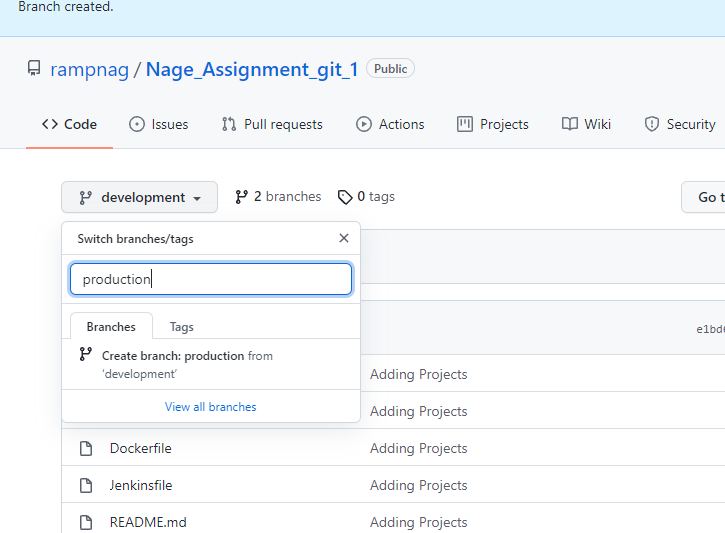
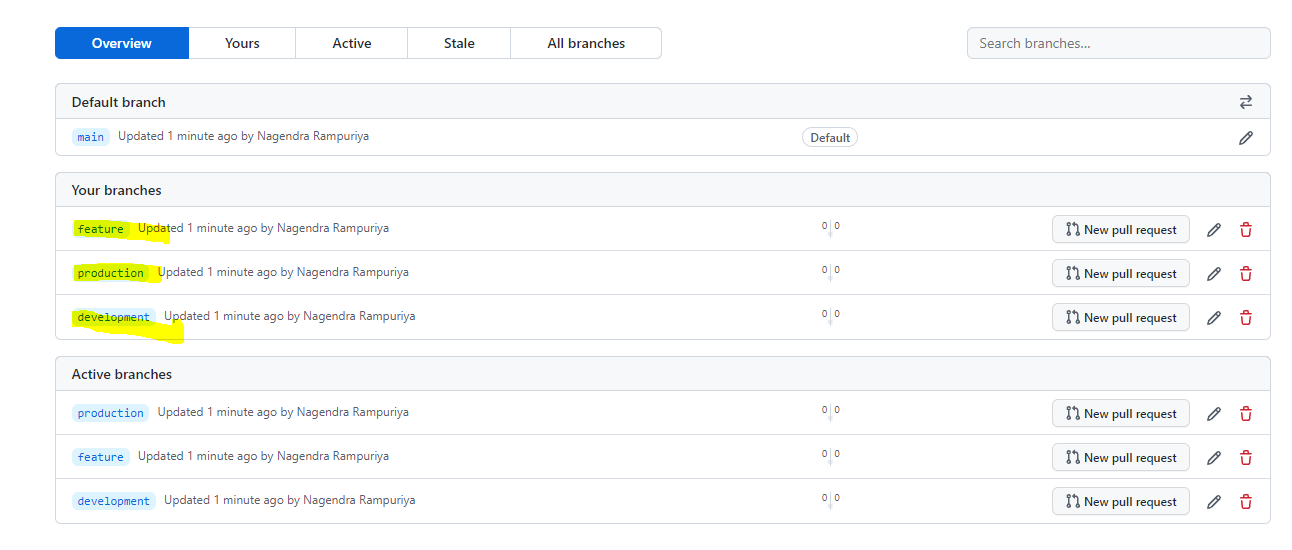
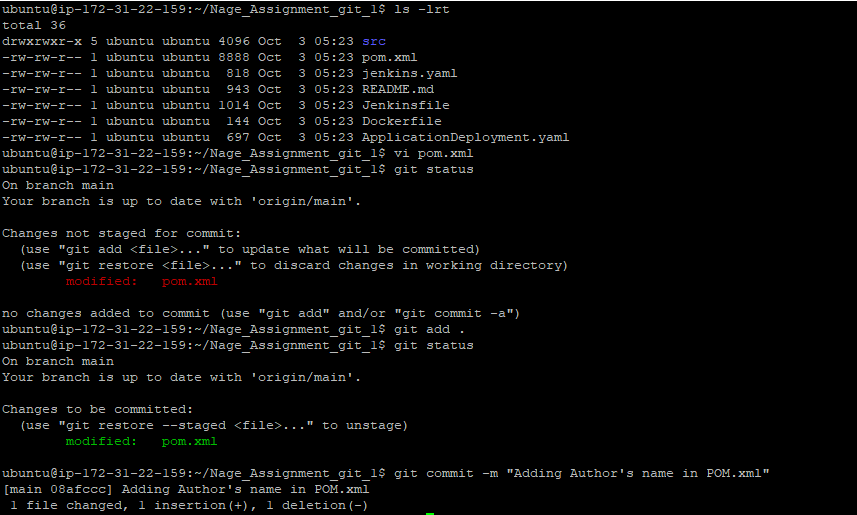
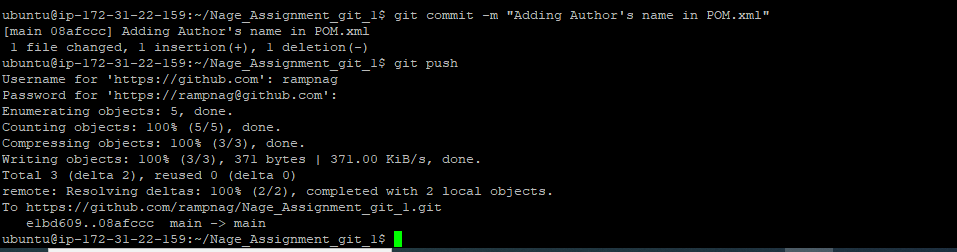
1 - Create a branch(development/production/feature)



