commits will be deleted

Note: <mode> -- > It decides when the commit is deleted the related files present in staging and working directory to be deleted or not

The allowed modes are

--mixed

It is the default mode. To discard changes from local repo and staging we need to use –mixed mode

It wont touch working directory.

By mistake if we have reset the values , we can easily revert as the files are present in working directory

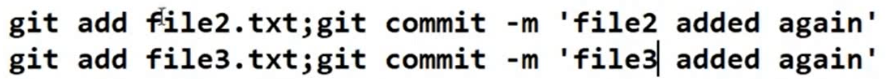
--soft

--hard

To discard a specific commit, means some commit id from the middle

This is not possible, random commit undo is not present in git

We can revert back mixed mode as we have the copy in working directory



Soft reset

It will remove files only from local repo, but not from staging and working directory

Hard reset

It will remove changes from all the places (wd , stag , local repo)

We can’t even revert back the changes as the changes are unavailable



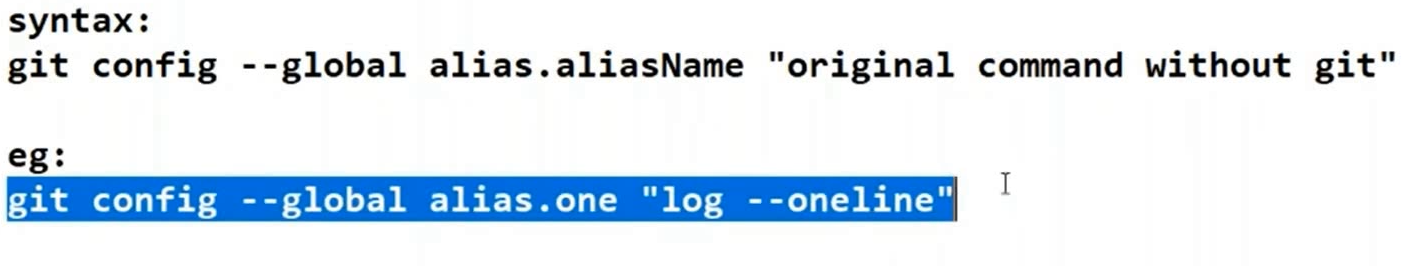
V16

GIT Aliasing

We need to know whether the alias name is available or not

Type the alias name type git1 , if git says no such command then we can consider so that the alias name is free

By using got config command we can define aliases



Where are these aliases stored?

.gitconfig -- > Following file will be there in users home directory

C:\users\oursystemname -- > Here we have .gitconfig

V 17

.gitignore -- > If we want to ignore set of files we need to create file with .gitignore and we need to specify the file names which should not be tracked

V 18

Git doesn’t consider directories. It only considers files.

Whenever we are adding files in an directory , implicitly directories will also be added

V 19 -- > Branching

Master branch is the default or the main branch in git

What is the need of creating a branch when we have main branch?

Conclusions

Child branch is created from master branch

V 20

Branching related commands

git branch -- > To view available branches \* symbol indicates current active branch

git branch branch\_name -- > To create a new branch

Ex: git branch android

git checkout command is used to the branch we need to switch

Ex: git checkout android

git checkout -b branch\_name -- > It creates new branch with the specified branch name and switches to the new branch

V 21

Branching is logical duplication of files

But there won’t be multiple copies

HEAD points to latest commit

heads is a directory which contains all the information about the branches

V 23

Merging of Branch -- > Whatever changes are present in child branch needs to be merges with main branch that is called as merging

