CHIRAG GHOSH

01

GIT

A CODER'S LIFELINE

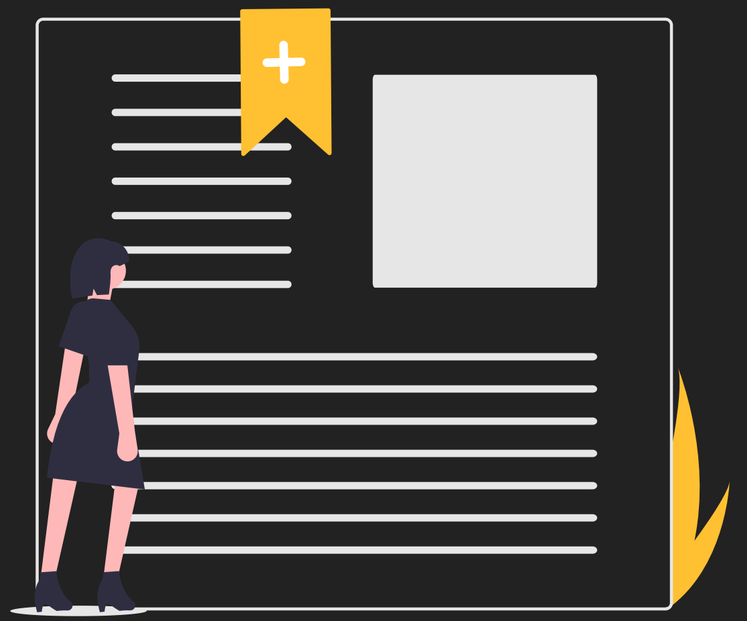
02

Table of Contents

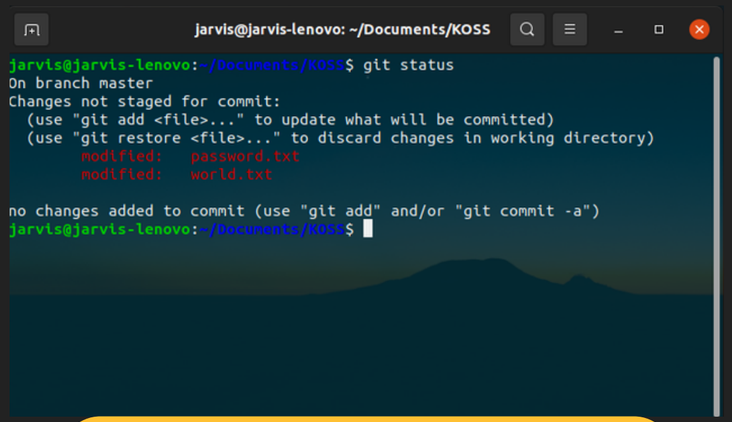
git stash

03

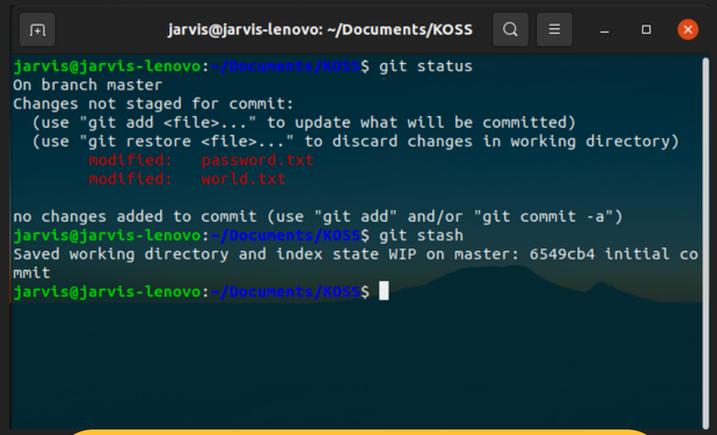
- You made a tasty dish. You cannot eat it because your boss scheduled a meeting. What do you do ? You store it in a refrigerator right ? git stash is something like that.
- Stash is a temporary storage.
- It saves your uncommited changes locally so that you can freely change branches, do other work and come back to work on it again.