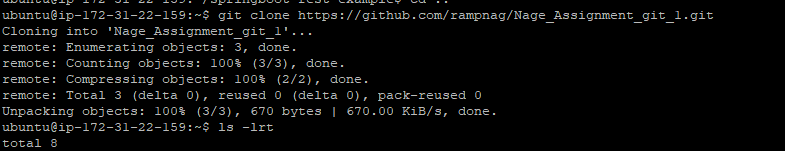


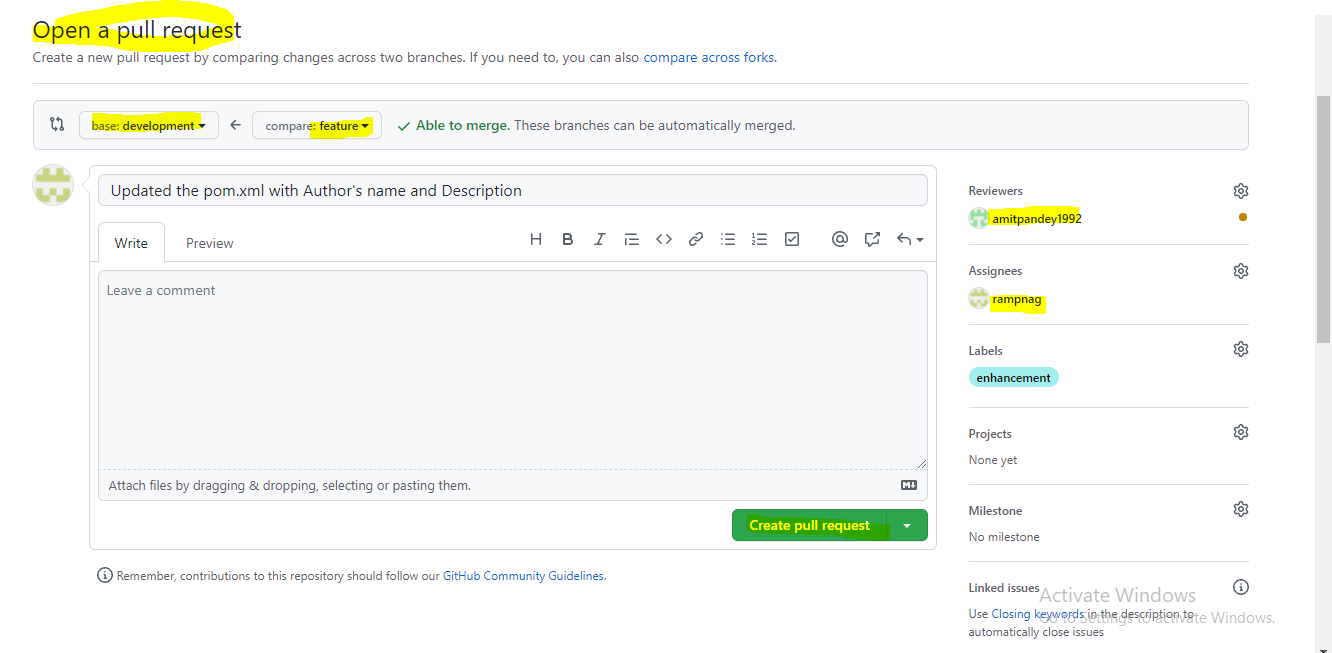
2 - Edit files or create new files followed by commit

