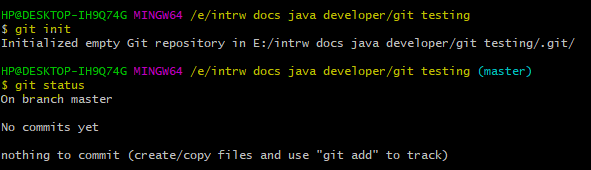
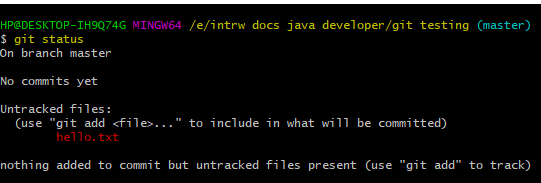


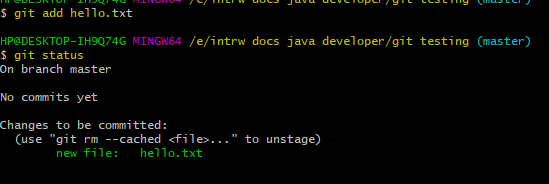
* **Git init** - for starting with new project
* **Git clone** – take an already exisitng project from repository and work on it

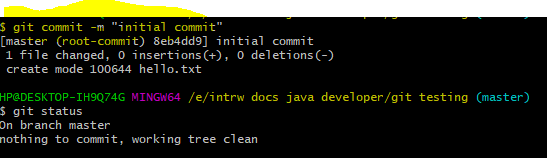


* **Git status** - to get current status in branch

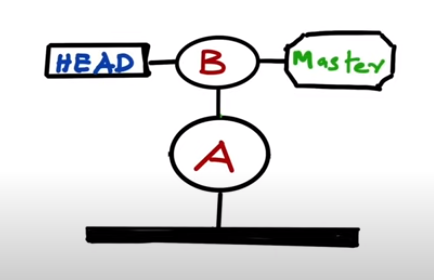


* In git everythng is on versioning. If we want to do versioning there is 2 steps.
* **Git add filename** -this will add that file to the cache of git (to add all files we can use **git add --all**)
* **Git commit –m “message”** - this is used for committing that change to the repo and version has created after this step.In real time projects after each module (eg:login) we will do a commit so that versioning will be created in our project.





* On every commit a commit id will be created.
* Head is label
* Master is the default branch
* Head will move on each commit to the new commit id’s
* Master branch also will move to the latest commit.

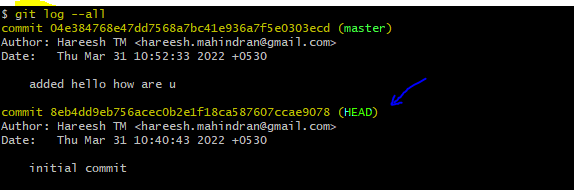


* **Git log** - this will show all the commits in our current commit id .Will get to know where head label and master branch is pointing.
* **Git log –all** will show all commits in all the commit ids(branches)



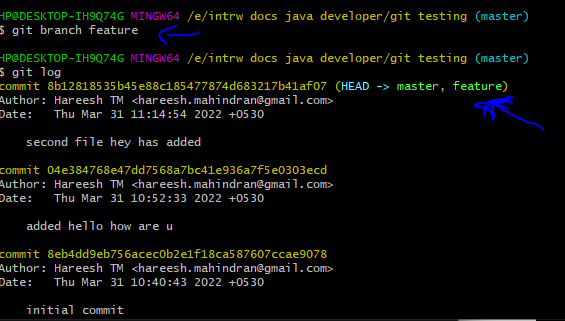
* **Git checkout branchname/commitid** - this command will be used to move to specific branch or commit id

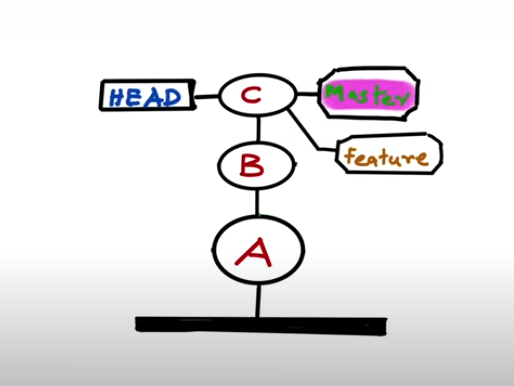
Head will move to that specific comit id after this operation.



