**8218-Arsh Nadeem-Introduction to Version Control-Jan-2025**

1. Git Setup https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html

-> using command : git -- version

On hitting this if git is installed, the current version will be displayed

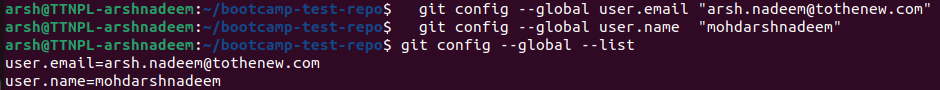
If not install



-> installing git for demo purpose

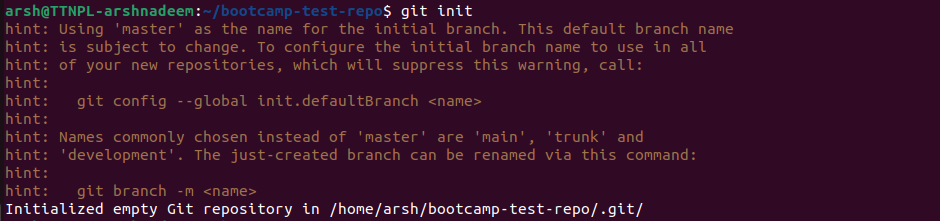


-> configuring git & listing the configuration



2.Initialize a Git Repository

-> using git init command for initializing empty git repository

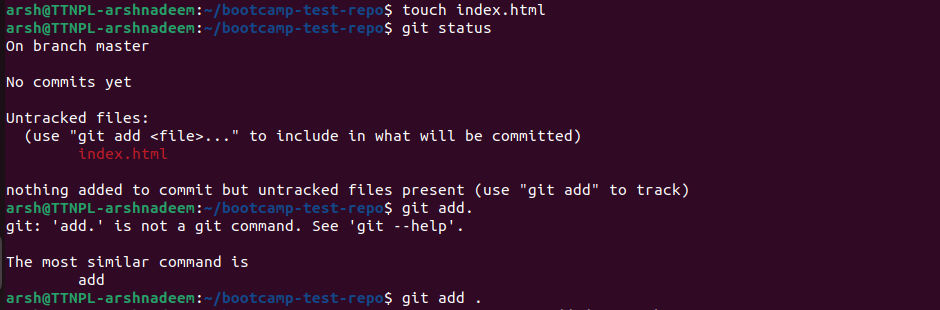


3. Add files to the repository

->this can be using add command

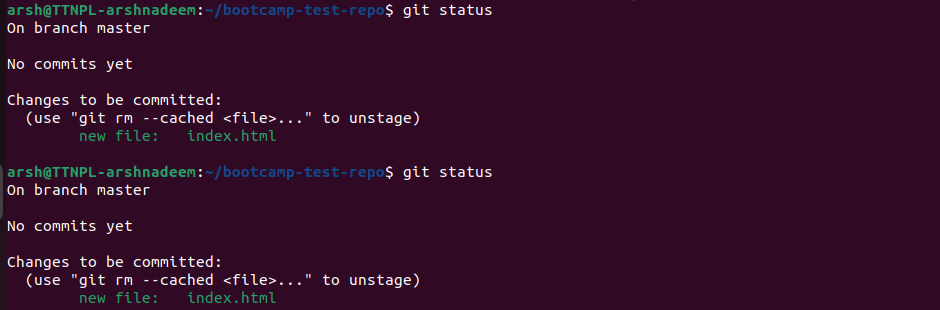
For all files use git add .