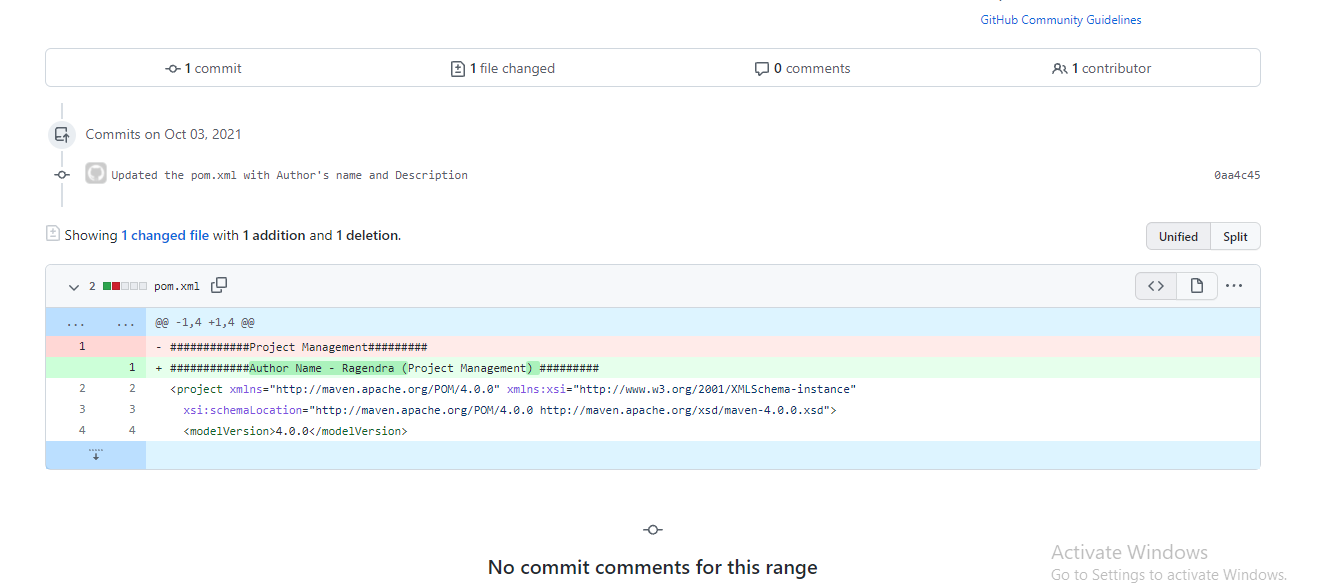


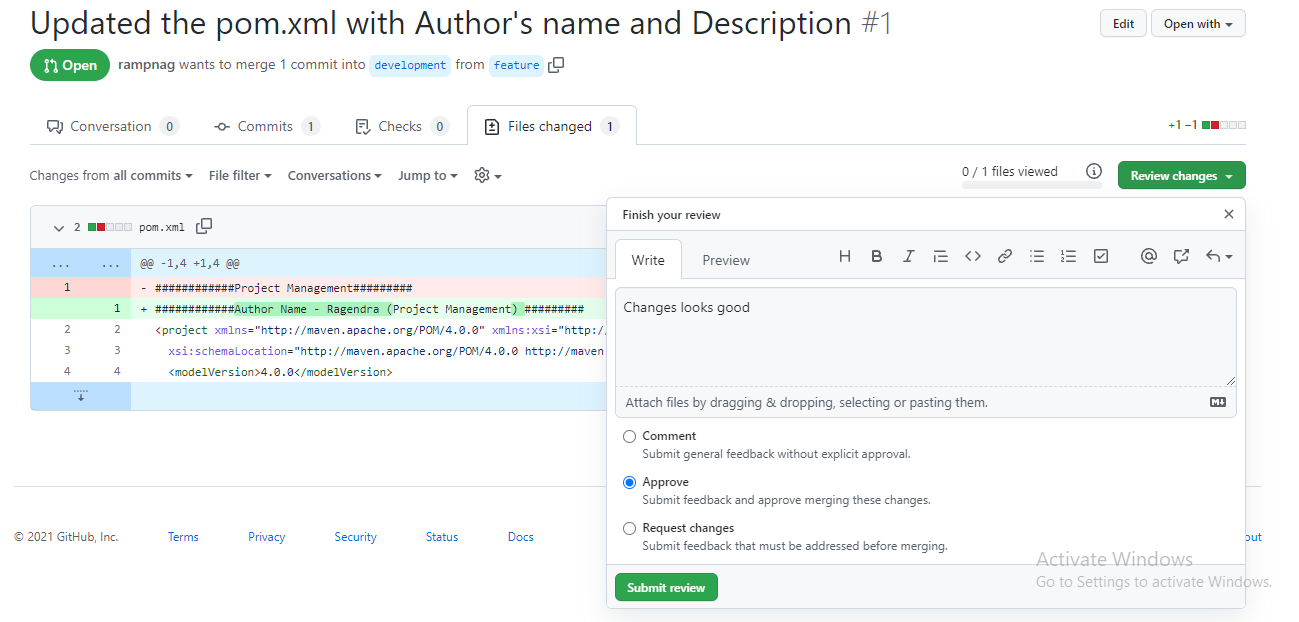