For example we have master and child branch , we need to merge child branch changes to master so we need to go to master branch and then merge the changes

git merge child\_branch

Types of Merging -- > Fast forward, Three-way merge

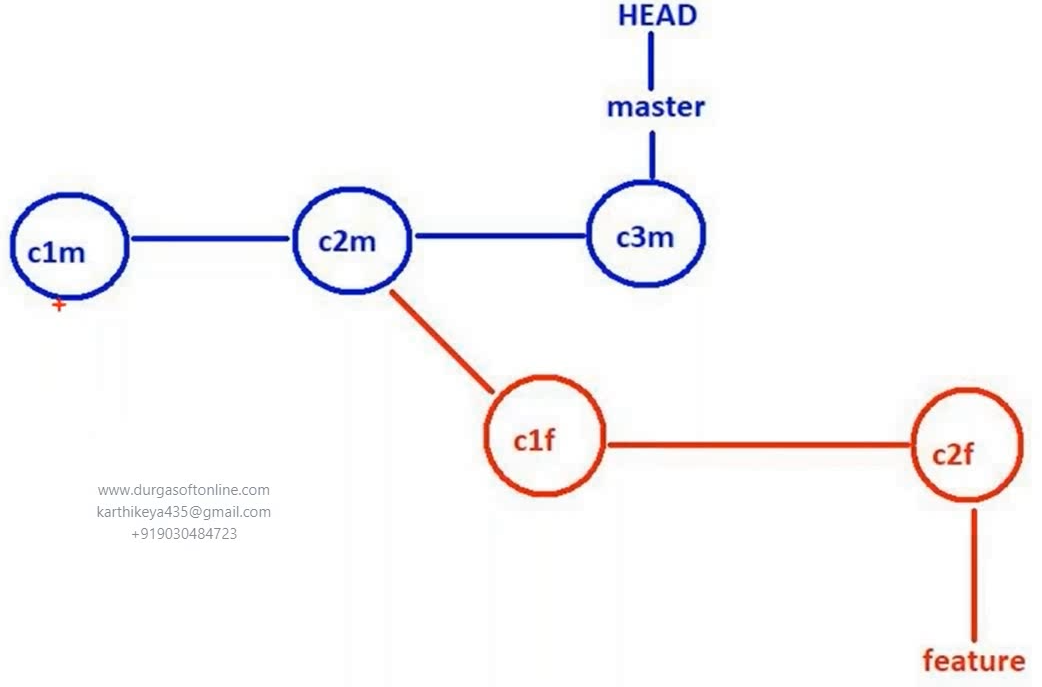
After creating a new branch from master branch and if we don’t do any changes in the master branch then git performs fast forward merge

In fast forward merge master will move to last commit

Three way merge -- > For example if we perform changes in master branch and try to merge created new child branch with master branch then we will come across Three way merge , conflict issue will come in to picture)

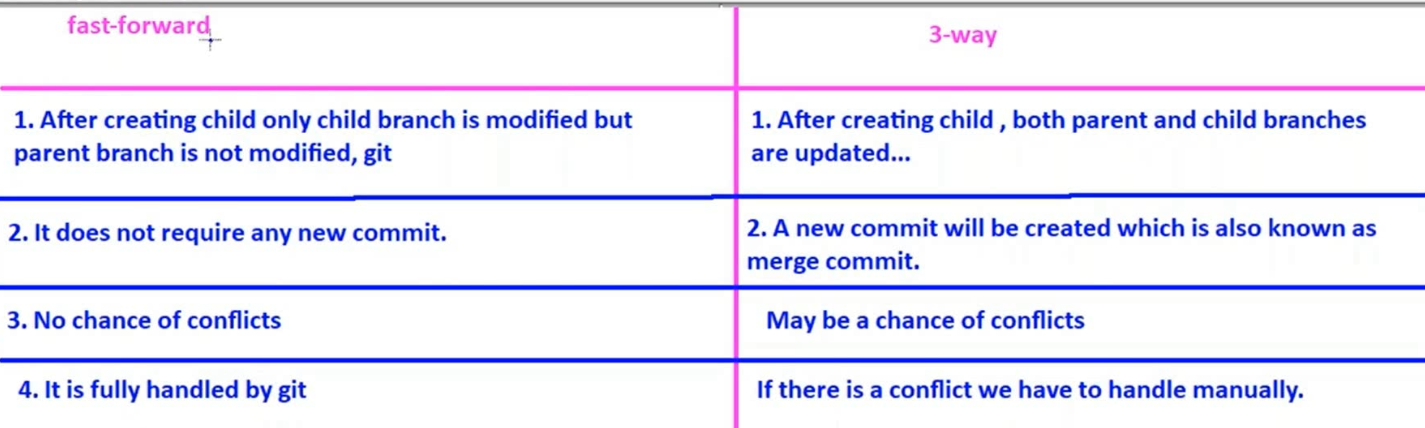
In 3 way merge a new commit will be created and HEAD pints to new commit