For single file use git add fileName.txt



-> checking status

Using git status



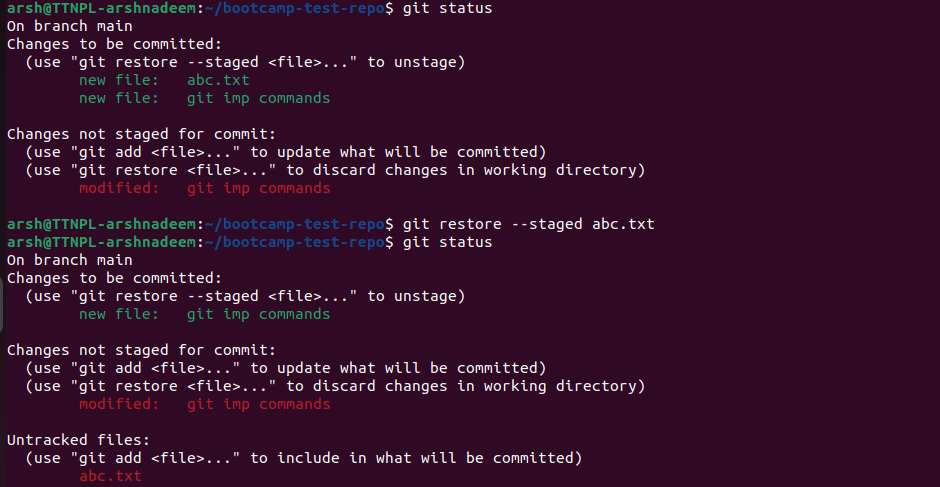
4.Unstage 1 file

-> this can be done using restore command and reset command

but there is a difference.

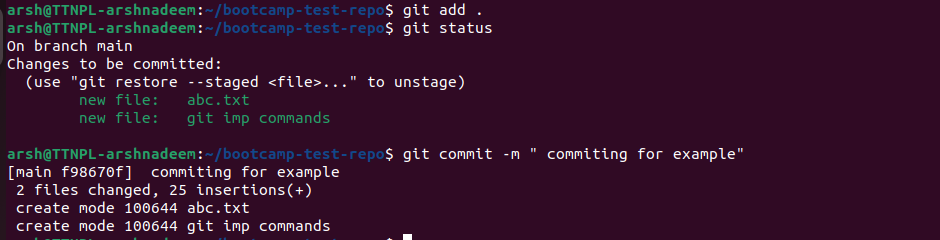
-> generally git restore is used for unstaging.

-> reset is used for commit level operation.



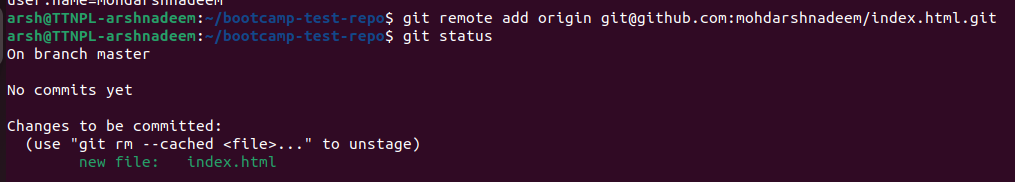
5. Commit the file

-> can be achieved using git commit command



6. Add a remote

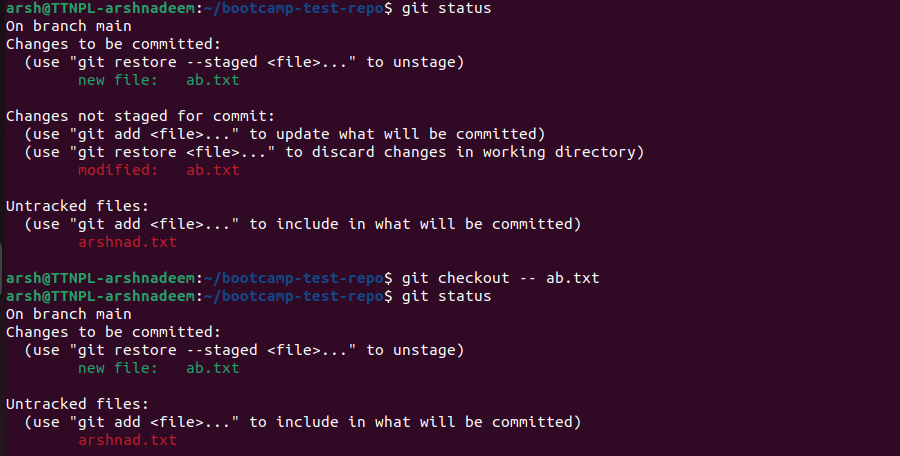
-> this can be done using git remote add origin (url)



7. Undo changes to a particular file

Using ‘git checkout’ for files that are modified but haven't staged yet or committed yet

