**GIT**

**git init**

**git remote add origin {repositoryURL}**

**git pull origin {branchName}**

**git add {fileName}**

**git commit -m {commitMessage}**

**git push [origin {branchName}]**

**git status**

**git pull - -rebase origin {branchName}**

**git mergetool resolve conflicts first**

**git pull – -continue**

**git push origin {branchName}**

**git branch {branchName}**

**git checkout {switch to a branch}**

**Master (Production branch)**

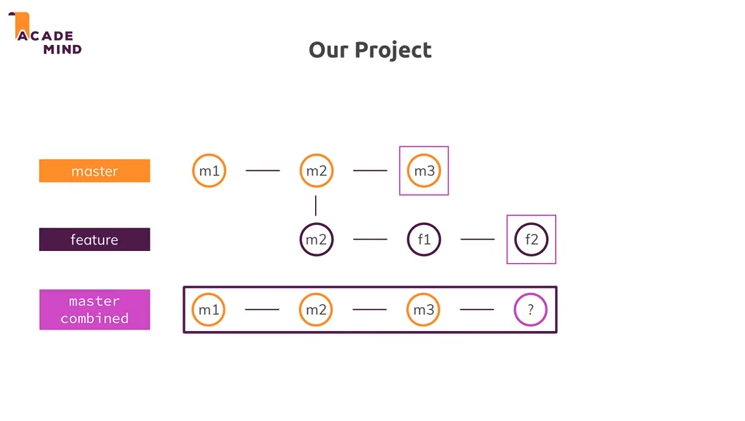
**Feature branch**

**Develop branch**

**Release branch**

**Hotfix branch**

**git log**



**We are inside master branch >**

**git merge feature**

**git merge - -squash feature**

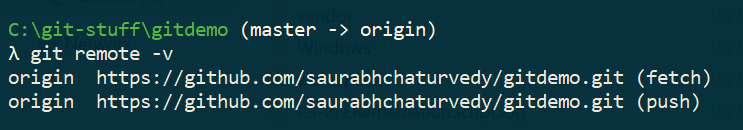
**squash allows us to summarize all changes we had in the feature branch in the last commit and then merges the last commit in feature to latest commit in master branch .**

