Git – Squash

Squashing combines multiple commits into a single commit based on your commit history. With the help of squashing you can clean your branch history and can maintain an organized commit timeline. It is used before pulling requests or merging feature branches.

What is Git Squashing?

Assume you are building a project using Git as your version control. You released Version One after that you wanted to add new features for the Version Two release and you even fixed a few bugs found in Version One. You had plenty of commits piled up after your 1st release. Is there a way to merge all the commits after 1st release into a single commit? yes, it is possible using squash. Squash is one of the useful and powerful features available in the git rebase command's interactive mode.

Git Squash-Merge Command Line

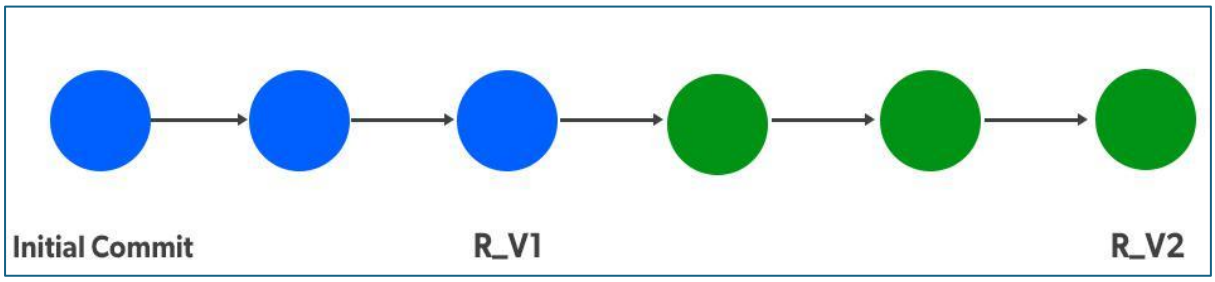
By using the git squash-merge command you can combine multiple commits into a single commit. After using the git squash command your repository will be clean and organized manner.

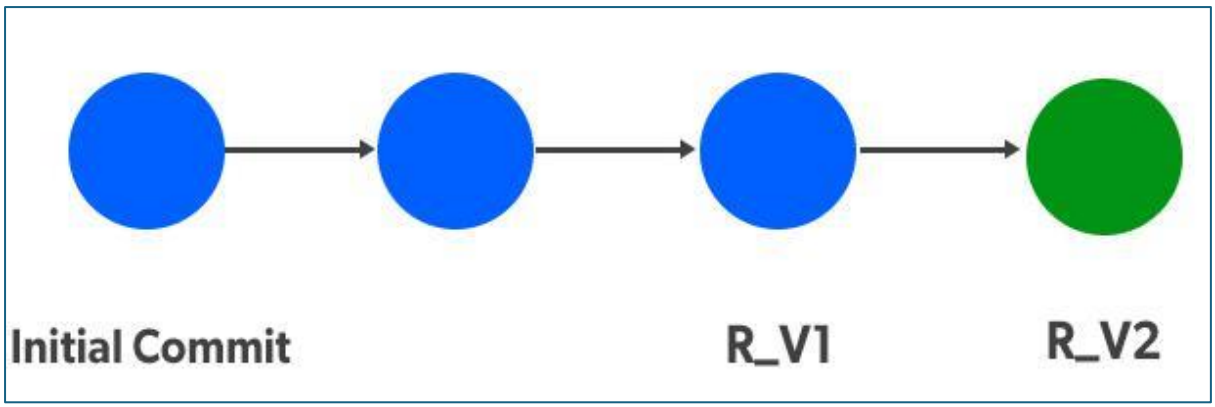
```
git merge --squash feature-branch
```

To merge the changes in the feature branch to your current branch we can use the above command.

When To Squash Commits?

Now we have a repository called GFG_VIDEO, which is an open-source video conferencing tool. GFG_VIDEO has released the 1st version of their tool with basic features such as peer-to-peer video calling and messages with the tag R_V1 (Green colored). After the R_V1 release team GFG_VIDEO started to add new features such as creating groups, group video calls, and fixing minor bugs from R_V1 such as call drops, etc. Now GFG_VIDEO is ready for their new release R_V2.





If you observe, we have three commits from our initial commit to R_V1 (1st Release). After our R_V1 we have 3 commits for our R_V2 (2nd Release), It kind of looks untidy and difficult to follow. Here we can use the Squash concept and merge all the commits after R_V1 till R_V2 to a single commit which makes our repository log more tidy and easy to follow.

How To Squash Commits?

