# **DEVOPS GIT HANDS ON ASSIGNMENT**

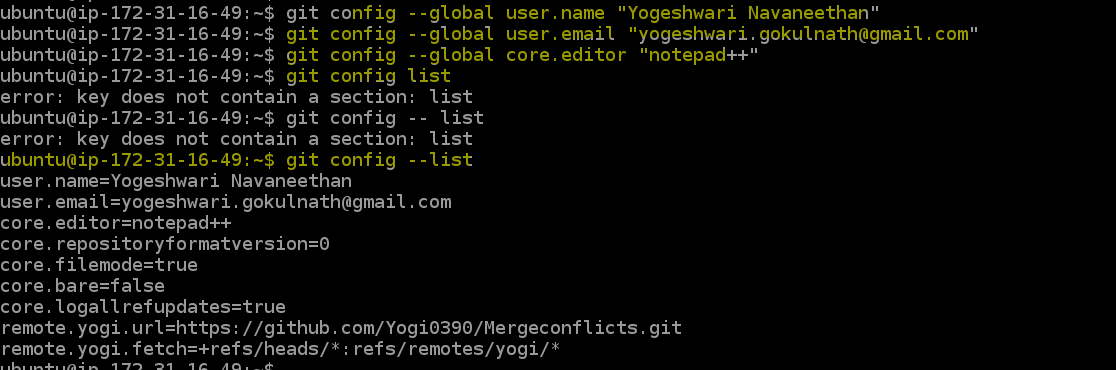
1. Set the global configuration file with your user name, email and editor as Notepad++. List all the properties which you just set.

Git config - - global user.name “Yogeshwari Navaneethan”

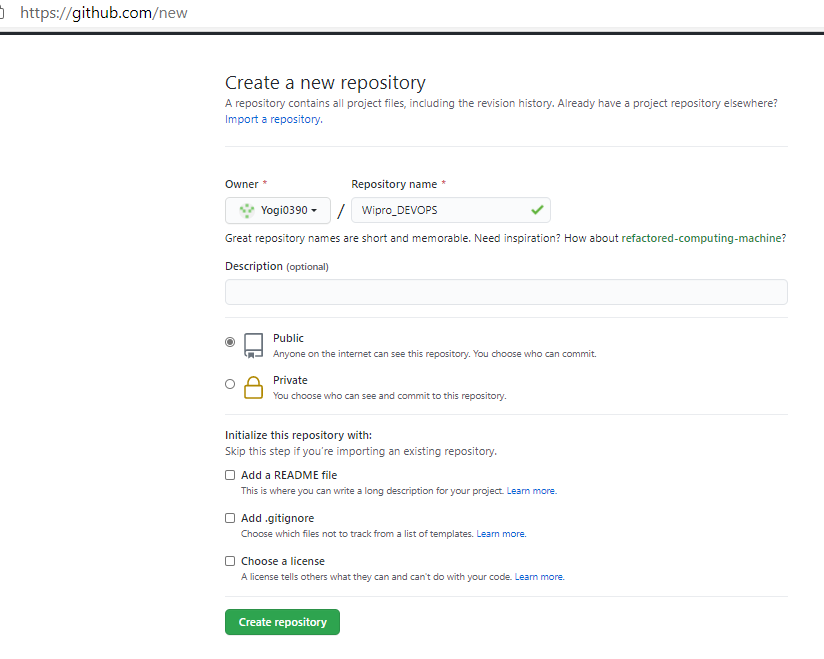
Git config - - global user.email “[Yogeshwari.Gokulnath@gmail.com](mailto:Yogeshwari.Gokulnath@gmail.com)”

Git config - - global core.editor “Notepad++”

Git config - - list



1. Make a fresh Git project



Created a new project in local repository.

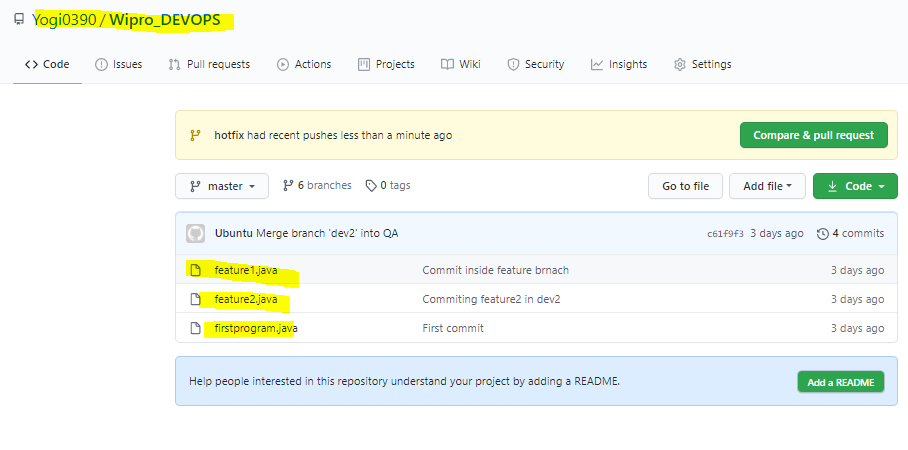






1. Create a Github account (Or use the account if already registered). Clone a project from the remote repository to your local repository