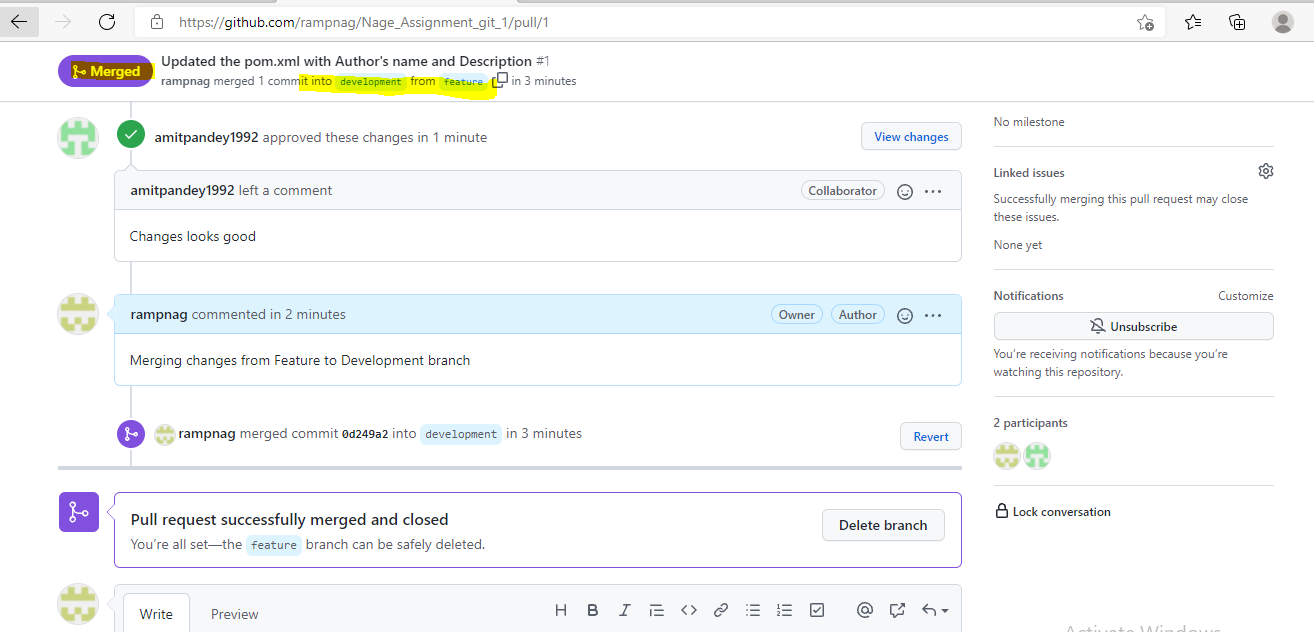
3 - Clone the repo and Create pull-request



