**GIT Rebase**

Rebasing is the process of combining or moving a sequence of commits on top of a new base commit. Git rebase is the linear process of merging. It is also called fast forward merge.

• Create a folder

• Open Git Bash

• Convert the folder into working directory

**git init**

• Create file

**touch file1**

• Check file status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Check file status

**git status**

• Create file

**touch file2**

• Move all the file to Staged Area

**git add .**

• Check file status

**git status**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Create Branch

**git branch <Branch\_name>**

• After creation of feature branch still we need to move in the master branch

• Change Branch

**git checkout <Branch\_Name>**

• Create file

**touch file3**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit3”**

• Create file

**touch file4**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit4”**

• Check all Commits

**git log --oneline**

• We are able to see master branch & feature branch commits.

• See all Branch

**git branch**

• Check the files

**ls**

• We are able to see master branch & feature branch files because system copy the commit history.

• Move to Master Branch

**git checkout master**

• Check the files

**ls**

• Check all Commits

**git log --oneline**

• Create file

**touch file5**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit5”**

• Check all Commits

**git log --oneline**

• Check the files

**ls**

• To do rebase we need to be in the child branch

• Move to Child Branch

**git checkout <Branch\_name>**

• Rebase Commit

**git rebase master**

• Move to Master Branch

**git checkout master**

• Merge Branch

**git merge <Branch\_Name>**

• Check all Commits

**git log --oneline**

**• Difference Between Merge & Rebase**

Merging is a safe option that preserves the entire history of your repository, while rebasing creates a linear history by moving your feature branch onto the tip of main

Re arrange Commit History

• Create a folder

• Convert the folder into working directory

**git init**

• Create file

**touch file1**

• Check file status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Check file status

**git status**

• Create file

**touch file2**

• Move all the file to Staged Area

**git add .**

• Check file status

**git status**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Create file

**touch file3**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit3”**

• Check file status

**git status**

• Create file

**touch file4**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit4”**

• Check all Commits

**git log --oneline**

• Re arrange the commit history

• We need to rearrange the last two commits

**git rebase -i HEAD~2**

• Press Enter

• Press I

• Copy the Commit & paste where we want to schedule

• Press ESC

• Enter the command

**:wq!**

• Check all Commits

**git log –oneline**

Merge Commit

A merge commit is just like another commit, the state of your repository at a given point in time plus the history it evolved from.

For Example, while working developer has created one commit & after few days he thing that there is no use of the commit we can merge the commit.

• Note: We can’t merge the first commit.

• Note: It always merge with the above commit.

• Create a folder

• Convert the folder into working directory

**git init**

• Create file

**touch file1**

• Check file status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Check file status

**git status**

• Create file

**touch file2**

• Move all the file to Staged Area

**git add .**

• Check file status

**git status**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Create file

**touch file3**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit3”**

• Check file status

**git status**

• Create file

**touch file4**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit4”**

• Check all Commits

**git log --oneline**

• Squash Commit

**git rebase -i HEAD~3**

• Press Enter

• Press I

**Remove pick & change it to squash**

• Press ESC

• Exit the file

**:wq!**

• Press ESC

• Exit the file

**:wq!**

• Check All Commit

**git log --oneline**

• Check file in commit

**git show --name-only –onelin` e <Commit-Hash>**

Certain Commit

• The specific existing commits to include in another branch

• Create a folder

• Open Git Bash

• Convert the folder into working directory

**git init**

• Create file

**touch file1**

• Check file status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Check file status

**git status**

• Create file

**touch file2**

• Move all the file to Staged Area

**git add .**

• Check file status

**git status**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Create Branch

**git branch <Branch\_name>**

• Change Branch

**git checkout <Branch\_Name>**

• Press Enter

• Create file

**touch file3**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit3”**

• Create file

**touch file4**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit4”**

• Check all Commits

**git log --oneline**

• We are able to see master branch & feature branch commits.

• See all Branch

**git branch**

• Check the files

**ls**

• We are able to see master branch & feature branch files because system copy the commit history.

• Move to Master Branch

**git checkout master**

• Check the files

**ls**

• Check all Commits

**git log --oneline**

• Create file

**touch file5**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit5”**

• Check all Commits

**git log --oneline**

• Check the files

**ls**

• Move to Child Branch

**git checkout <Branch\_Name>**

• Check all Commits

**git log --oneline**

• Note down the Commit Hash for those commit we want to merge with master branch.

• Move to Master Branch

**git checkout master**

• Merge

**git cherry-pick <Commit-Hash>**

• Check all Commits

**git log –oneline**

**AMEND**

• It is used to modify the most recent commit.

• Create a folder

• Open Git Bash

• Convert the folder into working directory

**git init**

• Create a file in the folder

• Add some lines in the file

• Check the files

**ls**

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Do Some changes in the file

• Check status

**git status**

• If we modify any file system will automatically move the file from local repository to working directory.

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Do Some changes in the file

• Whenever developer do any changes in the file it is not required to create a new commit every time.

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Commit

**git commit --amend -m "commit2“**

• Check status

**git status**

• Check all Commits

**git log –oneline**

**Merge Conflicts**

• When working in Git, users can combine commits from two different branches through an action known as merging. Files are automatically merged unless there are conflicting sets of changes (i.e. the commits update the same line of code differently).

• A merge conflict is an event that occurs when Git is unable to automatically resolve differences in code between two commits.

• Create a folder

• Convert the folder into working directory

**git init**

• Create a file in the folder

• Add some text in the file

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Create new branch

**git branch <Branch\_Name>**

• Move to another branch

**git checkout <branch\_Name>**

• Check the files

**ls**

• Read the file

**cat <File\_Name\_With\_Extension>**

• Edit the file

**vi <File\_Name\_With\_Extension>**

• Press Enter

• Convert into Insert Mode

**Press i**

• Enter More Text in the file

• Press ESC

• Save & exit the file

**:wq!**

• Read the file

**cat <File\_Name\_With\_Extension>**

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Read the file

**cat <File\_Name\_With\_Extension>**

• Move to Master branch

**git checkout master**

• Read the file

**cat <File\_Name\_With\_Extension>**

• We are not able to see the lines which was added in the feature branch.

• Edit the file

**vi <File\_Name\_With\_Extension>**

• Press Enter

• Convert into Insert Mode

**Press i**

• Enter More Text in the file

• Press ESC

• Save & exit the file

**:wq!**

• Read the file

**cat <File\_Name\_With\_Extension>**

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit3”**

• Merge Branch

**git merge <Branch\_Name>**

• We are getting error while merging the branch because we have used the same file in both the branch

• Read the file

cat <File\_Name\_With\_Extension>

• Edit the file

vi <File\_Name\_With\_Extension>

• Convert into Insert Mode

**Press i**

• Remove the Extra lines

• Press ESC

• Save & exit the file

**:wq!**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit4”**

• Check status

**git status**

**Reset**

• Reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

• Create a folder

• Convert the folder into working directory

**git init**

• Create multiple file in the folder

• Add Some Text into the files

• Check the files

**ls**

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit1”**

• Create more files in the folder as per client requirement

• Check the files

**ls**

• Check status

**git status**

• Move all the file to Staged Area

**git add .**

• Move All files from stage area to local repository

**git commit -m “commit2”**

• Check All Commit

**git log --oneline**

• Now client requirement is whatever changes done in the last commit needs to be remove.

**Reset**

• Note down the previous commit hash

• Git Reset

**git reset --hard <Commit-Hash>**

• Check All Commit

**git log --oneline**

• Check the files

**ls**

• Now whatever changes done in the file or created new files all will get deleted after reset.