**REBASE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Rebasing is the process of moving or combining a sequence of commits to a new base commit. With the rebase command, we can take all the changes that were committed on one branch and replay them on another one. The main difference of rebasing and merging is that Rebasing replays changes from one line of work onto another in the order they were introduced, whereas merging takes the endpoints and merges them together. Rebasing is most useful and easily visualized in the context of a feature branching workflow.

Use the git log command to track the changes (commit history). Checkout to the desired branch you want to rebase.

Rebase command is as follows,

git branch

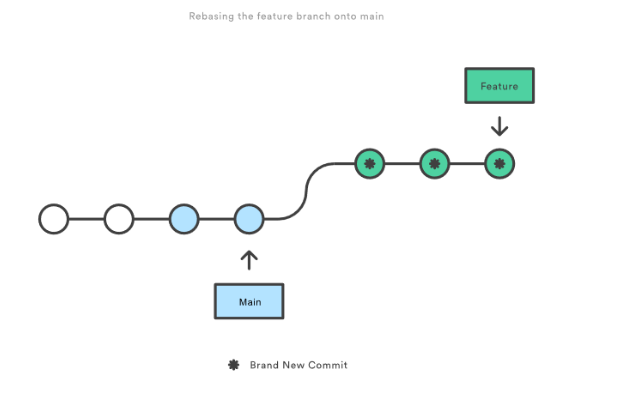
git checkout <branchname>

git log --oneline

git rebase <branchname>

**What is the use of rebase**

Instead of using merge commit ,rebasing rewrites the project history by creating brand new commits for each commit in original branch.



**What does a git rebase do?**

A Git rebase changes the base of developers branch from one commit to another so it looks like they have created a branch from a different commit.

**Tags**

Tag means creating a specific point in the history of data. By default, git tag will create a tag on the commit that HEAD is referencing. The main purposes of tags are to make release point on your code and to create historic restore points. We can have any number of tags on a branch or different branches.

Types of git tags,

1)Annotated tags -Annotated tags are tags that store extra information like name, email, date, and more. They are stored as a bundle of objects in the git database.

2)Light-weighted tags -The motive of both tags is the same as marking a point in the repository but in light weighted tags it does not store unnecessary information to keep it light-weight.

1.Checkout to the branch you want to create the tag.

git checkout master

2.Create Lightweight Tags

It will contain a tagger name and some message.

**git tag -a v6.7 -m "my version 6.7”**

3)Tagging Old Commits

**git tag v0.6 <<commit id>>**

4)To verify the tag

**git tag –v v1.0**

**git push –tag** // git push –tags” will push all tags at once.

5) To show

**git show Release\_1\_0** //show available tags in repository

6)To delete tag

**git tag -d Release\_1\_0** //git tag -d {tag name}

7) View the state of a repo at a tag by using

**git checkout Release\_1\_0**

8) ReTagging/Replacing Old Tags

**– git tag -a -f v1.4 15**

**How to roll back deleted untracked files?**

We cannot Restore the files once that are deleted .

**What Is Git Rebase: All About Configuration**

You can use git config to set some rebase properties if you want. Here are some configuration options you can take advantage of. Note that these options will alter the Git rebase output’s feel and look.

* rebase.stat- This is a boolean set to “false” by default. This option toggles the display of visual diffstat content showing what has changed since the last rebase.
* rebase.autoSquash- This boolean value toggles the --autosquash behavior.
* rebase.instructionFormat- This is a Git log format string used for formatting an interactive rebase display.
* rebase.missingCommitsCheck- This option can be set to multiple values, changing rebase behavior around missing commits. The values are:
  + Warn. This prints warning output in interactive mode and warns you of removed contents.
  + Error. This stops the rebase and prints any removed commit warning messages.
  + Ignore. This value is set by default and ignores missing commit warnings.

Now, about those rebase commands.