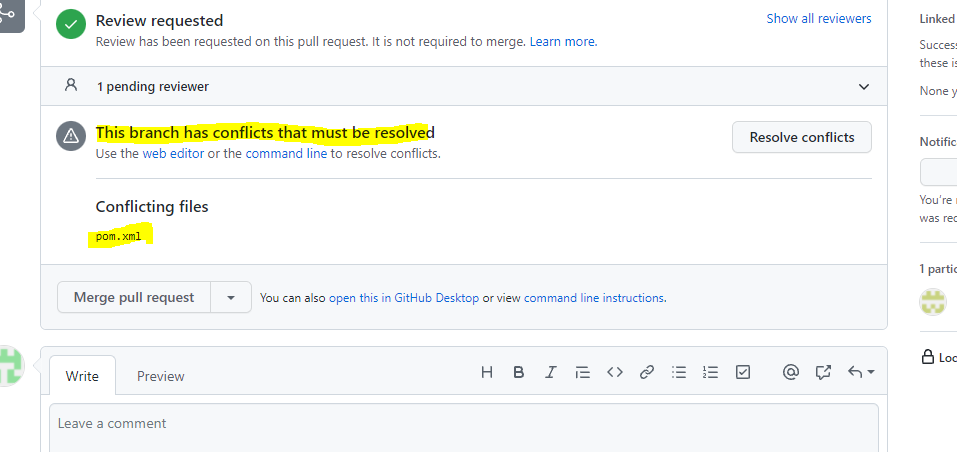


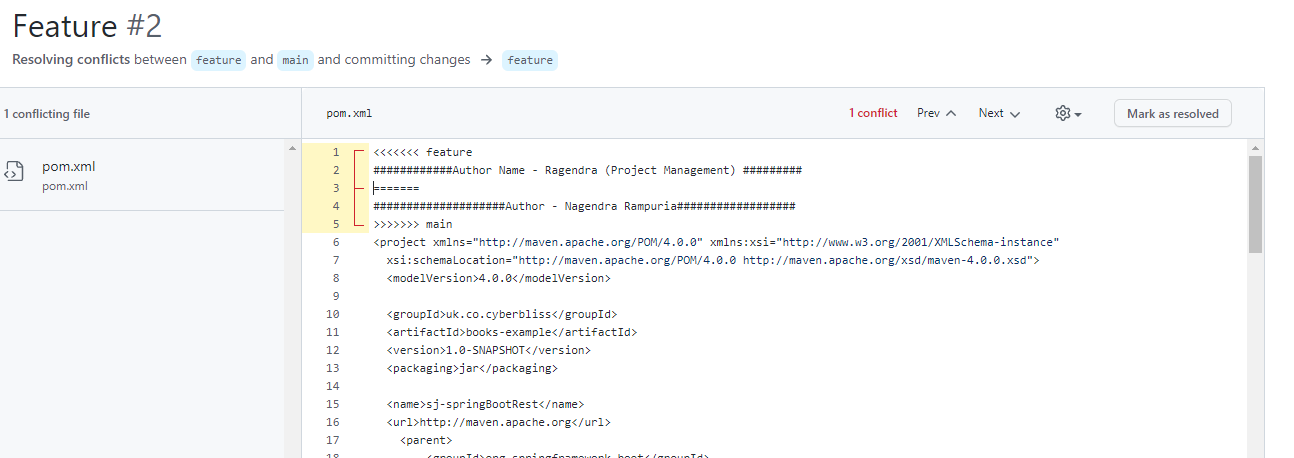


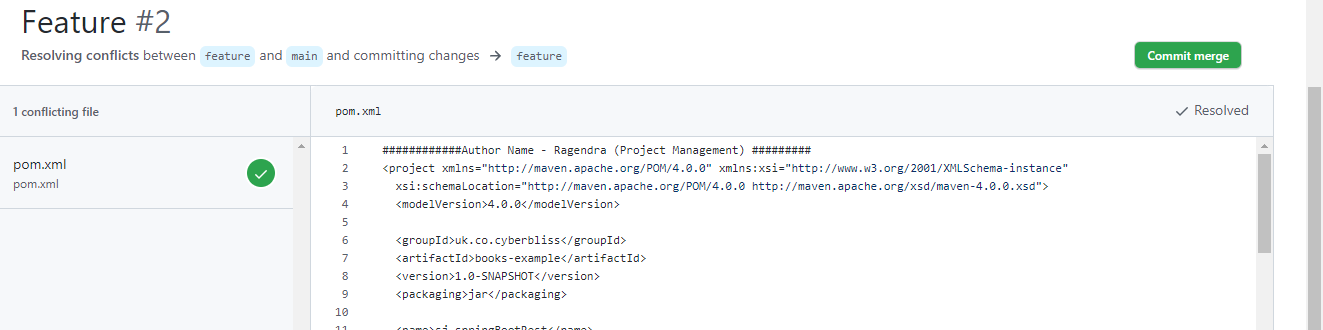


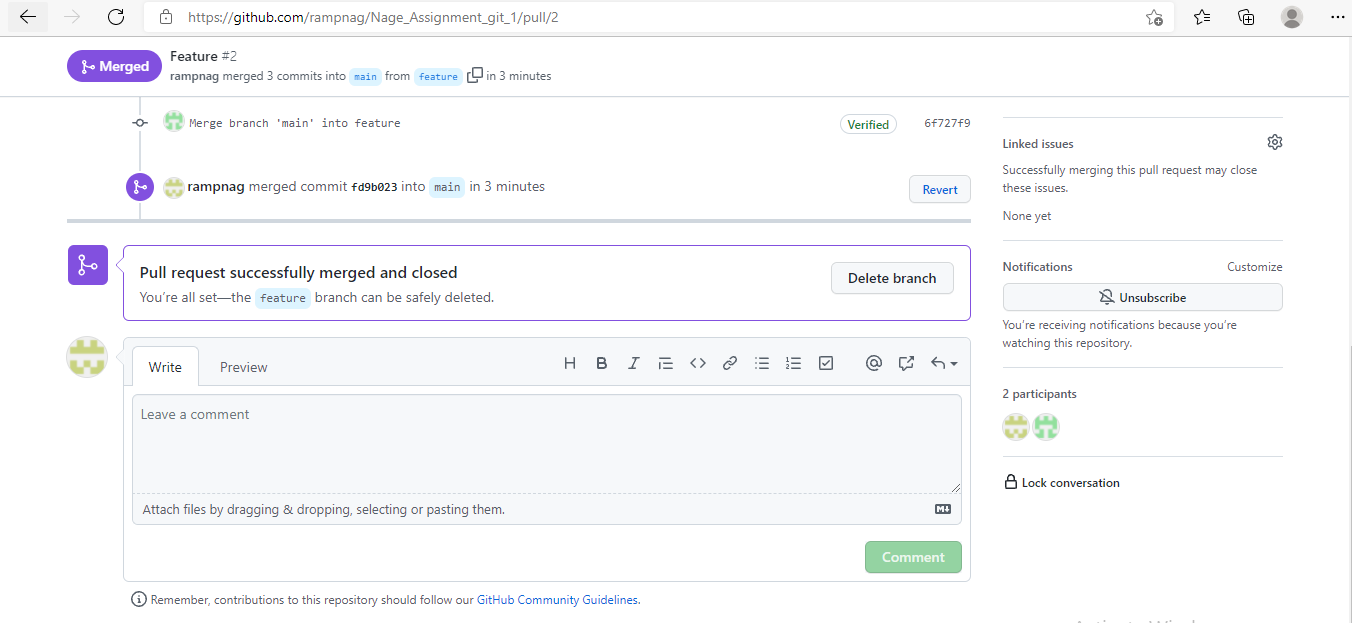