Before merge operation total commits are 5



After merge new commit id will be generated

Differences b/w fast forward and 3 way merge



V24

After creating the branch if updation happens in both the branches 3 way merge happens

We are aware of that in V23, now lets see the conflicts

If same file modified by both master and child branches then if we try to perform merge operations then we come across a scenario called as conflict

We need to use merge tool here

Merge tools setups are in V 10

V25 -- > Deleting a branch

Git branch -d branch\_name (branch name to be deleted)

V28

Stashing

While we are working on some work , we got some urgent work then we need to save the un committed changes from working directory and staging area changes to some location. We can do that using stash

Git stash is applicable only for tracked files

To perform git stash at least one commit should be there

git stash



Index area is staging area

We can stash multiple times git stash list

git show stash@{0}

If we want to know what are the changes that are present in particular stash

Un-stashing - -> Bringing back the stashed changes to working directory

We can perform stashing in 2 ways

git stash pop stash@{0}

It will bring stashed changes from temporary location to WD , corresponding entries will be deleted

git stash apply stash@{0}

It will bring stashed changes from temporary location to WD , corresponding entries wont be deleted and we can use these stashes for later use

After un stash all the untracked changes will come to working directory

V 29

Partial stash -- > I want only few files to be stashed but not all files then I will go for partial stashing

git stash -p -- > This command will ask for confirmation whether to stash or not for each and every file

V 30

How to Delete the stashes -- > git stash clear (Deletes all the available stashes)

How to delete particular stash -- > git stash drop stash\_id

V 31

Remote Repository

When we want to share our code with other developers then we need to go for remote repo

Direct communication problems within peers

On every peer system git server is required else the developer cant share code with other developer

If dev A wants to communicate with dev B then dev A should be aware of host name or the ip address of dev B , dev A should be aware of the dev B port number, on which port number git is running on dev B system

Tomorrow Dev B configurations like host name , port num and other things may change that is the reason why we are going for common repository