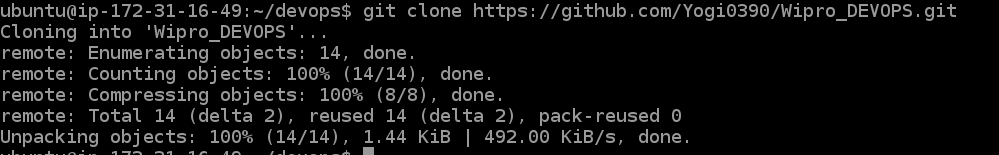
**Remote Repo Projects:**



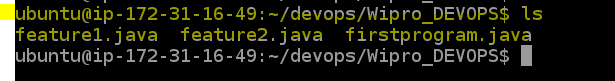
**Cloned to local repo:**



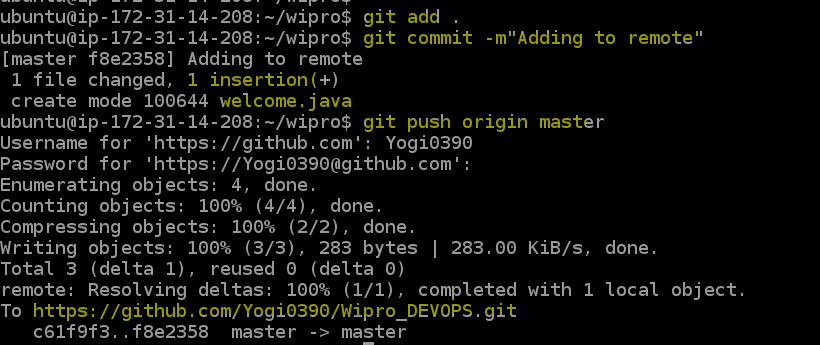
**After cloning from remote repository:**

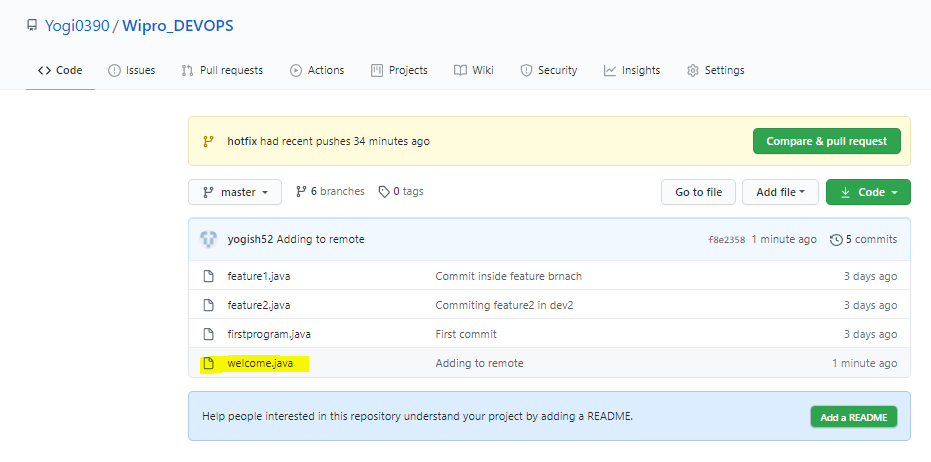




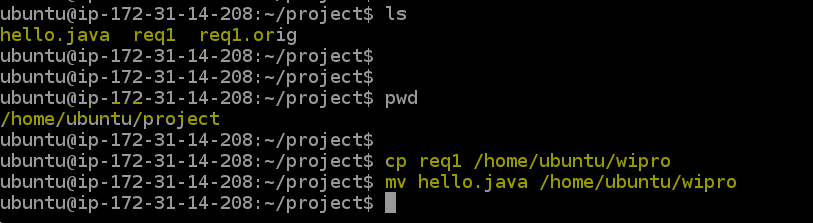


1. Push the project created in assignment 2 to the remote repository.





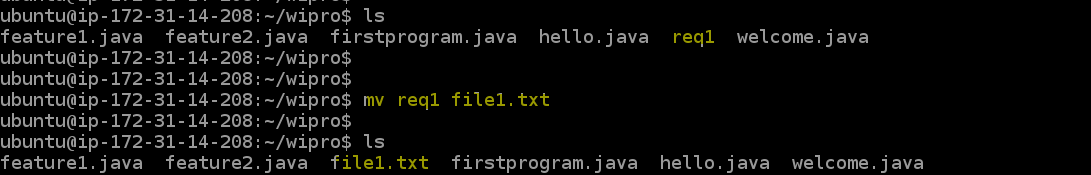
1. Use the different ways of renaming and moving files