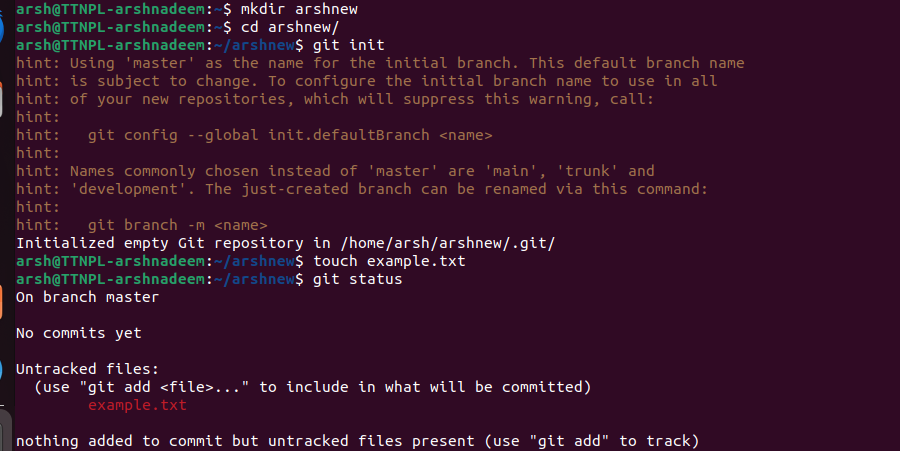
Checkout is also used to create new branch and switch between branches



8. Push changes to Github

-> Making a new directory and creating an initial git repo inside it using ‘mkdir’ and ‘git init’ .

-> creating a file to push on remote repo that is created on github



-> adding this file to stage for committing i.e git can’t commit it until it is added using ‘add’ command.

-> now commit changes

-> we are on master and want to push on main

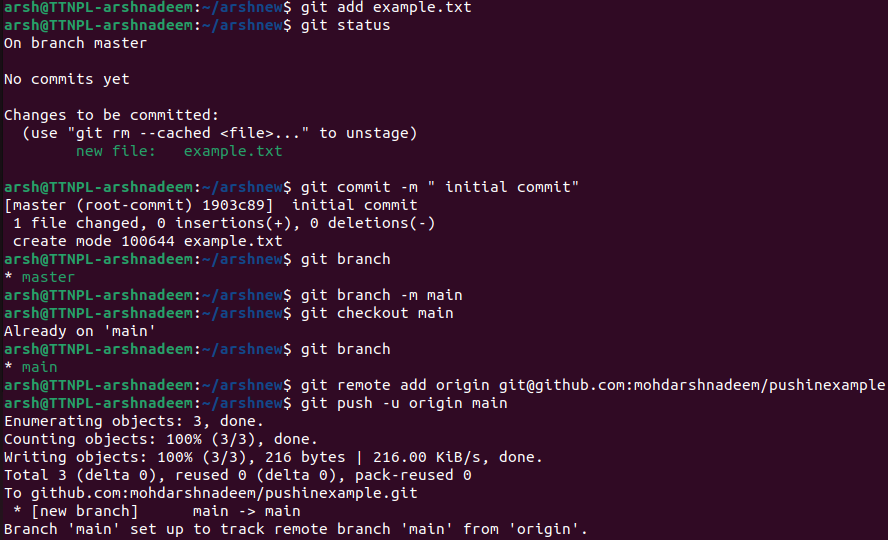
-> so changing the branch using ‘branch; or ‘checkout’

-> to push our file on main we need an access.

-> we can get it via the ‘remote add origin’ command.

-> and pasting the ssh url in front of it.

-> once the access is provided or link is established we can easily push or file using push command

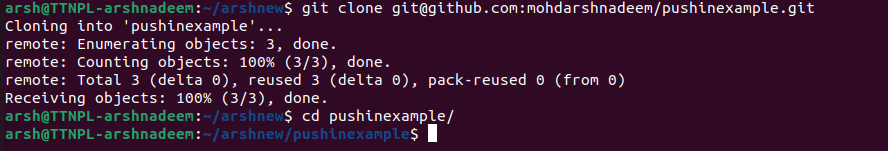


9. Clone the repository

-> cloning basically means creating a local copy of remote repo in local machine.