V 32

Remote repositories are used for sharing our code

Git Hub is one of the remote repositories

How to create free account in git hub

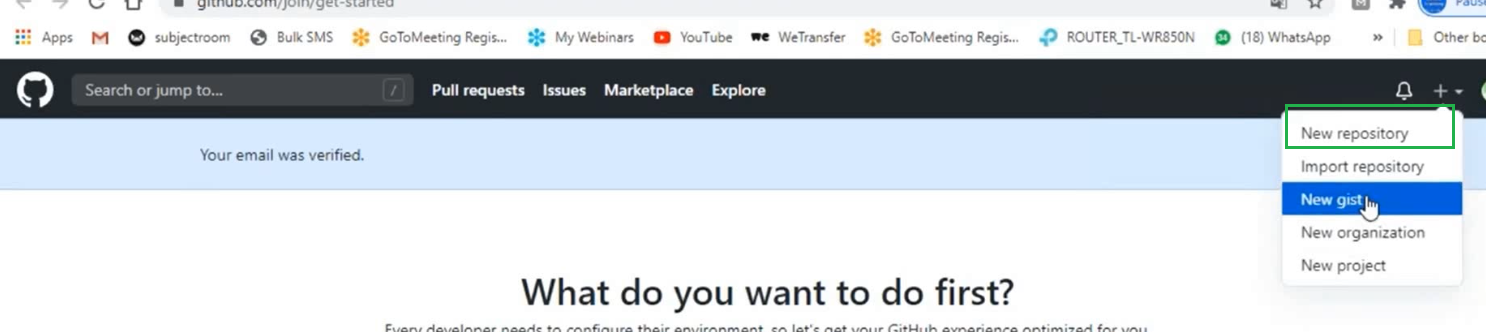
Sign up and create account

<https://github.com>

username is very important while signing up

How to create remote repository in git hub

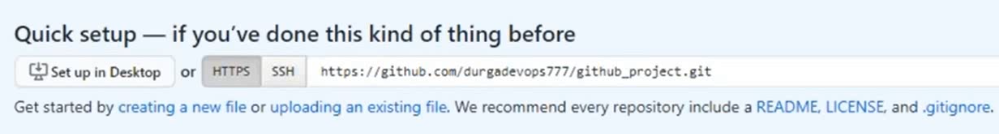
Create a New repo



Public means read access for anyone but no write access

Private repo means read/write can be done only by few people

We require remote repo URL to access remote repo from local repo



How to work with this remote repository

We have to connect local repo to remote repo so that we can share files

To do that we need to configure following things

git remote -- > git remote add alias\_name remote\_repository\_url

We need to specify alias name as we cant use the same remote url every time

Once the alias name is configured, we need to use alias\_name instead of remote repo url



git remote -- > Gives us all the list of remote URL’s alias names

git remote -v -- > Gives the URL information as well

git push -- > I want to send the changes from local to remote repo we use git push command

git push <remote repo name> <branch name>

git push origin master

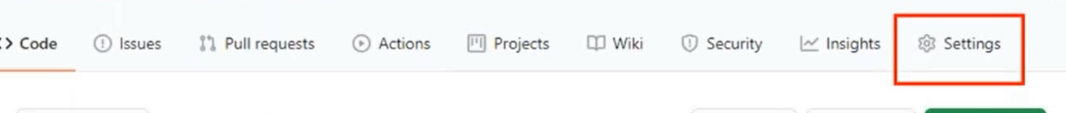
For free account total repo size is 2gb

Max allowed file size – 100MB

We can work with multiple remote repos simultaneously

V 33

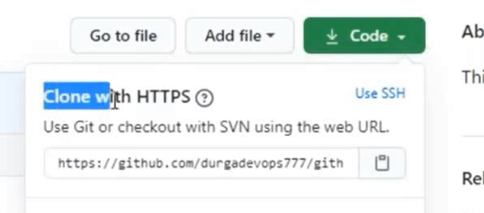
Using settings option, we can rename or perform required changes to remote repository



git clone -- > we can clone complete remote repo to local repo

git clone remote repo url

We can get clone url from here as well



git clone remote repo url my\_project -- > Using this we can create our own project name

We don’t need to use gitinit command before closing because git clone command itself is responsible to create local repo

V 35

Git tagging -- > git tag -- > To define static references we need to go for git tagging.

It is a label or mark to a specific commit in our repository. Generally, we use tags concept for release purpose

There are 2 types of tags 1) Lite weight tag 2) Annotated tags (tags with information)

Lite weight tag - -> git tag <tag name> - > Label acts as a tag for latest commit as we haven’t specified the commit id

git tag –list -- > Following command gives us the information about list of tags

Where tags will be stored

.git/refs/tags - -> All our lite weight tags information will be stored here

Is it possible to delete a tag? git tag -d <tag name>

git tag –delete <tag name>

With in the repository tag names should be unique

V 36

Annotated tag -- > We are going with this as we require more information about the tag created