04

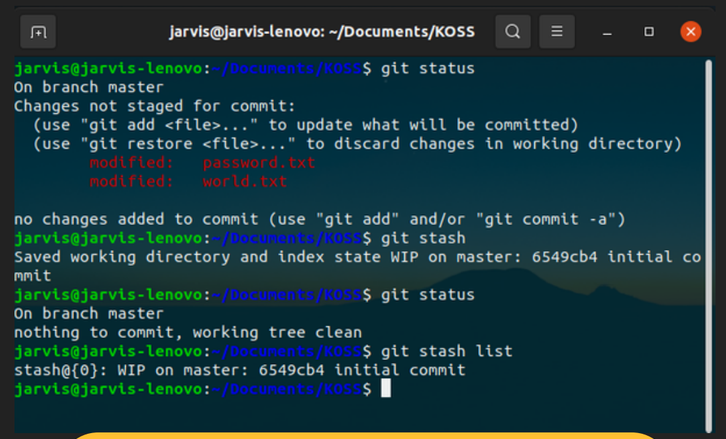


before stash



stash command

use command
"git stash apply"
to reapply changes

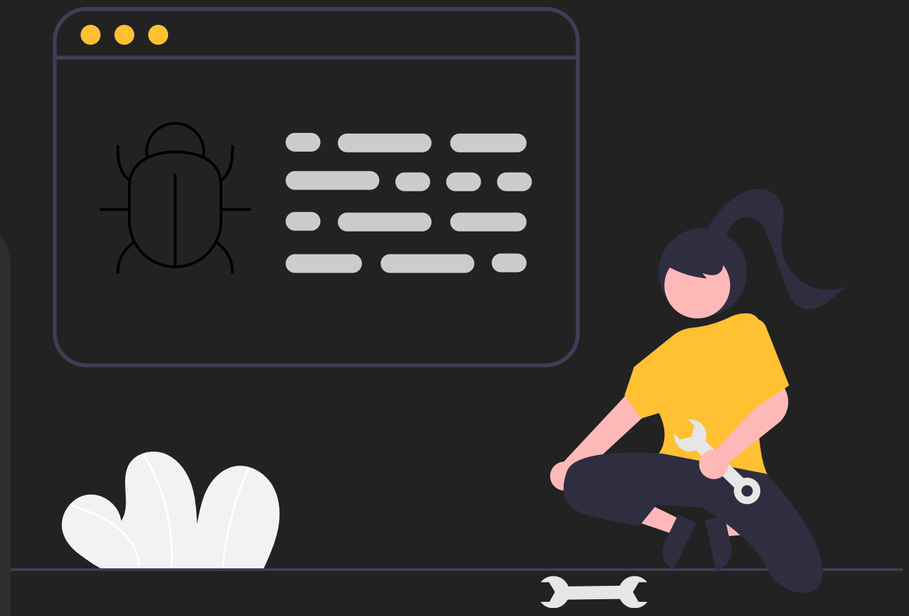


after stash

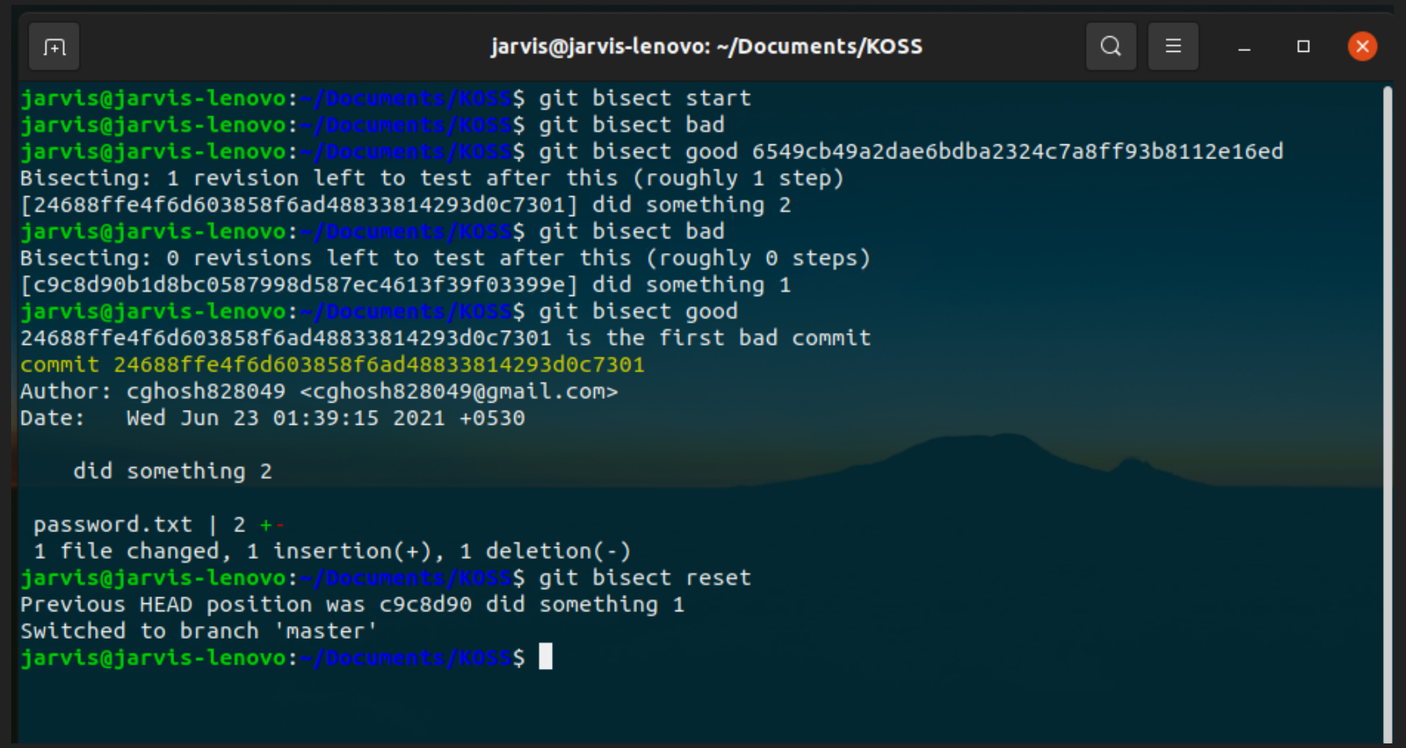
05

git bisect

- You make several commits to your project daily. Suddenly, you realize that the feature you added a week ago is not working. when did it go wrong ?
- Git lets you bisect all the changes made and find the commit which caused such havoc!
- Basically, it takes you back in time to see where you face the problem and where you dont.
- It then pin-points which change has created the problem.



- "git bisect start" starts the process.
- Tell git whether the present state is good or bad.(almost always bad)
- Specify which last change surely doesn't have problem.
- Git will take you to different commits. Try then and see if problem is there or not and report.
- After some repetitions, you have that bad change found.

