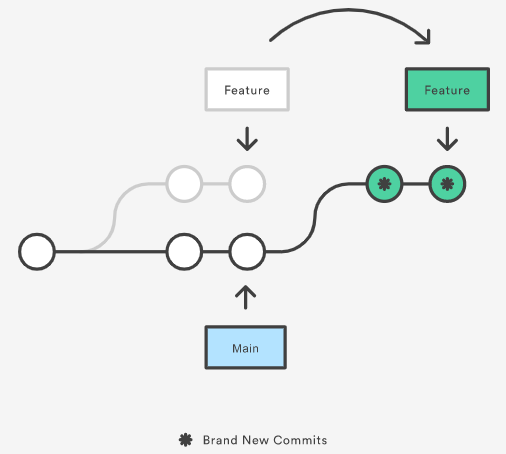
**Git rebase and squash**

**Rebase:** is the process of moving or combining a sequence of commits to a new base commit. Rebasing is most useful and easily visualized in the context of a feature branching workflow. Rebasing is changing the base of your branch from one commit to another making it appear as if you'd created your branch from a different commit. Internally, Git accomplishes this by creating new commits and applying them to the specified base.



**Squash:** means to combine multiple commits into one. It is usually used when merging branches, by using Git’s interactive rebase. Squashing commits is the preferred way to merge a feature branch back into a long-running branch like "master" or "main".

Data from

<https://www.git-tower.com/learn/git/faq/git-squash>

**Git merge vs Git rebase**

Git rebase and merge both integrate changes from one branch into another. Where they differ is how it's done. Git rebase moves a feature branch into a master. Git merge adds a new commit, preserving the history. Rebase merge all commits into one and merge maintains all the commits separately.

The team decides whether if they go for rebase or merge, there is no right or wrong answer

Benefits

Here are the top three benefits for Git rebase and for Git merge.

**Git Rebase**

* Streamlines a potentially complex history.
* Avoids merge commit “noise” in busy repos with busy branches.
* Cleans intermediate commits by making them a single commit, which can be helpful for DevOps teams.

**Git Merge**

* Simple and familiar.
* Preserves complete history and chronological order.
* Maintains the context of the branch.

Data from

<https://www.perforce.com/blog/vcs/git-rebase-vs-git-merge-which-better#:~:text=Git%20rebase%20and%20merge%20both%20integrate%20changes%20from,merge%20adds%20a%20new%20commit%2C%20preserving%20the%20history>.