4 - While collaborating your work, showcase how conflicts are resolved

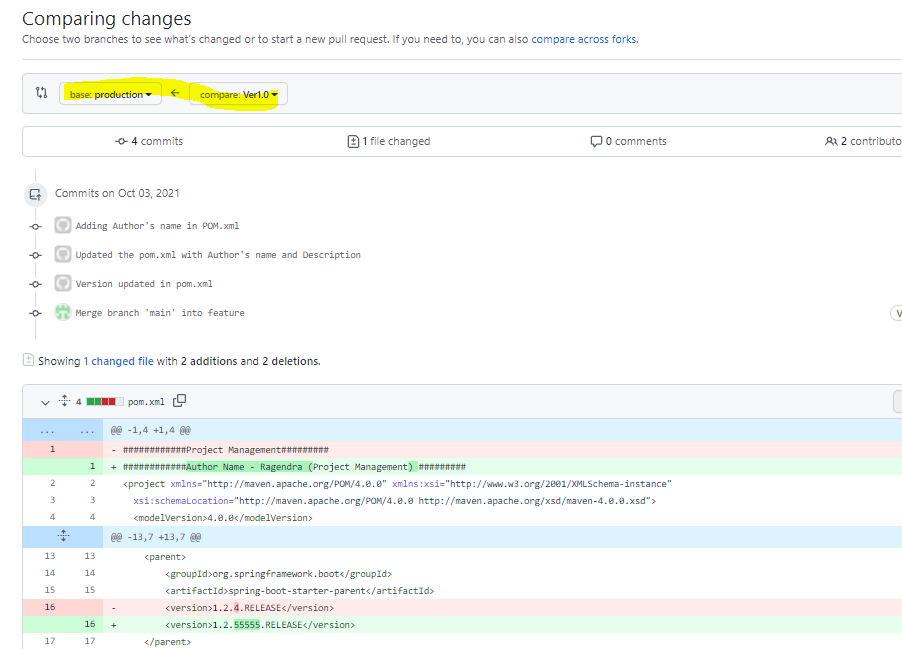




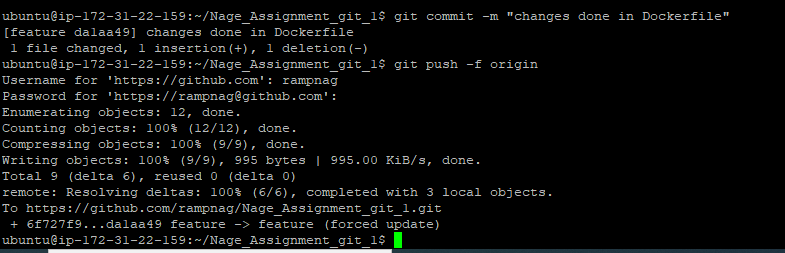