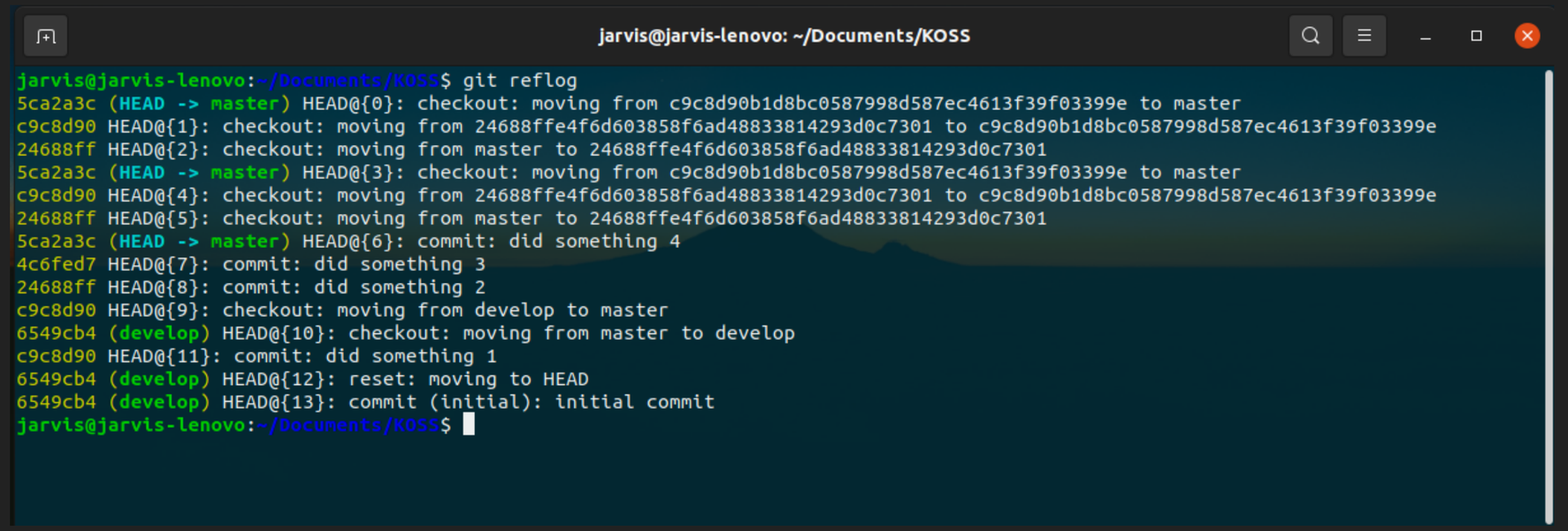


git reflog

07

- Every git repository has a head which specfies our current time.
- Every commit is a timestamp and every branch is a different timeline.
- Every commit and branch-change is recorded by git.
- Reflog is a register of all these records.
- Technically, it is a record of the changes in head.

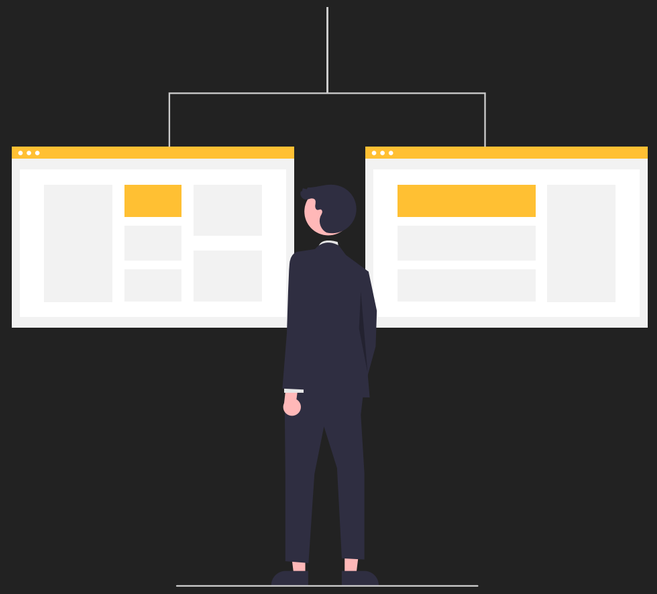
- git reflog is a complete history of the changes in the repository unlike git log which shows commits in the present branch.
- It allows us to revert or delete any changes in the repository.