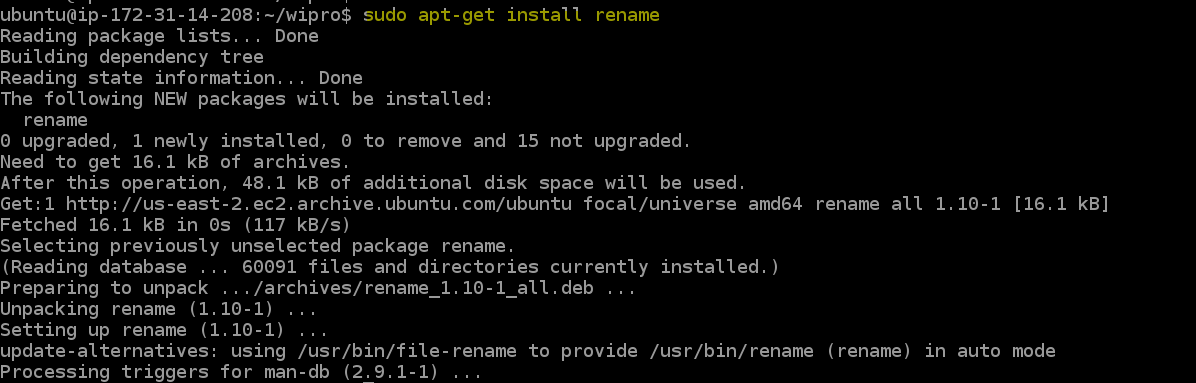
**Check in Wipro folder after moving a file from project folder**

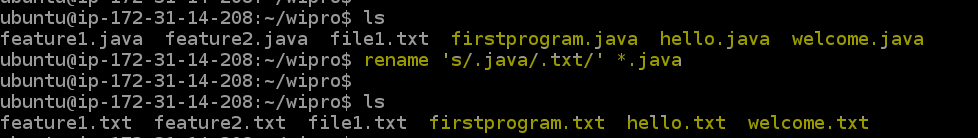


**Renaming a file:**

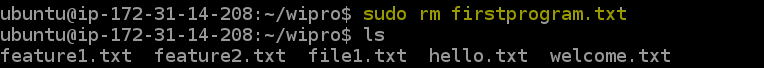


**Renaming using rename command:**

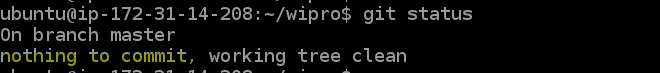




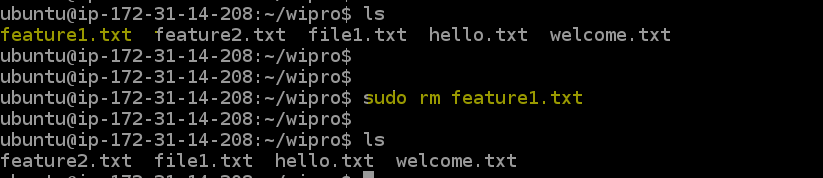
1. You just created a new file, but then you decided that the file is to be removed. How do you delete this untracked file.



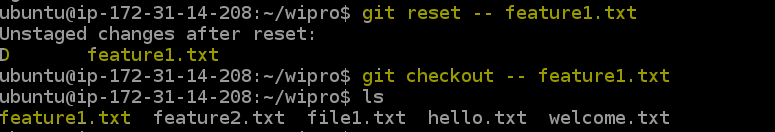
1. Demonstrate the following:
   1. delete of a tracked file
   2. backing out staged deletion
   3. recursive deletion