**we are in feature >**

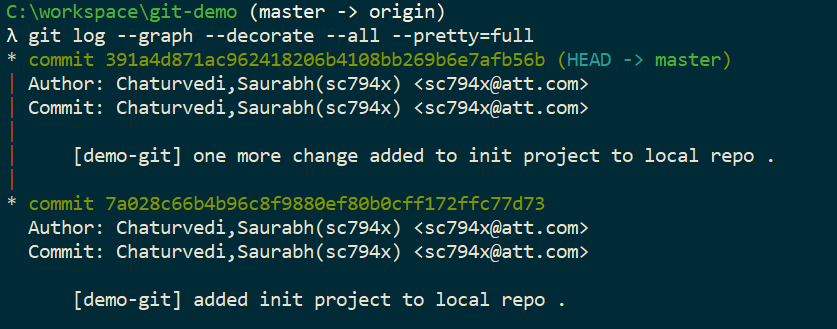
**git branch rebase**

**git pull = git fetch + git merge**

**git fetch gets changes from remote to local repository without merging changes to local repository .**



**Git remote -v gives list of origins that we have added till now .**



**git log --oneline**

**Resolving a merge conflict**

**git pull origin master (Target Branch)**

**git add {files}**

**git commit -m ' abckkkdfkfkfkl‘**

**git push conflictbranch (Source Branch)**

**to set our head to a particular commit DO**

**git checkout {abcd578785fdfd0008990} HASH**

**git branch {branchName} creates a new branch from the branch where we are currently in**

**How to merge code from Branch xyz to master**

**CD into master branch (target)**

**From master**

**Run git merge {branch xyz}**

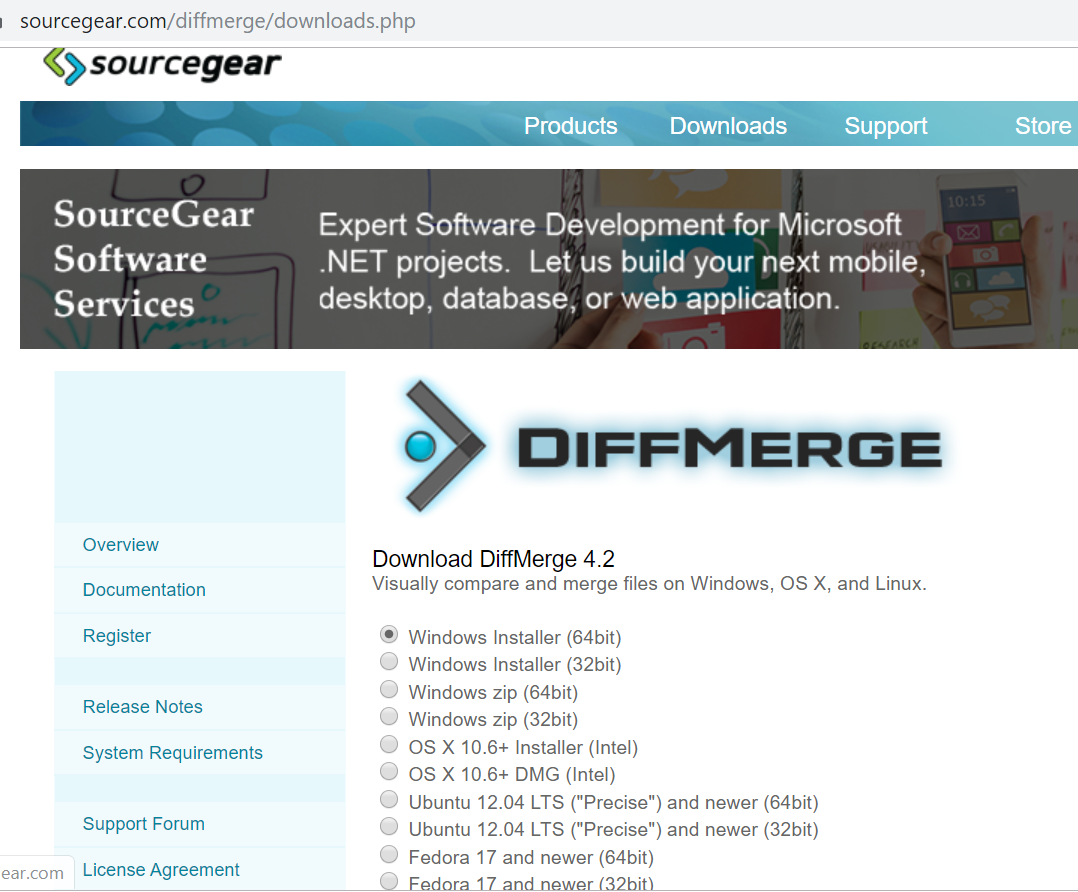
**In real world we do not merge branches locally but through a PULL request .**

**How to delete a branch :**

**git branch -d {branchName}**

**git branch -D {branchName} [ Force Delete ]**

**we can delete a branch while being at the same branch**



**git config –list**

**<<<<<<**

**>>>>>>> called conflict markers**

**git checkout {fileName}**

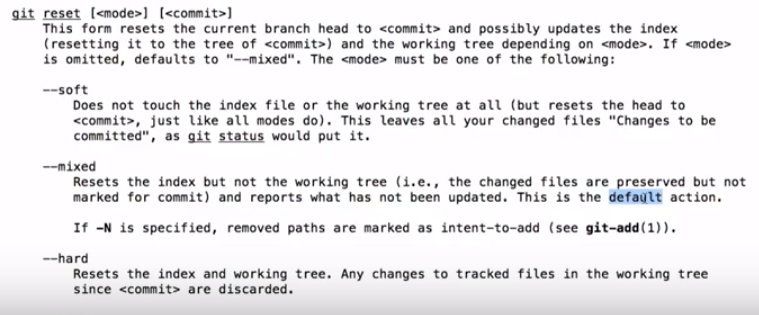
**it will discard changes to all files in working area**

**git checkout . discard changes from all files in the working area**

**git reset {fileName} unstage the file from staging area**

**git reset . unstage all files from the staging area**

**git reset HEAD~1**



**Type of GIT Resets**

**MIXED**

**HARD**

**SOFT**

**Git reset –mixed HEAD~1 changes are kept in working area**

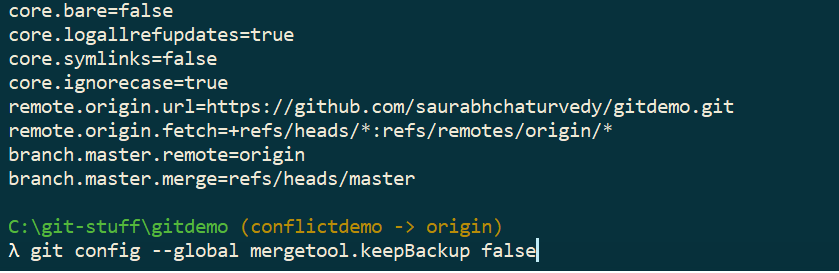
**Git reset –soft HEAD~1 changes are kept in staging area**