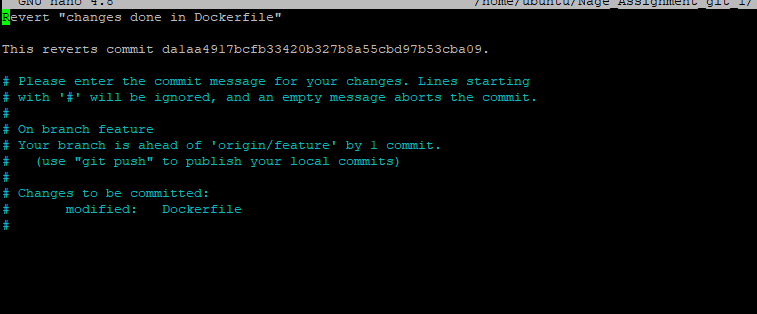


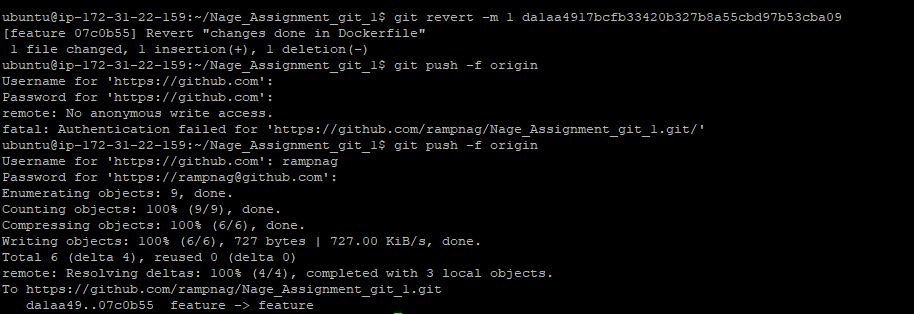
5 - Create tag such as open issue, or feature-added