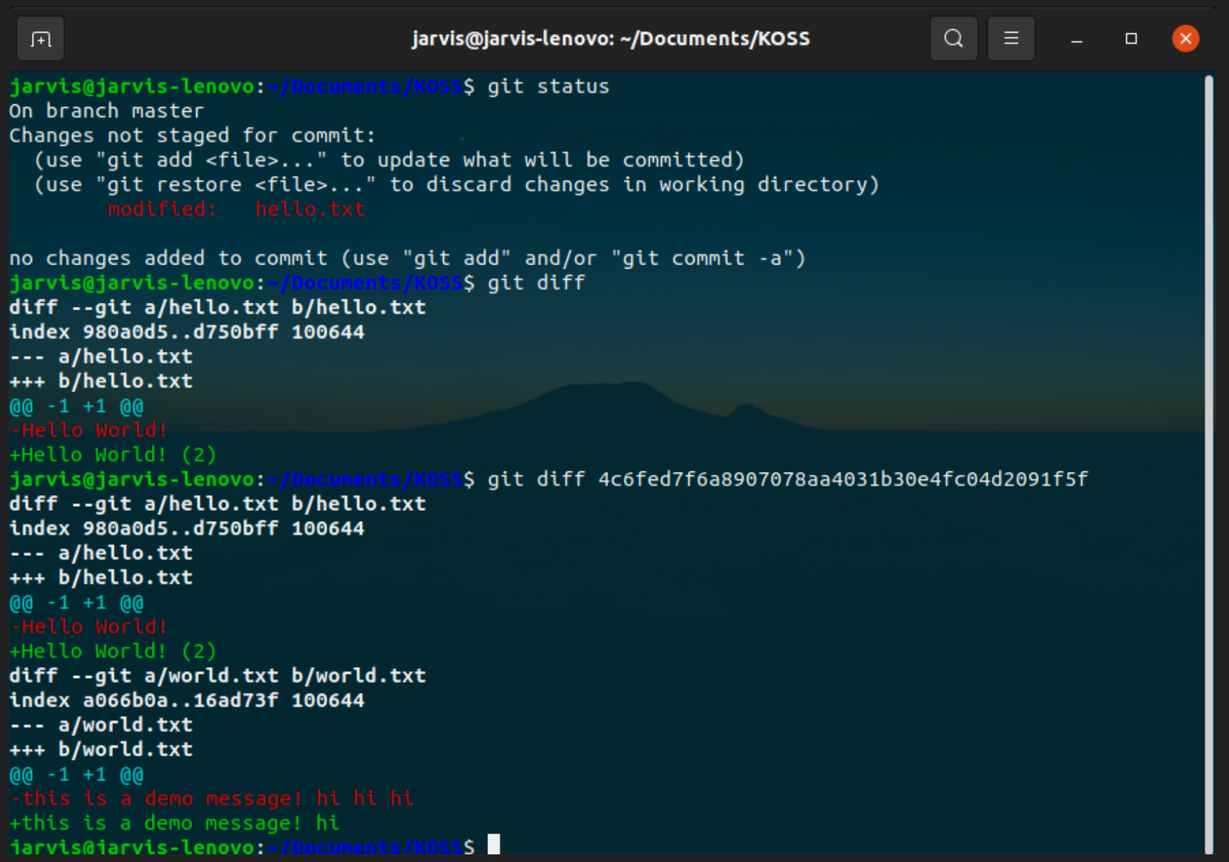
git diff

09

- diff is basically used to see the recent changes.
- you want to know what changes you have made recently ? git diff at rescue.
- git diff checks the present state with the last saved state or any previous state you specify.

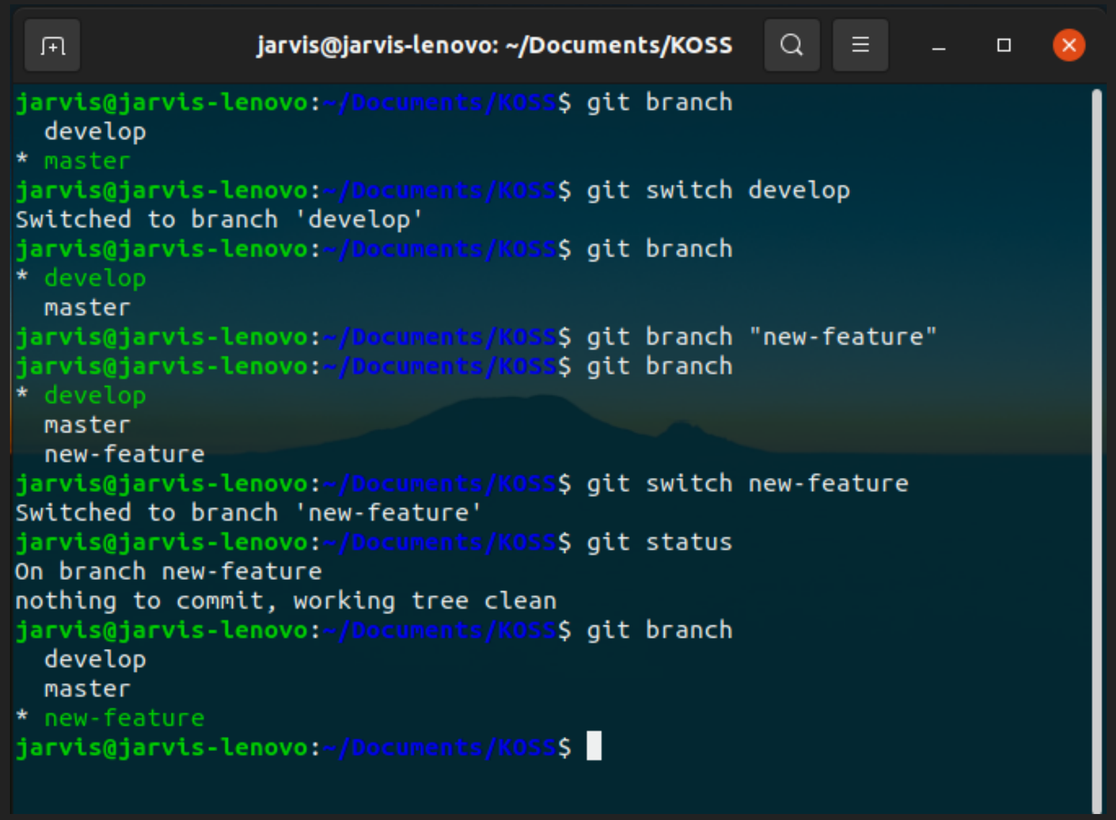


- As shown here, we can find changes relative to any commit made earlier by specifying its id.
- The default is the last commit.