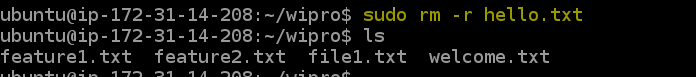
* delete of a tracked file



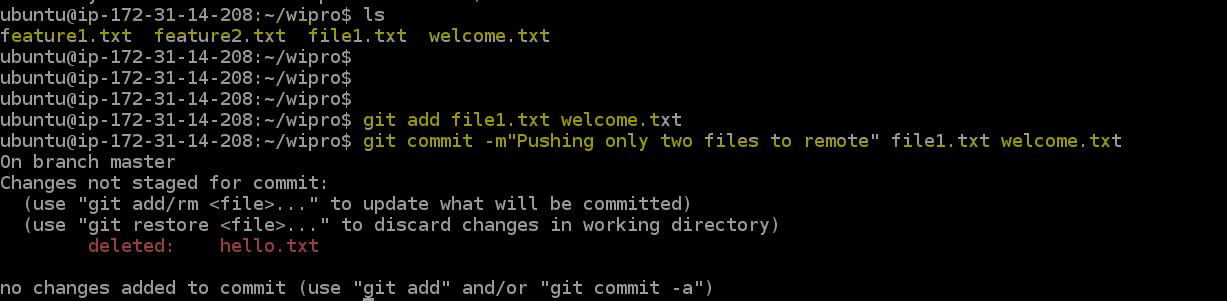
* backing out staged deletion



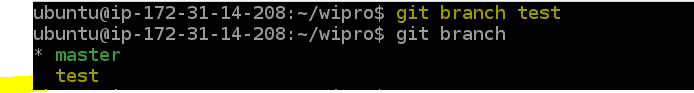
* recursive deletion



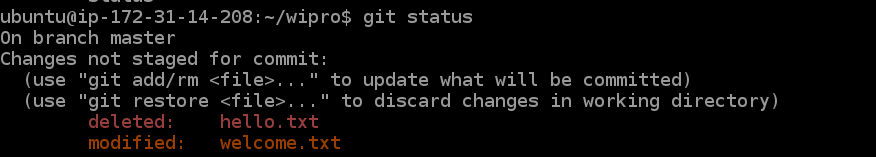
1. You do not want to push certain folders/files of your project; how do you manage this?



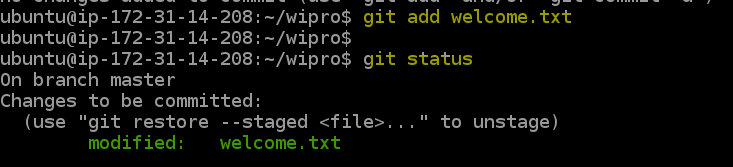
1. Create a branch called “test”. Make some changes in the master branch. Let there be some changes in the working directory and some in the staging area. Make some changes in the test branch as well. Issue the command to show the differences for



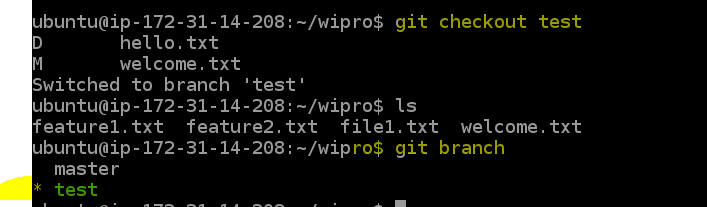
Modified welcome.txt file in master



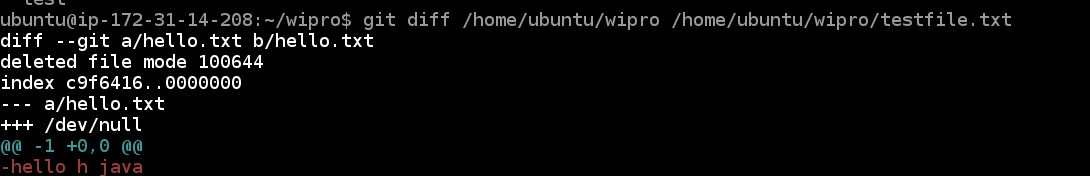
Modified in staging area



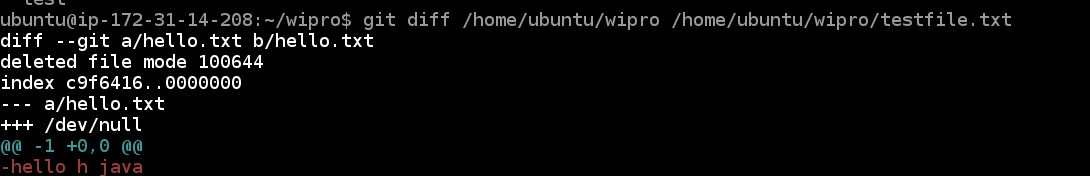
Modifying in test branch



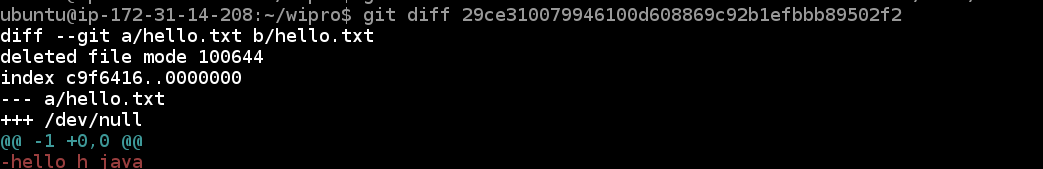
* ***Working directory vs Staging area***