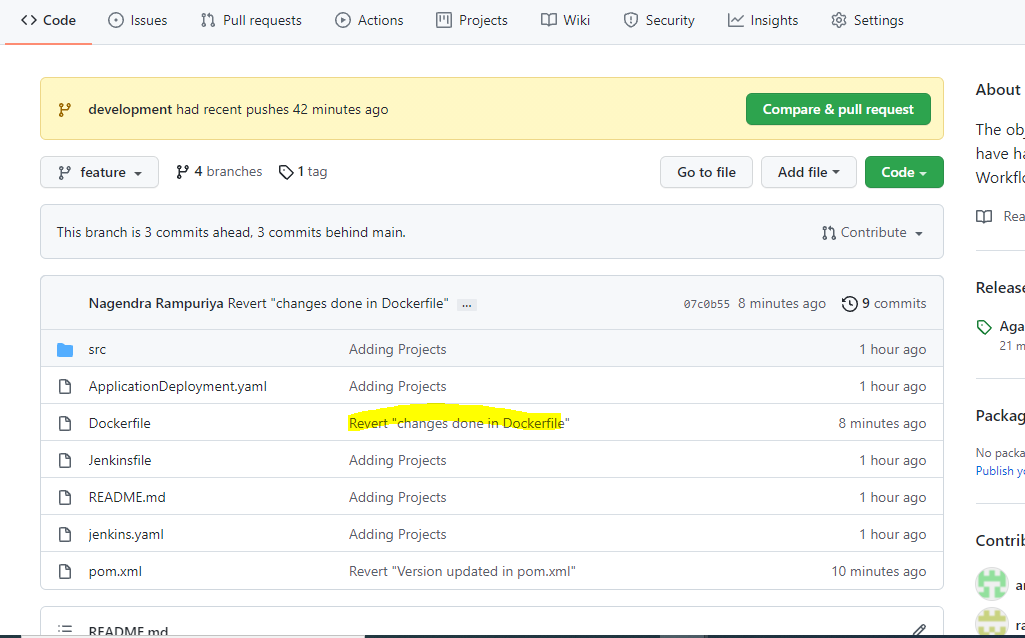


6 - Do a force push/commit and then later reset the changes







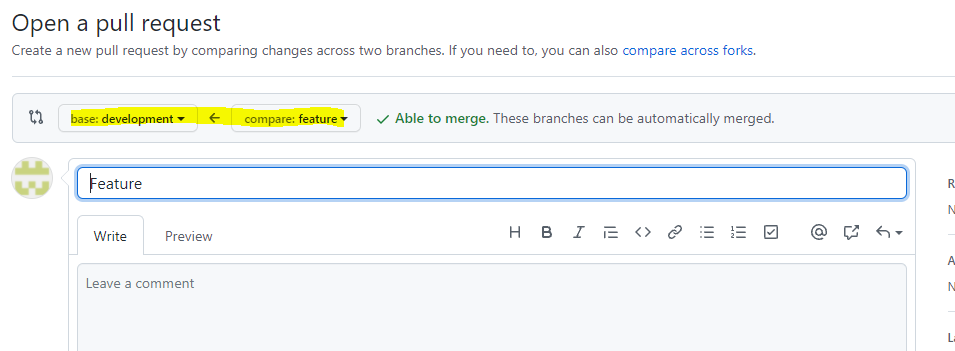




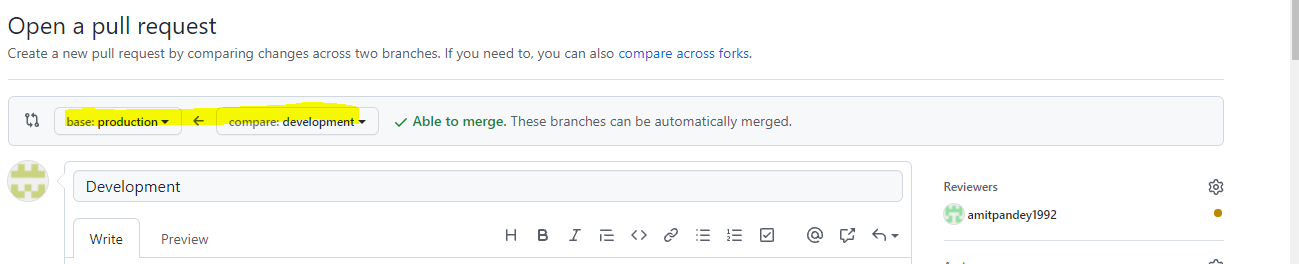
7 - Stage “development branch to production branch”

The thought process here is that you spend most of your time in development. When in development, you create a feature branch (off of development), complete the feature, and then merge back into development. This can then be added to the final production version by merging into production.

Feature to Development –

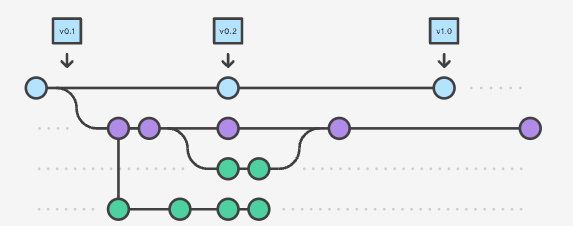


Dev - > Production



8 - Showcase how features are released in versions (merging production to master branch)

Each new feature should reside in its own branch, which can be pushed to the central repository for backup/collaboration. But, instead of branching off of production, feature branches use develop as their parent branch (Development). When a feature is complete, it gets merged back into develop. Features should never interact directly with production.



Note that feature branches combined with the develop branch is, for all intents and purposes, the Feature Branch Workflow. Feature branches are generally created off to the latest develop branch.

9 - Also, state importance of Readme and gitignore files and their usage while working in a distributed environment.

README –

Readme’s provide an introduction to the files contained in the repository and they prevent a person viewing or using your repository from needing to read your mind.

Typical content in an all-purpose, good readme.md would be: the purpose of the repo, who contributed code, instructions on how to use the content within the repo, what technologies were used in the creation of the project or homework and potentially much more. There are also wide range of inputs such as text, links, pictures, screen shots, that could be used to tell a story and document your repo. Specifically for students submitting projects or homework that will be evaluated by a TA, instructor or a potential employer, the readme file provides instructions and direction. Typical good readme’s contain, homework / project name, links to specific files, listing of which files are important, a sentence or two about what worked as well as what didn’t or who worked on the project and can be used to show off your understanding of the required tasks and even the full analysis of the work. A really great readme can also then be easily polished up after grading so that potential employers could easily evaluate your work.

Gitignore –

The purpose of gitignore files is to ensure that certain files not tracked by git remain untracked. The purpose of gitignore files is to ensure that certain files not tracked by git remain untracked. Say you run git add . from the root directory of your repo, and you have nothing ignored.

**10 - Bonus point: If Pull Requests are linked with e-mail to the manager who finally approves the changes.**

Email features is working when I am adding Amit as a contributor, I got confirmation from Amit as he is receiving alert emails against PR.