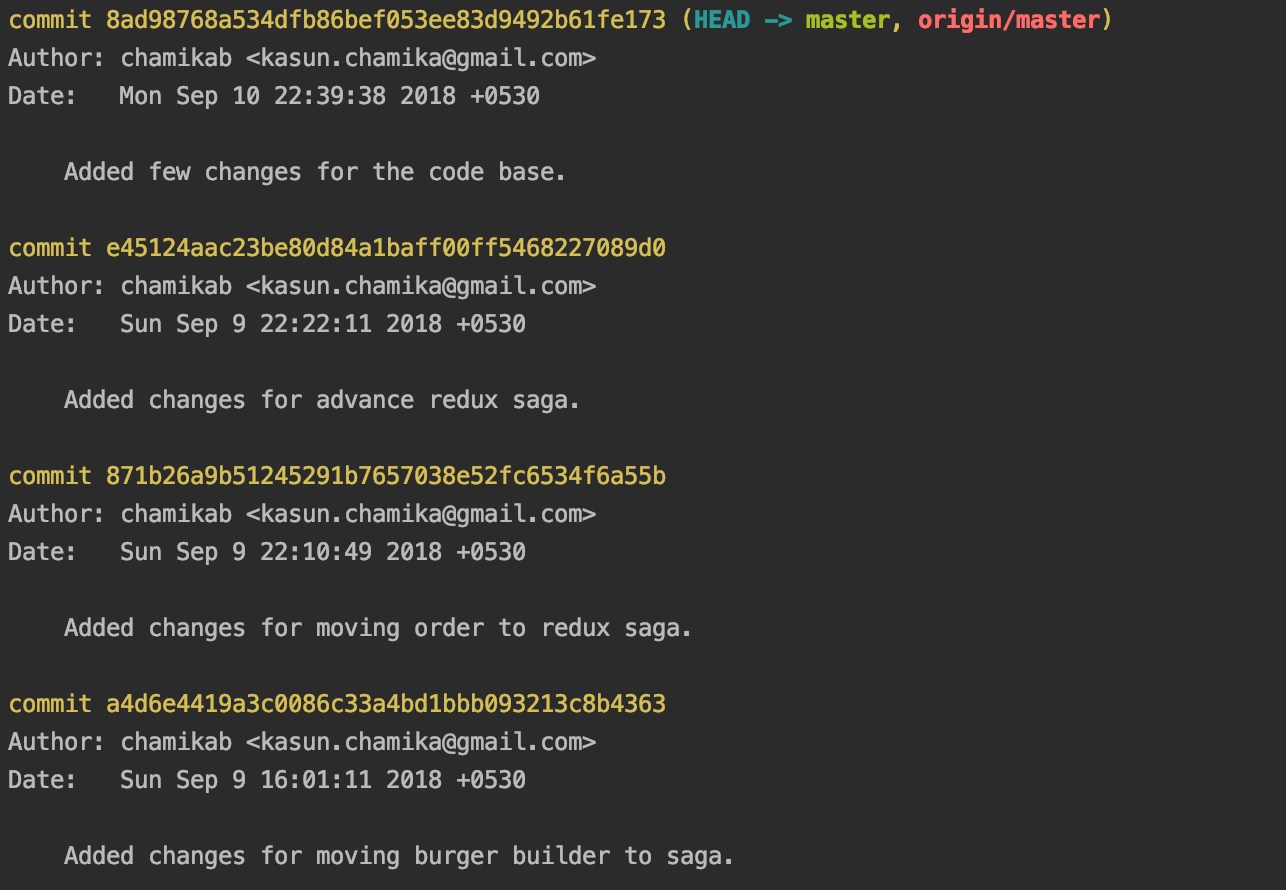
**When to Use**

Let’s think of a scenario where you have a branch called feature which has not released to the production and a separate team working on another feature and they have a branch called feature2 , then you might want to get few commits from that feature2 branch into your feature branch because of a dependent task that you need to complete, and you don’t want to have all the changes in their branch as that branch also not pushed to the production.

**How to Cherry Pick**

Before you cherry pick makes sure that you are on the branch where you need to pick the commits and double check you have all the latest commits. To be dead sure hit git pull again to update your working branch :)

Next what we need to do is, we need to see the commits logs from the working branch and pick one or more commits which you need to apply to another branch. To do that hit git log (or you can use, git log --oneline) on the current working branch. Then you will see an output similar to below.



git log output.

Then from there pick one or more commit ids and switched to the other branch where you want to apply theses selected commits.

Once you switched to the other branch where you need to apply the commits you selected. Hit the following command for a single commit hash.

git cherry-pick <commit-hash>

if you want to apply more than one hashes you can leave a space between each of the commits and execute the same command as follows.

git cherry-pick <commit-hash1> <commit-hash2> <commit-hash3>

Now you should see your branch has been updated with the selected commits.

**Important Notes**

There are a few important things you might want to know regarding the cherry-picking git commits.

*If you cherry-pick from a public branch, you should consider using*

git cherry-pick -x <commit-hash>

*If you have notes attached to the commit they do not follow the cherry-pick. To bring them over as well, You have to use:*

git notes copy <from> <to>

*If you want to cherry-pick a merge instead of a commit*

git cherry-pick -m 1 <merge\_hash>

**Be careful!** It is preferred to use git merge instead of git cherry-pick for this purpose. When you cherry-pick a merge commit, it collapses all the changes made into that one commit. You lose the commit history.

**One Last Word**

Even though this feature is interesting and awesome, it has been discouraged in the git community. The main reason is that it creates a duplicate commit with the same changes and you lose the ability to track the history of the original commit.

Also note that if you are cherry-picking a lot of commits out of order, those will be reflected on your branch in the order you cherry picked, not the on the chronological order of the original commits. Sometimes this may lead to undesirable results in your branch