-> using git clone

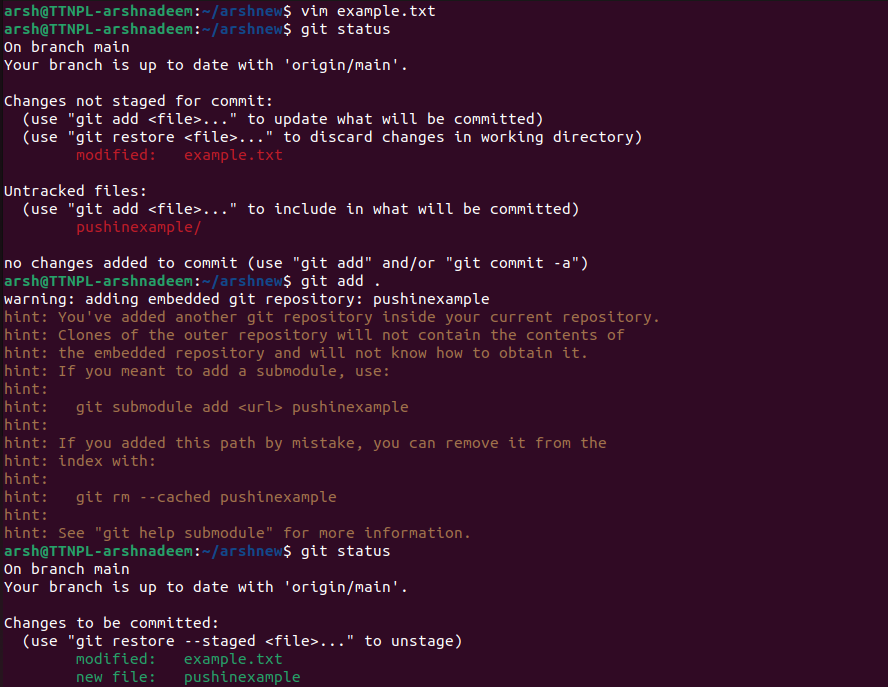
-> verifying using cd command



10.Add changes to one of the copies and pull the changes in the other.

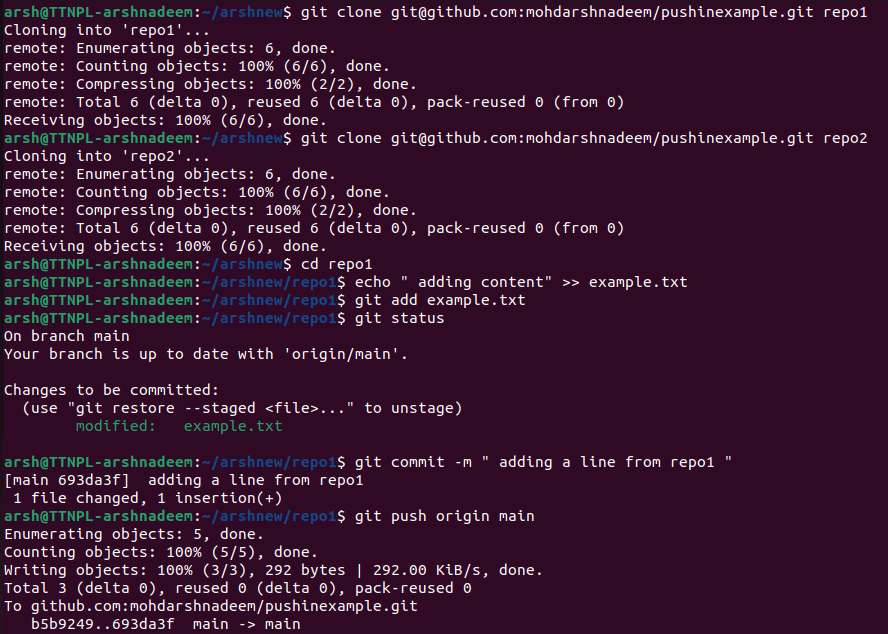
-> adding some content to the file because it was empty.