**Branching concept**

* **Git branch branchname** - used to create a branch
* Note: Head will be pointed to master only. But we have created another branch called feature in that commit.
* See below screenshot:

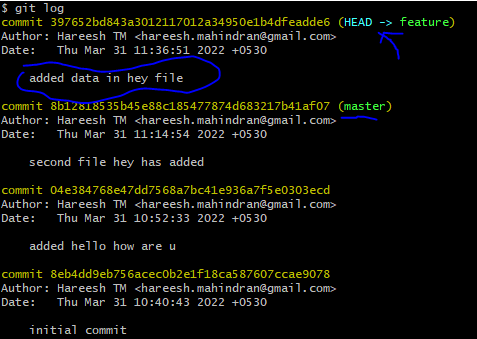


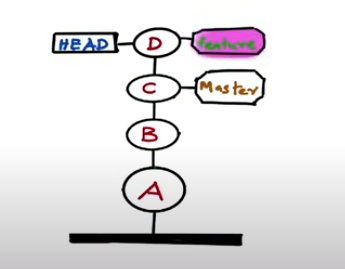


* **Git checkout branchname** -used to checkout to a branch



* Note: Now if we do any commits it will happen only in the feature branch not in the master branch. That is the difference.

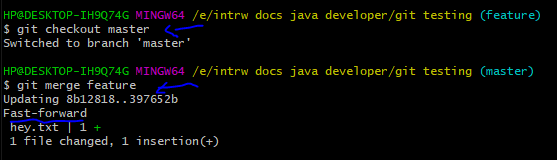


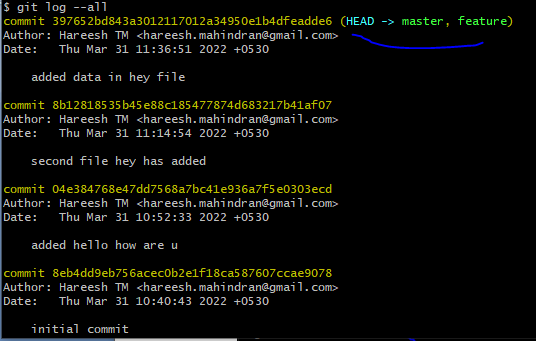


**Merging concept**

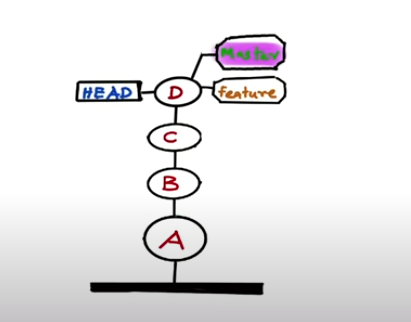
Eg: Merging feature branch with master branch

* First checkout to master
* Then use **git merge feature**
* This is fast forward merge : here in this kind of merge the labels will be placed , no need of content copying.

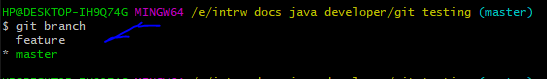




Fast forward merge representation:

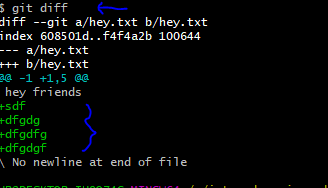


* **Git branch** - to know in which branch we are .also lists all branches.



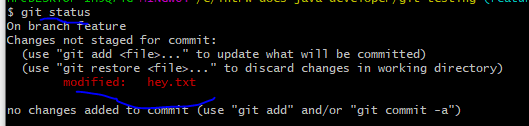
* **Git diff** - to know the changes made to the files

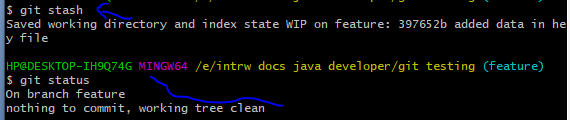
Eg: adding some text to hey.txt and using this command.



If we don’t want those extra coding we have done in the file we can use

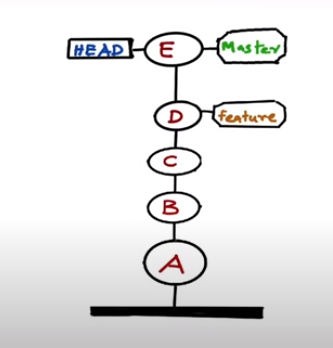
* **Git stash** - this command will revert the code to the old commit state.