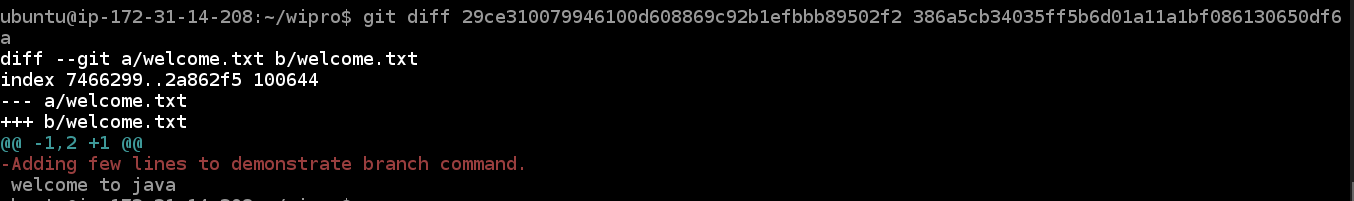
* ***Working directory vs Local Repository***



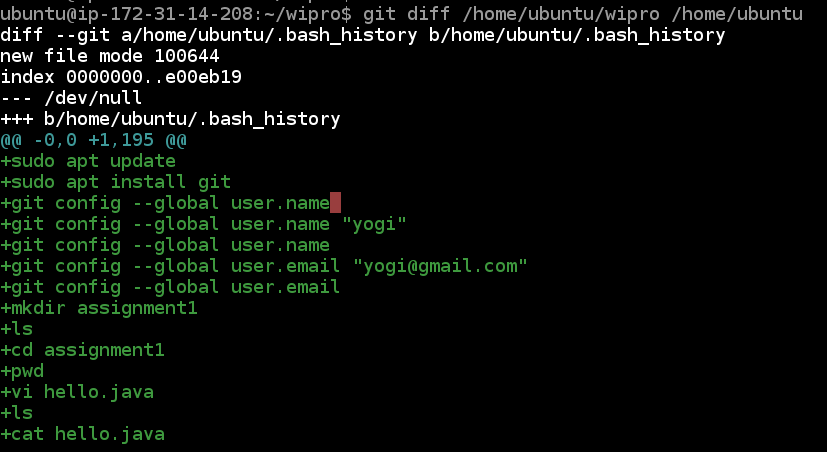
* ***Staging area vs Local Repository***