git switch

- switch is used to change timeline a.k.a. branches in git.
- It is different from git checkout since git checkout is used to change branches and also to revert local changes.
- git switch has only one work - changing branches.



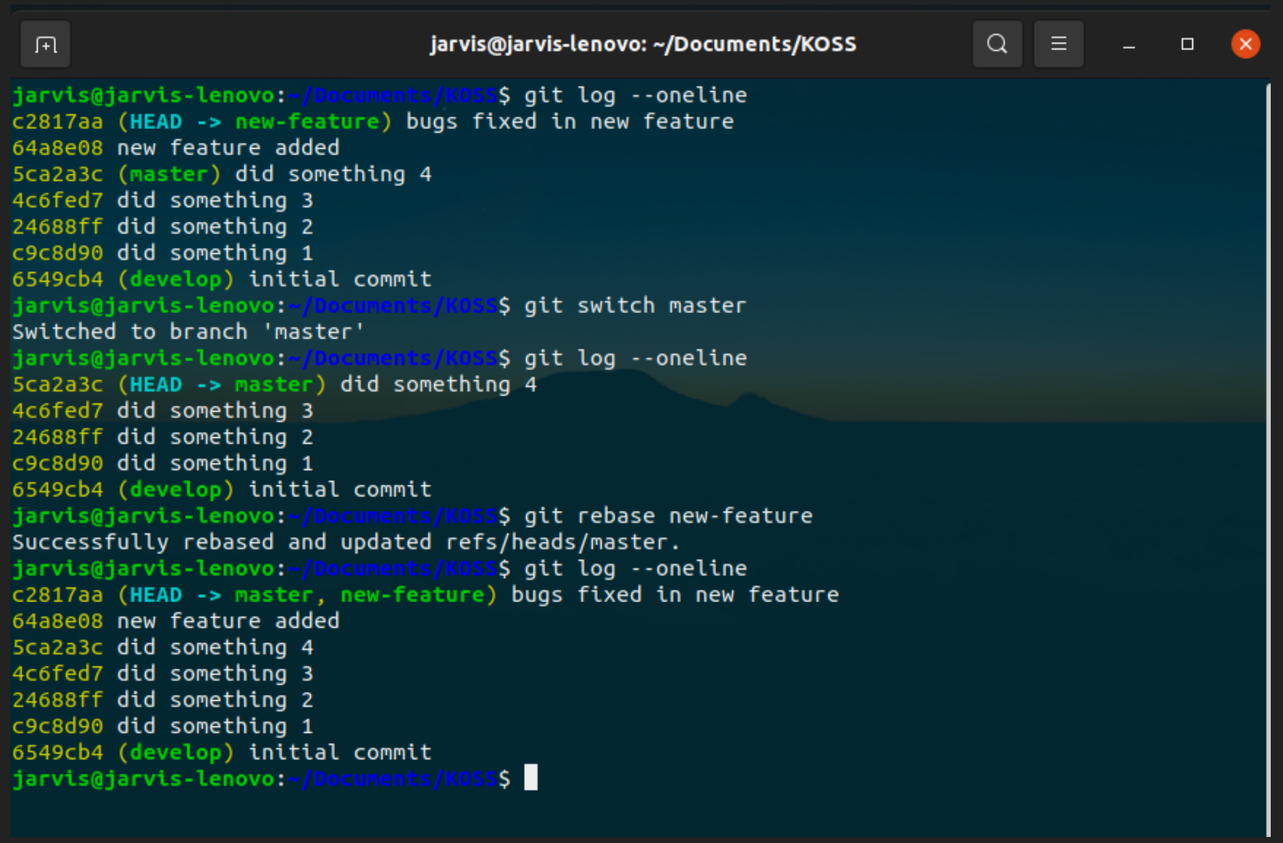
- The syntax can be seen here.

git rebase

- You have a master branch. You are currently working in a different branch for a brand new feature.
- Now after successly completing it, you want to make those changes in the master branch. Git rebase does exactly that.
- It recreates whatever changes you had made in the different branch on top of the master branch.
- It creates new commits in master branch but the changes are the same.