**Git reset –hard HEAD~1 changes are neither kept in staging nor working area .**

**git revert help revert changes after when being pushed and also make a fresh commit .**

Git Resolve Conflicts In Pull Request

**Git resolve conflicts in pull request .**



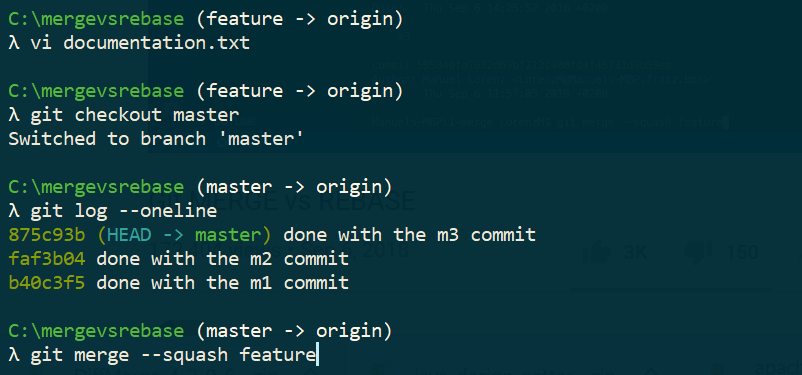
**GIT CHERRY PICK**

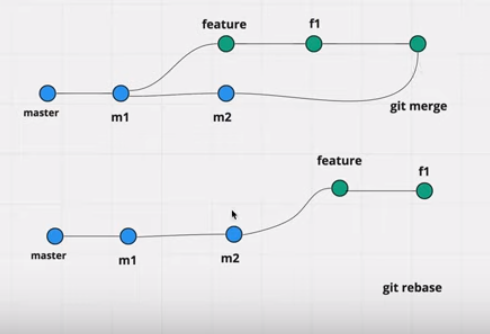
**Git commit {commit hexa value}**

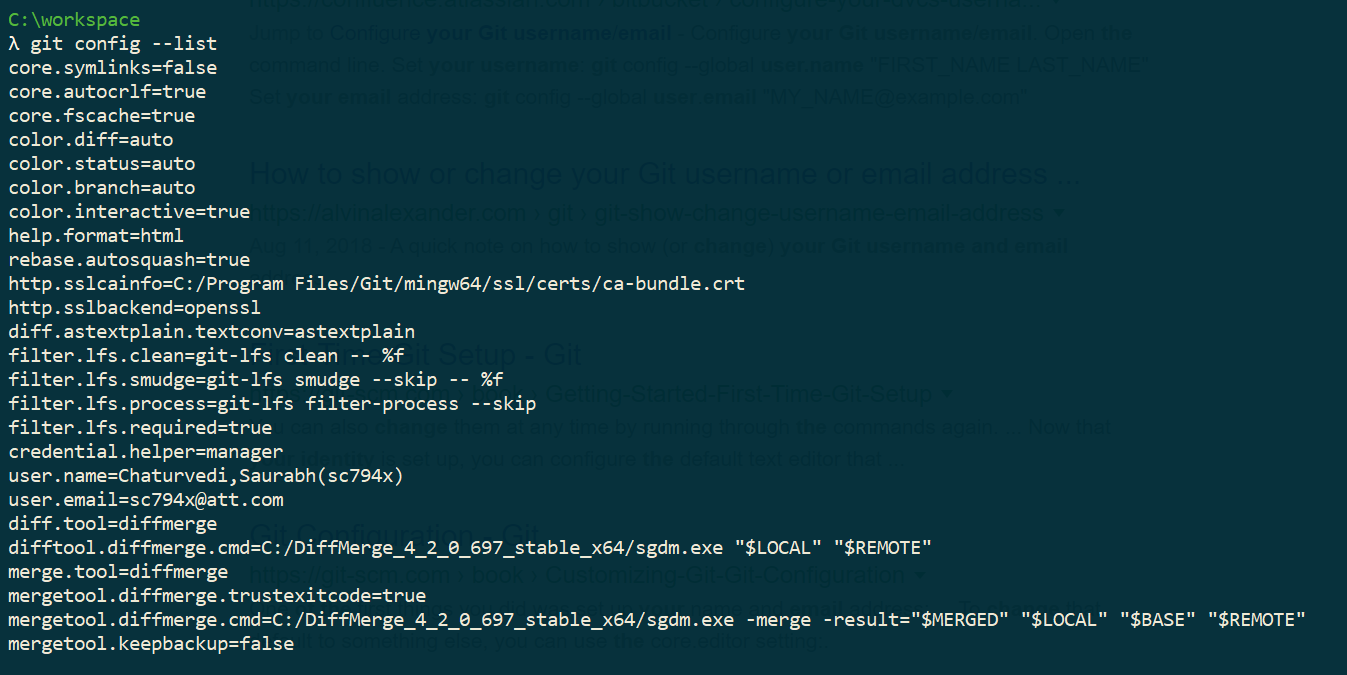
**GIT MERGE VS GIT REBASE**

**In GIT MERGE Git creates a new commit and integrates the changes in feature branch to master branch .**

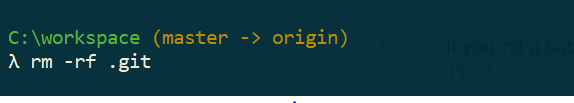
**General use case of GIT REBASE is keeping Git history linear .**







**UNGIT a GIT Repository**



<https://javagyanmantra.wixsite.com/website/single-post/2018/04/24/Java-EnumMap>

**git ls-files --deleted | xargs git add**