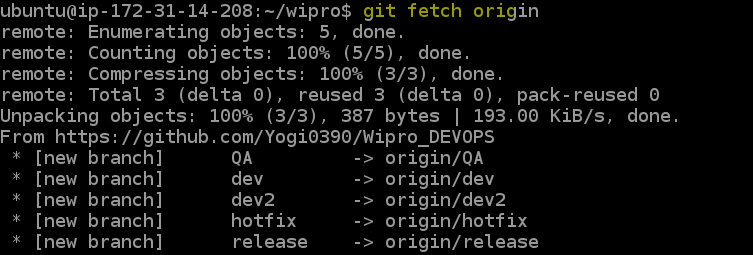
* ***Between two commits***



* ***Between two tags***



* ***Local vs Remote Repository***

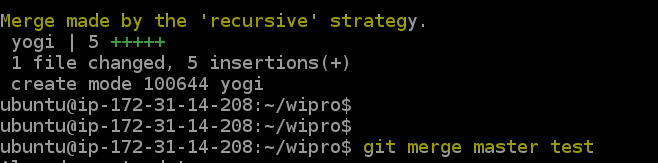


***Master branch vs test branch***



1. Merge the changes from test branch to master branch.
   1. FastForward merge







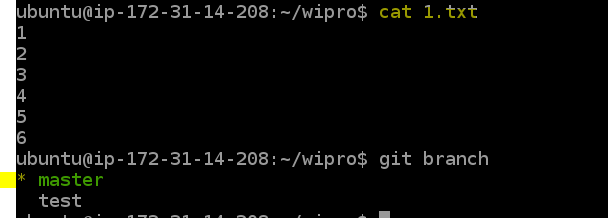
* 1. Disabling FastForward merge

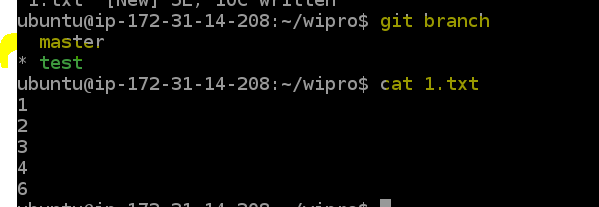


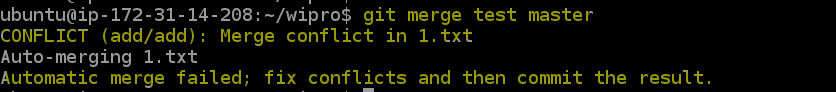
* 1. What is the difference between option a and option b

The --no-ff flag prevents git merge from executing a "fast-forward" if it detects that your current HEAD is an ancestor of the commit you're trying to merge. A fast-forward is when, instead of constructing a merge commit, git just moves your branch pointer to point at the incoming commit.