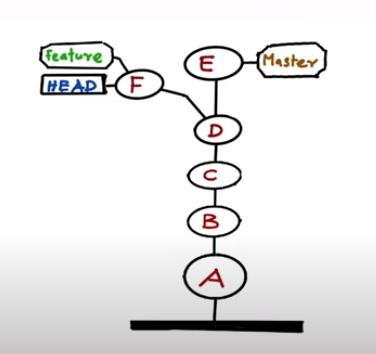


Another Merging concept: Recursive merge

* Modifying hey file by master branch
* Modifying hello file by feature branch
* Then merging both the branches together.
* Below represents : after checking out to master modified hey file and committed

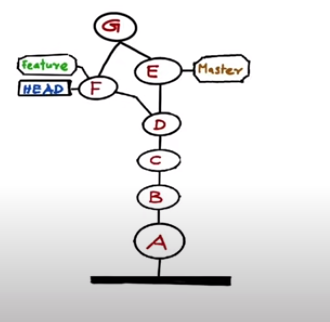


* Now checkout to feature and modify hello file
* Below represents : after checking out to feature modified hey file and committed



* **Git log –all –decorate –graph**  - to see structure of the current repo
* Now we need to merge the code in both feature and master branches which are staying independently.
* In this case we will do recursive merge.
* In the case of recursive merage a new commit will be created in order to merge 2 branches.

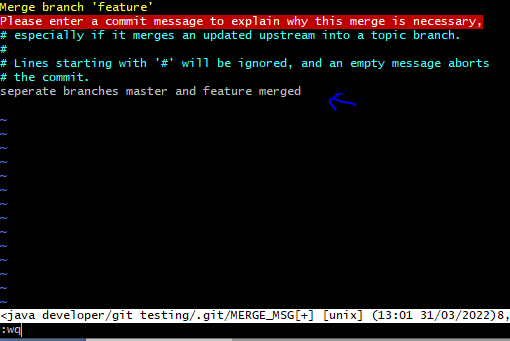
Pictorial Representation:



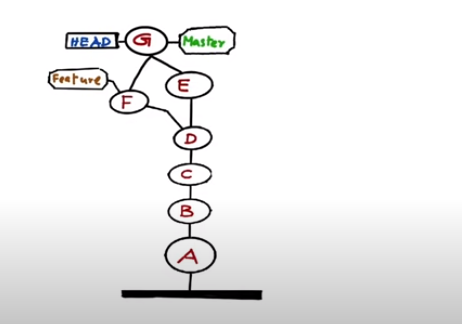
* We can merge either from master or feaure

Eg:merging from master

**Git merge feature**



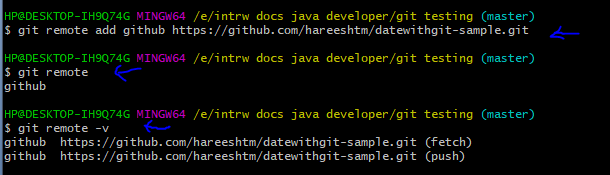
Representation after rec merge:

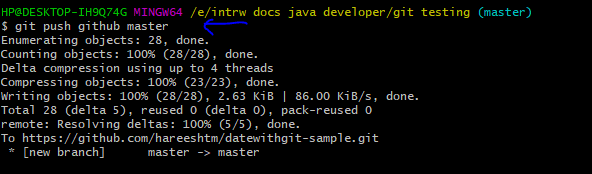


**Git Remote ----- Communicating with server**

git remote set-url origin

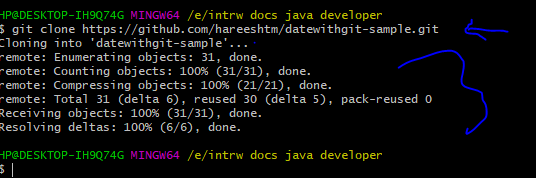
* Git remote add “some name” link -- this is used to link a remote repository in our system.
* Git remote -- will show how many remote repostories are linked in our current repo of system.
* Git remote -v -- to see the links associated to each remote repo.
* Git push “some name” branchName -- for pushing local repo code to remote repo.





**Cloning a project from remote to local**

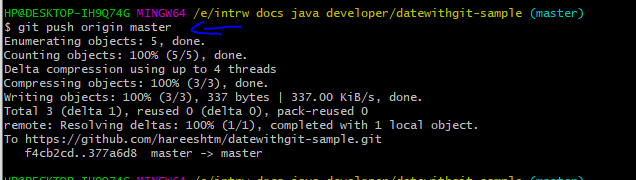
* Git clone link -- for copying code in remote repo to local.



Note: if we are cloning a repo then the name of the repo by default will be origin.



After cloning by making modifications to the code we can push the code as well.



* In real time projects the best practice is after cloning a branch will be created and developer will make changes to that branch only.
* After that the code changes has to be bought in to the server(remote repo).
* Before pushing that code develpoer need to check any other update in code is done in that remote repo by other developers.
* For that we need to fetch the code and merge the code with the master branch of our local repo.
* We we do fetch only then the remote repo code will come to origin/master. So seperately need to merge it with our master branch of local repo.
* So in order to do both these operations we can use **git pull** command.