Annotated tag internally implemented as object

git tag -a <tag name>

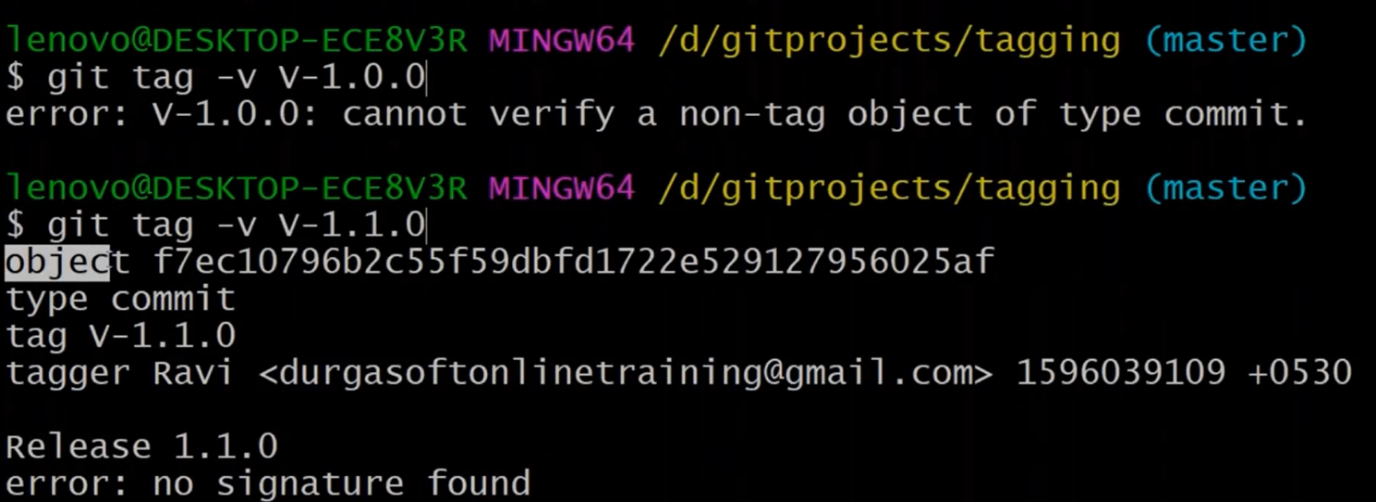
We need to create annotated tag with message as follows

git tag -a <tag name> -m ‘message needs to be typed here’

git tag -v <tag name> - -> Provides information about annotated tag

In the first command we got error because we are trying to get information about lite weight tag

-v option will work only for annotated tag



Lite weight and Annotated tag save location

Annotated tag will also be saved in .git/objects as it is internally saved as object



Hash of tag object



V 37

How to tag previous commits?

git tag -a <tag name> <mention the commit ID for which we want to tag> -m ‘message information’

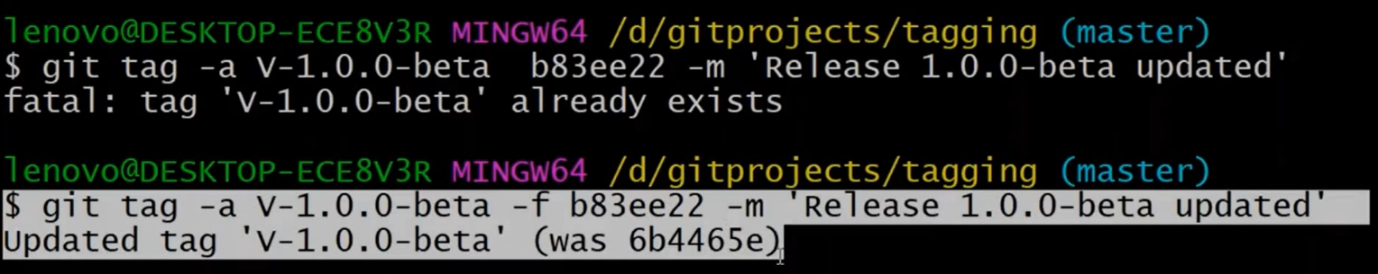
How to update existing tag?

We can perform it in 2 ways

Delete the tag and recreated the tag with correct commit id

By using -f option or –force option we can replace or update the existing tag without deletion

git tag -a <tag name> -f <correct commit id> -m ‘commit message’



For the same commit can we use multiple tags

Yes, we can

Tags can be used for getting difference or comparison b/w 2 commits

V 38

How to push tags to the remote repository

Whenever we are trying to push our code to remote repo tags wont be pushed to remote repo

We must push them separately

How to push a single tag to remote repo

git push origin <tag name>

How to push all tags to remote repo

git push origin master – tags

How to delete tag from local repo

git push origin :<tag name>

V 39

git reset command is destructive command and not recommended to use in public repos

git revert -- > Alternative to git reset

because git revert won’t delete commit history

git revert c3 -- > It will create new commit with previous commit c2 and commit ids of c3 and c4 remain and c2 will become latest commit



V 40

Cherry picking -- >

`