The below image shows we have 3 commits: Initial commit, Commit 2, and Version 1 Release. We've successfully released the 1st version (R_V1) of the GFG_VIDEO tool. After R_V1 new features are added and minor bugs are fixed from the previous release and the tool is ready for its 2nd release R_V2.

```
ubuntu@UbuntuBox:~/GFG_VIDEO$ git log --oneline
19e9d9a (HEAD → master, tag: R_V1) Version 1 Release
5551447 Commit 2
0b9f017 Inital Commit
```

```
ubuntu@UbuntuBox:~/GFG_VIDEO$ git log --oneline
af9ea62 (HEAD → master, tag: R_V2) Version 2 Release
01109f5 New Feautures Added
c195647 Bugs Fixed
19e9d9a (tag: R_V1) Version 1 Release
5551447 Commit 2
0b9f017 Inital Commit
ubuntu@UbuntuBox:~/GFG_VIDEO$
```

The above image of the GFG_VIDEO log is after the 2nd version release. It can be observed after the Version 1 Release (tag: R_V1) there are 3 commits for the Version 2 Release. This kind of looks untidy, to make it simpler to read we can do a squash operation.

Let's perform squash now

```
git rebase -i HEAD~3
```

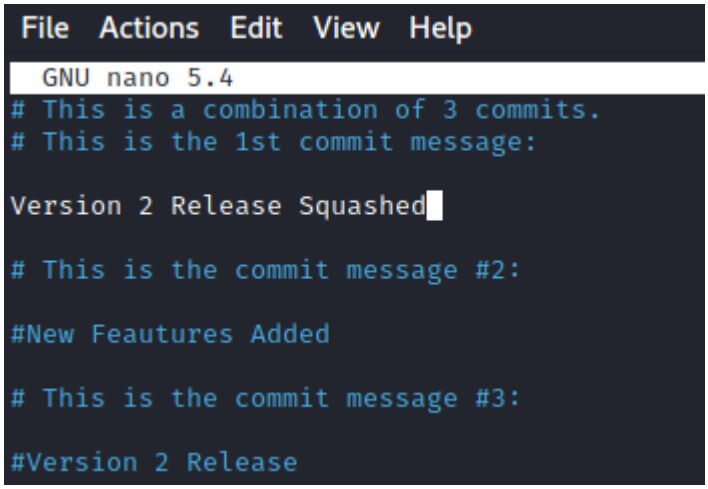
Note: Rebase is an action to rewrite commits and their history "**-i"** is to enter into an interactive mode of rebase **HEAD~n** states to perform our operation on n commits from **HEAD.**

Squashing By Interactive Mode

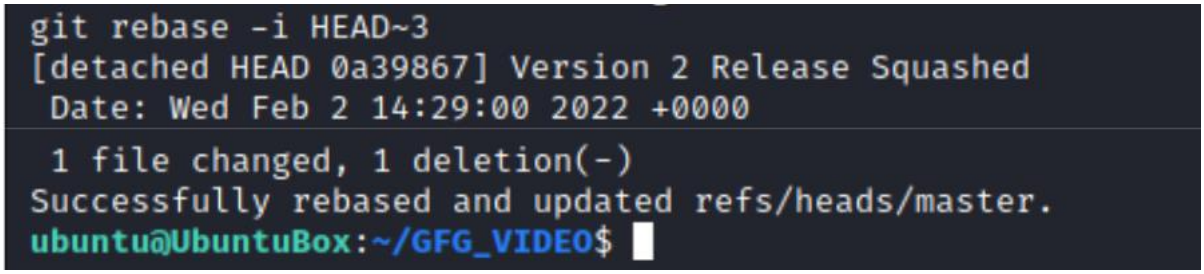
Upon entering the above command we'll get an interactive editor with all our selected commits which is where we'll be performing squash.

We can see we have selected 3 commits at the beginning of the interactive editor, below that we can see the commands list such as pick, reword, edit, squash, etc. To squash 2nd and 3rd commit with 1st commit, so we'll change the first word from pick to squash. whichever commits we want to squash we have to change it to squash from pick.

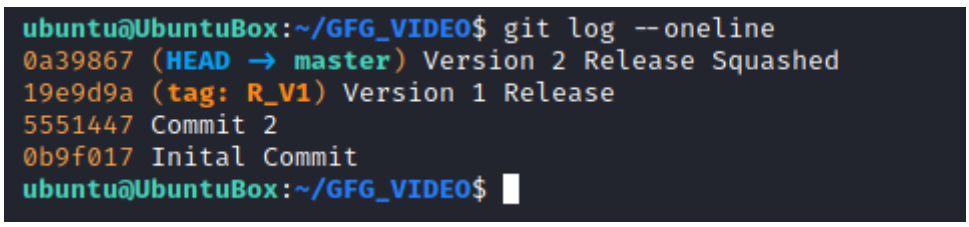
After changing commits from pick to squash save the file and close it, Immediately another editor will be opened where we have to enter the latest commit message. Enter the latest commit message and comment on the remaining old messages.



After adding the latest commit message save the file and exit the file. Now it shows rebasing is successful.



Now if we see our GFG_VIDEO log we can observe our 3 commits after version 1 release are squashed into 1 commit.



Git Squash vs Rebase

Git Squash	Git Rebase
You can combine multiple commits into a single commit.	Applies commits on top of the different base commits.
After using git squash your commit history will be more clean and organized.	You can form a new branch by using previous commits.
Must be done on private branches.	Rebasing can be done on a feature branch or shared branch.
By using squishing you can maintain clean and logical commit history.	By using rebase command you can keep your branch up to date.