1. Create a merge conflict situation. Resolve the conflict and merge the changes between the branches.

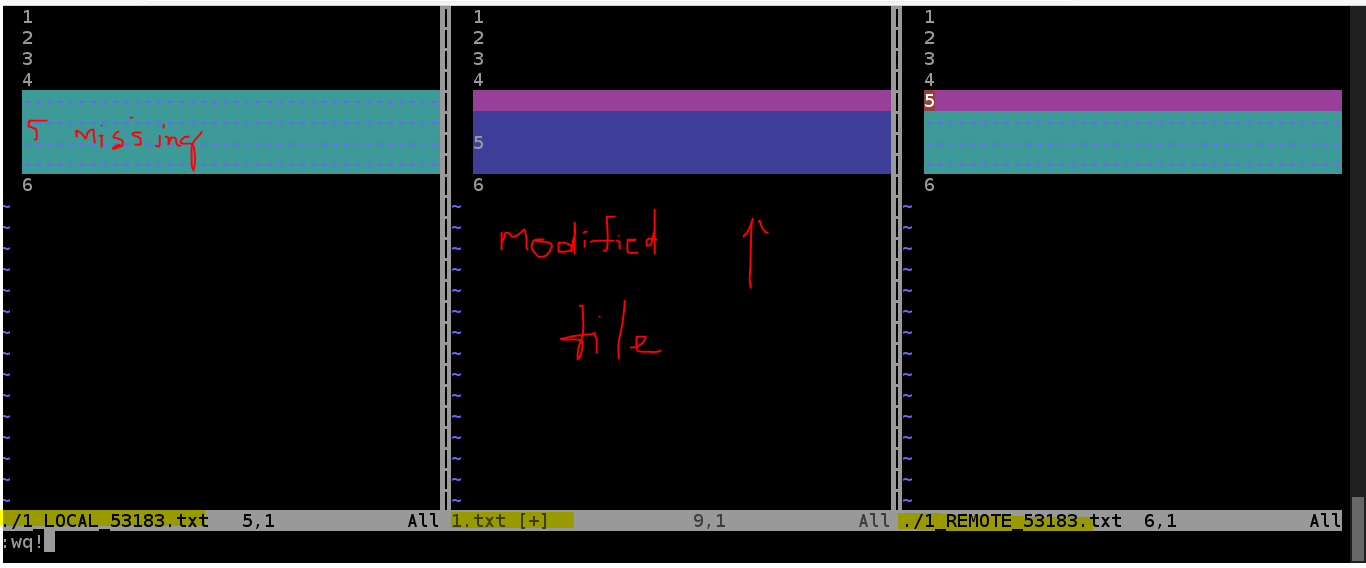


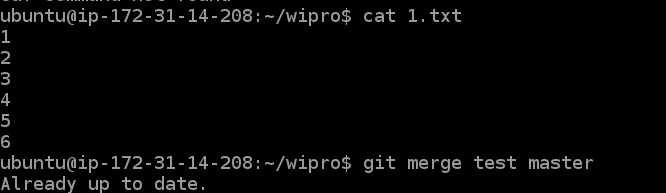




**Resolving merge conflict:**

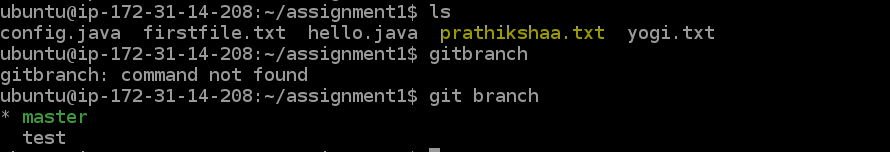


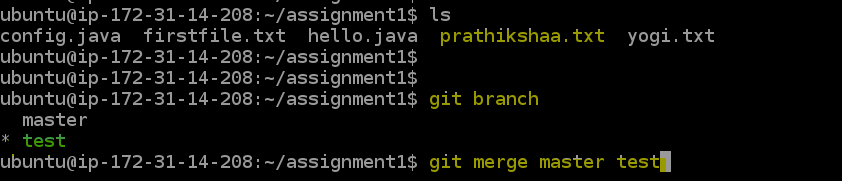




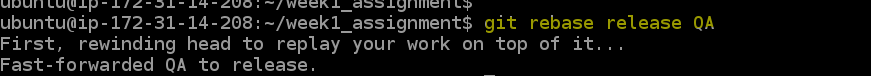
1. What is the difference between merge and rebase, demonstrate with the eg.

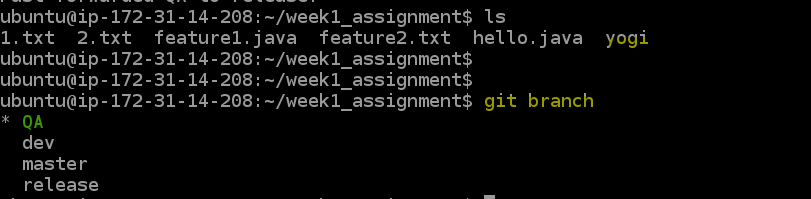
**Merge:**





**Rebase:**





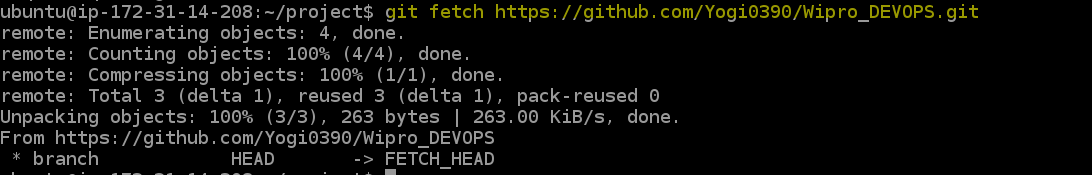
**DIFFERENCE:**

* When you do rebase a feature branch onto master, you move the base of the feature branch to master branch’s ending point.
* Merging takes the contents of the feature branch and integrates it with the master branch. As a result, only the master branch is changed. The feature branch history remains same.
* Merging adds a new commit to your history.

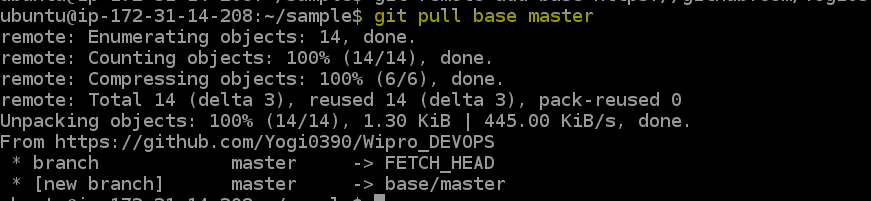
1. With an example, demonstrate fetch, clone and pull. What is the usecase for these operations. Are they same, different? Explain