-> committing and pushing on remote

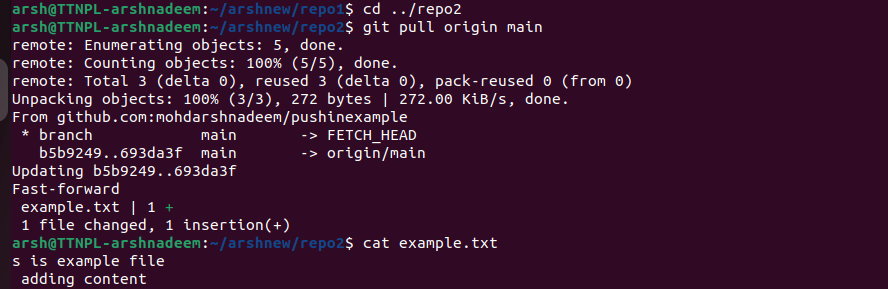


-> cloning the remote repo in local files

-> adding content on one file and pushing to remote repository



-> pulling these changes into repo2

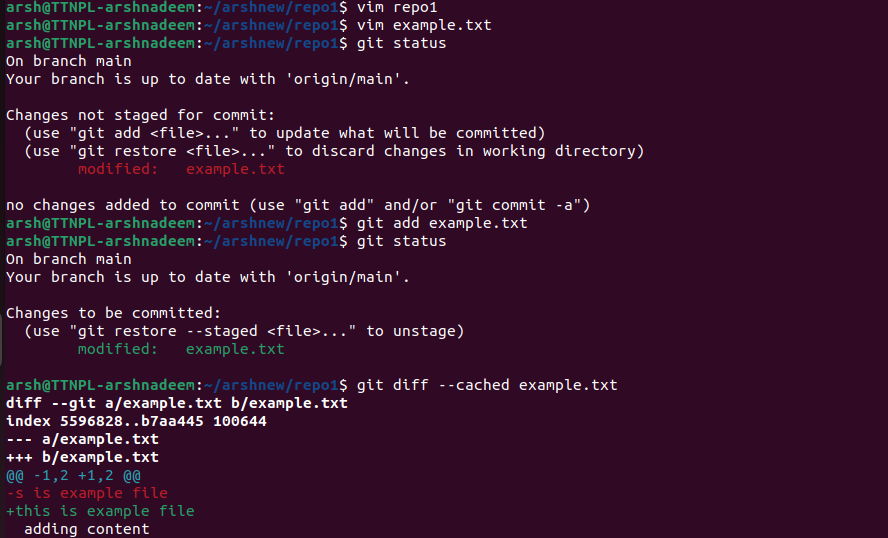


11. Check differences between a file and its staged version

-> changing content using vim

-> staging

-> finding difference using diff --cached filename



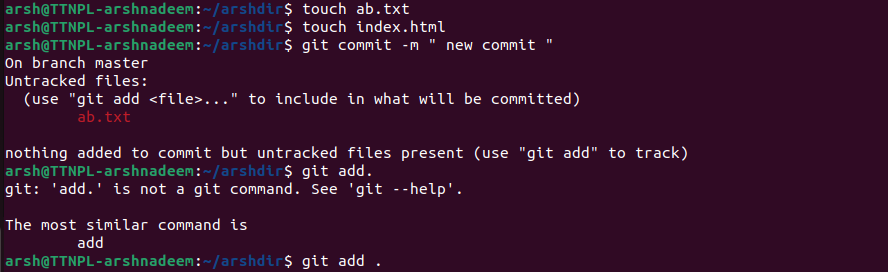
12. Ignore a few files to be checked in

-> creating .gitignore and adding in it file to be ignored, e.g. index.html

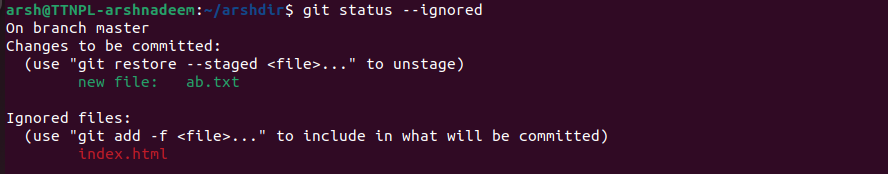


-> creating some file and committing , we can se that index.html is ignored

-> now staging



-> we can verify which file are ignored using git status --ignored command



13. Create a new branch

-> this can be done using git checkout

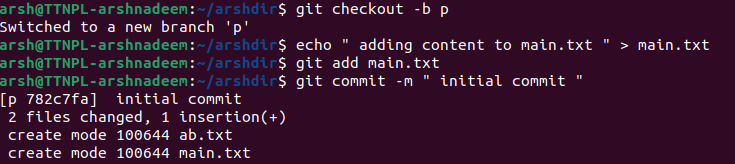


14. Diverge them with commits

-> making changes to this file

-> staging the file

-> committing changes to diverge



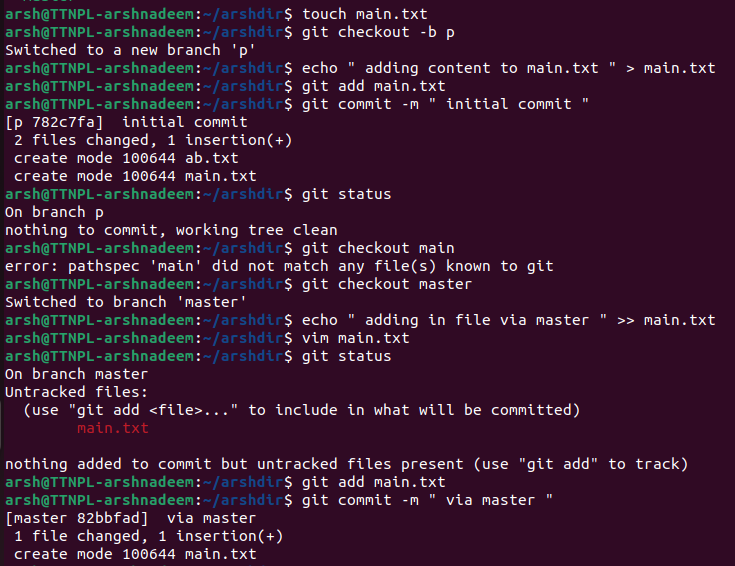
15. Edit the same file at the same line on both branches and commit

-> creating a branch P

-> adding some changes to it and committing it

-> changing the branch to master and then editing the file.

-> committing the changes