The main advantage of rebase is that I get all the commits I did individually. So I can change any particular commit in the future.

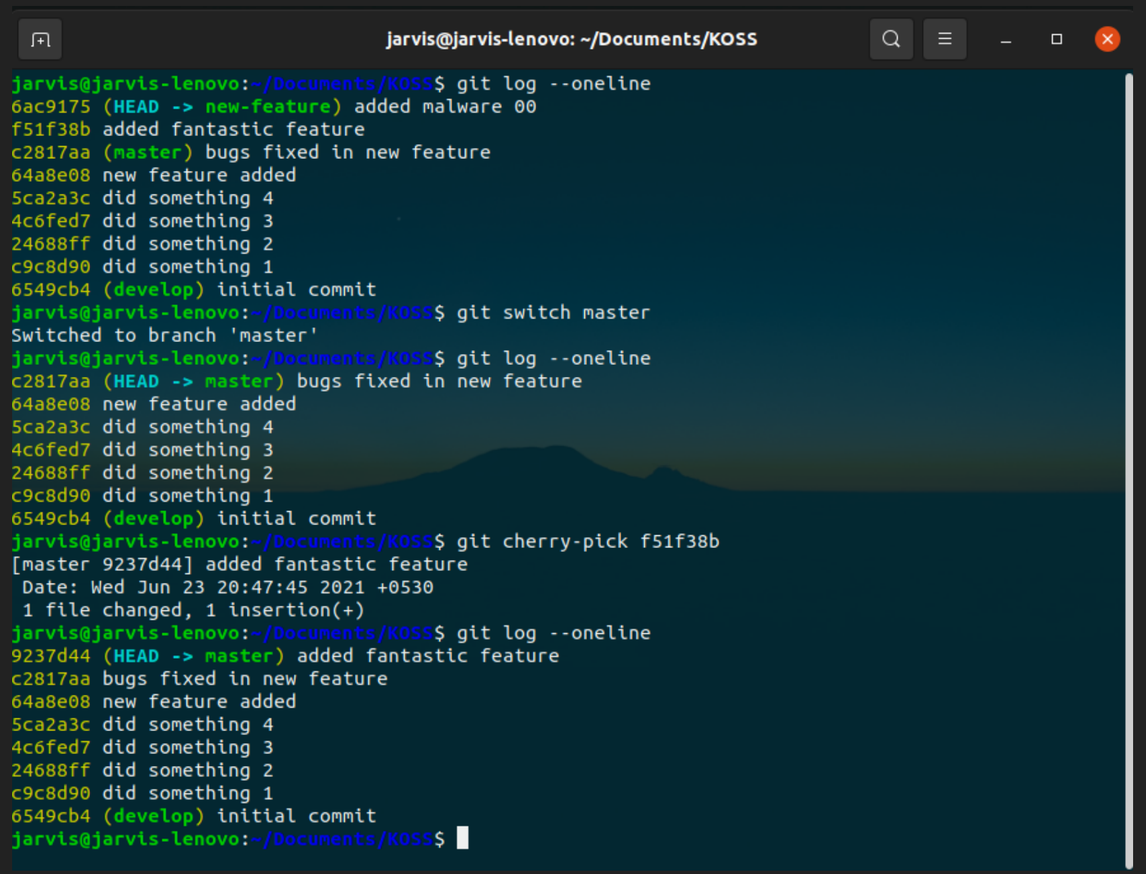
- The syntax can be seen here.
- I am on the master branch and I rebased the new-feature commits on top of it.



git cherry-pick

- Now you know how to rebase.Now you can bring commits from your partner's branch into yours.
- But wait. Your partner is an inconsistent coder. Only some of his commits are good.
- This is where cherry-pick is there to help you. It let's you pick particular commits and add to your branch.

- The syntax can be seen here.
- I want the "fantastic feature" but not the "malware".
- So I cherry-pick that particular commit into my master branch.



With this we come to an end today. I hope you understood these super helpful git features.

I hope you will use them in your projects and make your life easier.

HAPPY CODING!

Thank You