**FETCH:**





**PULL:**



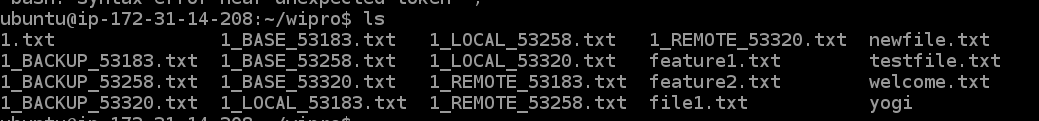


**CLONE:**

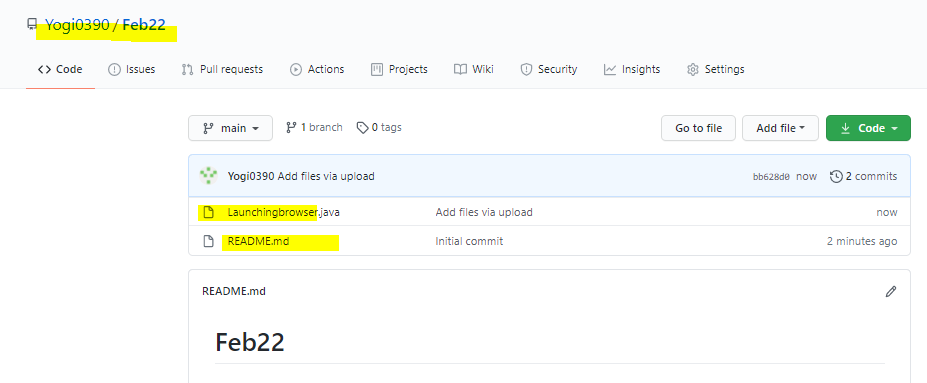


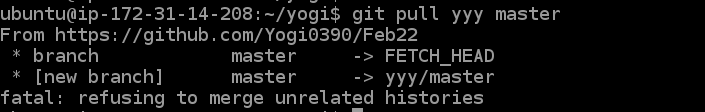
1. Create a new repository in GitHub, with a README file. While pushing to the remote repository, if the remote branch is ahead of the local repository (new file is added in remote repository, which is not there in local repository) and pull is failing, how do you solve this problem?

**LOCAL :**

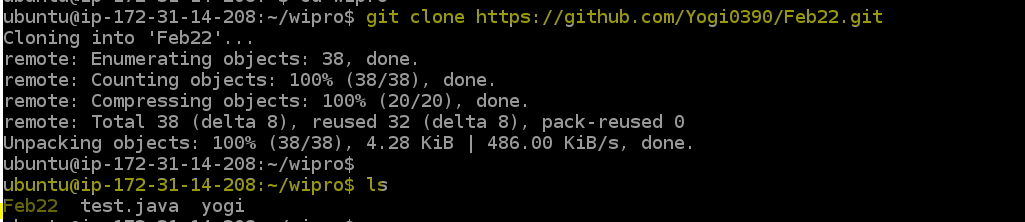


**REMOTE:**

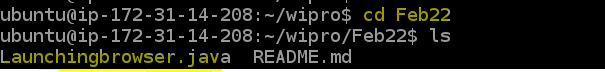




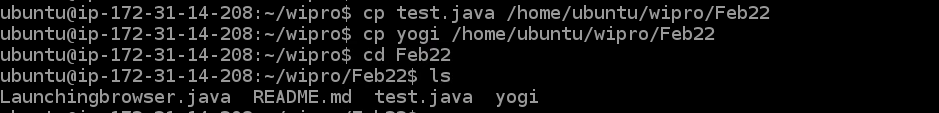
**Make a clone of the remote repository to local**



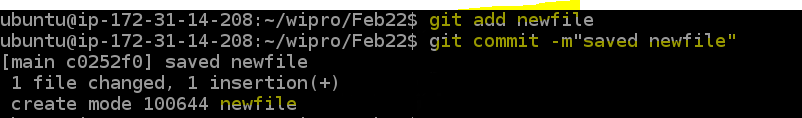
**Goto cloned folder**

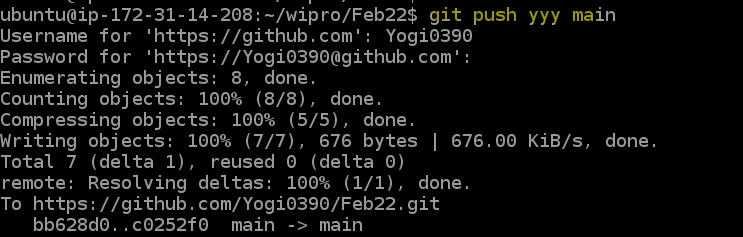


**Copy contents of working directory to cloned dir**



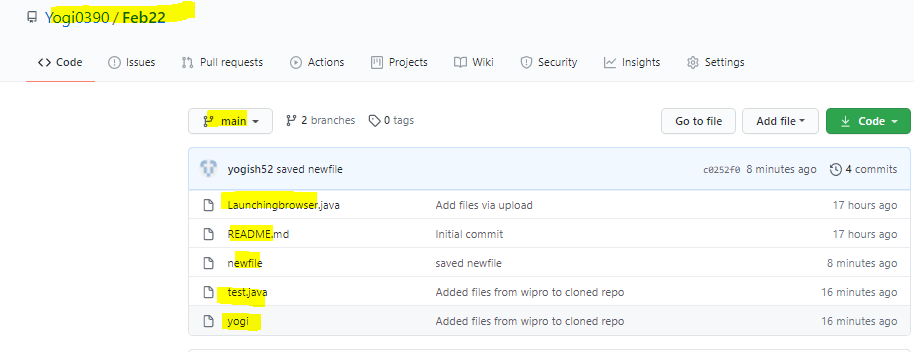
**Now add any new file if you want**







**Verify in remote repo now**



**Hence this is one of the simple method to resolve if this kind of pull request issue comes.**