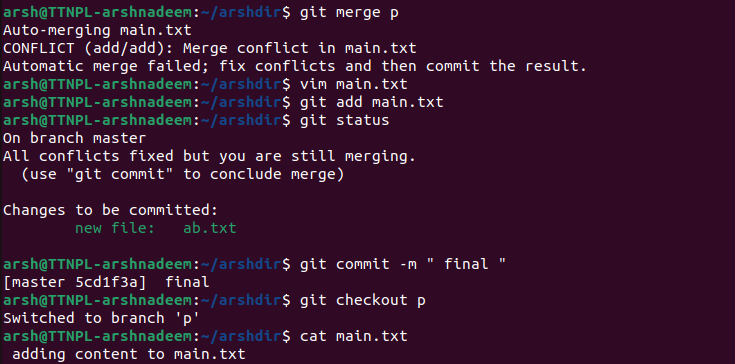
16. Try merging and resolve merge conflicts

-> merging using git merge

-> conflict arises

-> resolving the conflict by editing the file and only keeping the content we want

-> staging and committing



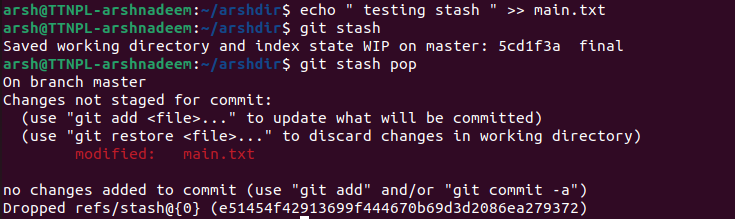
17. Stash the changes and pop them

-> can be done using git stash command, making sure that changes are not committed

-> i have already committed the changes so modifying main.txt a little

-> git stash saved my changes into stash stack, indicating stash was successfully

-> not pop the changes using the git stash pop command.

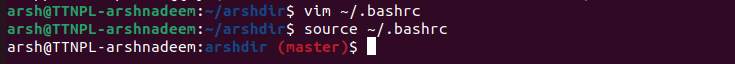


18. Add the following code to your .bashrc file : color\_prompt="yes" parse\_git\_branch() { git branch 2> /dev/null | sed -e '/^[^\*]/d' -e 's/\* \(.\*\)/(\1)/' } if [ "$color\_prompt" = yes ]; then PS1='\u@\h\[\033[00m\]:\[\033[01;34m\]\W\[\033[01;31m\] $(parse\_git\_branch)\[\033[00m\]\$ ' else PS1='\u@\h:\W $(parse\_git\_branch)\$ ' fi unset color\_prompt force\_color\_prompt

-> on adding this using vim

-> i used source to apply changes made in shell

-> branch names appear, i think this function runs git branch command



-> conflict resolved

-> first graph shows the flow of P branch

-> second branch shows the flow of main & P

