



C H I R A G G H O S H



01

G I T

A CODER'S LIFELINE





02

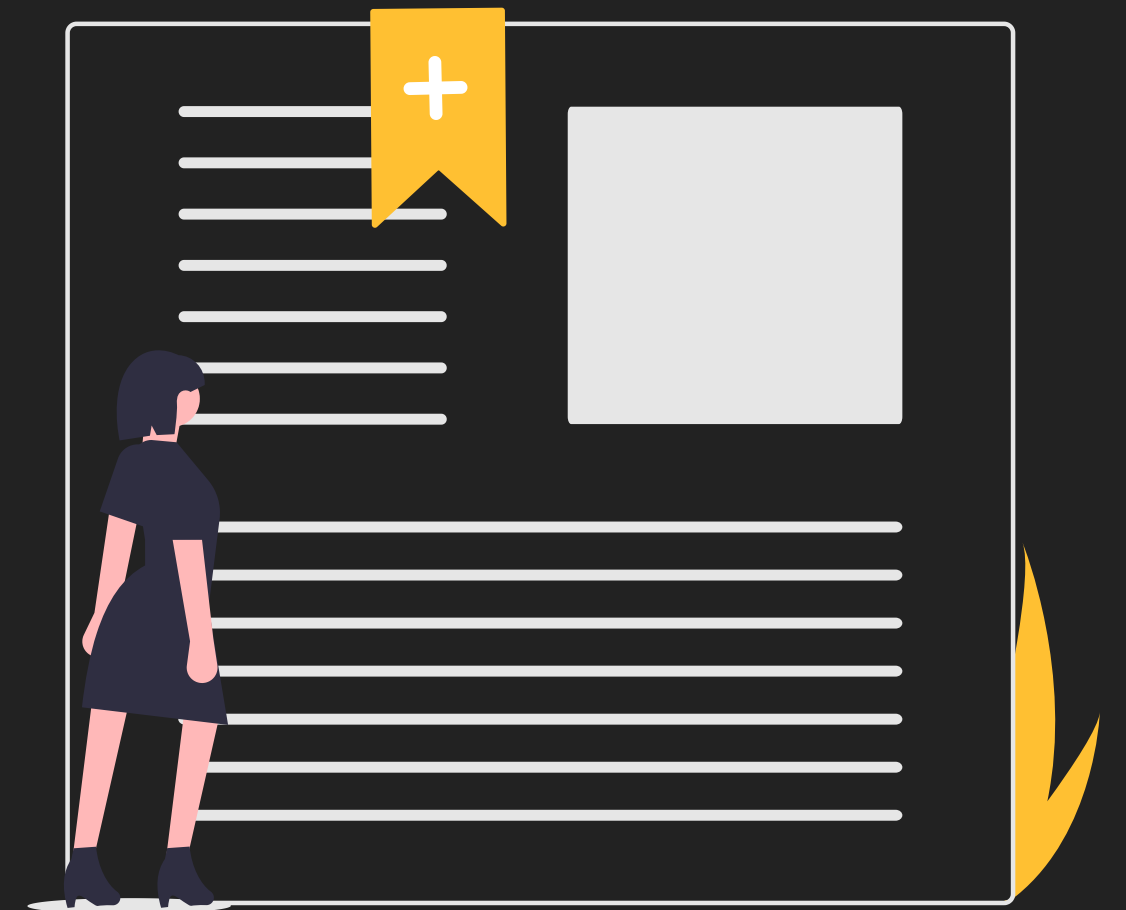
Table of Contents

- git stash
- git bisect
- git reflog
- git diff
- git switch
- git rebase
- git cherry-pick

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 uncommitted changes locally so that you can freely change branches, do other work and come back to work on it again.





04

```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   password.txt
        modified:   world.txt

no changes added to commit (use "git add" and/or "git commit -a")
jarvis@jarvis-lenovo:~/Documents/KOSS$
```

before stash



```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   password.txt
        modified:   world.txt

no changes added to commit (use "git add" and/or "git commit -a")
jarvis@jarvis-lenovo:~/Documents/KOSS$ git stash
Saved working directory and index state WIP on master: 6549cb4 initial co
mmmit
jarvis@jarvis-lenovo:~/Documents/KOSS$
```

stash command



use command
"git stash apply"
to reapply changes

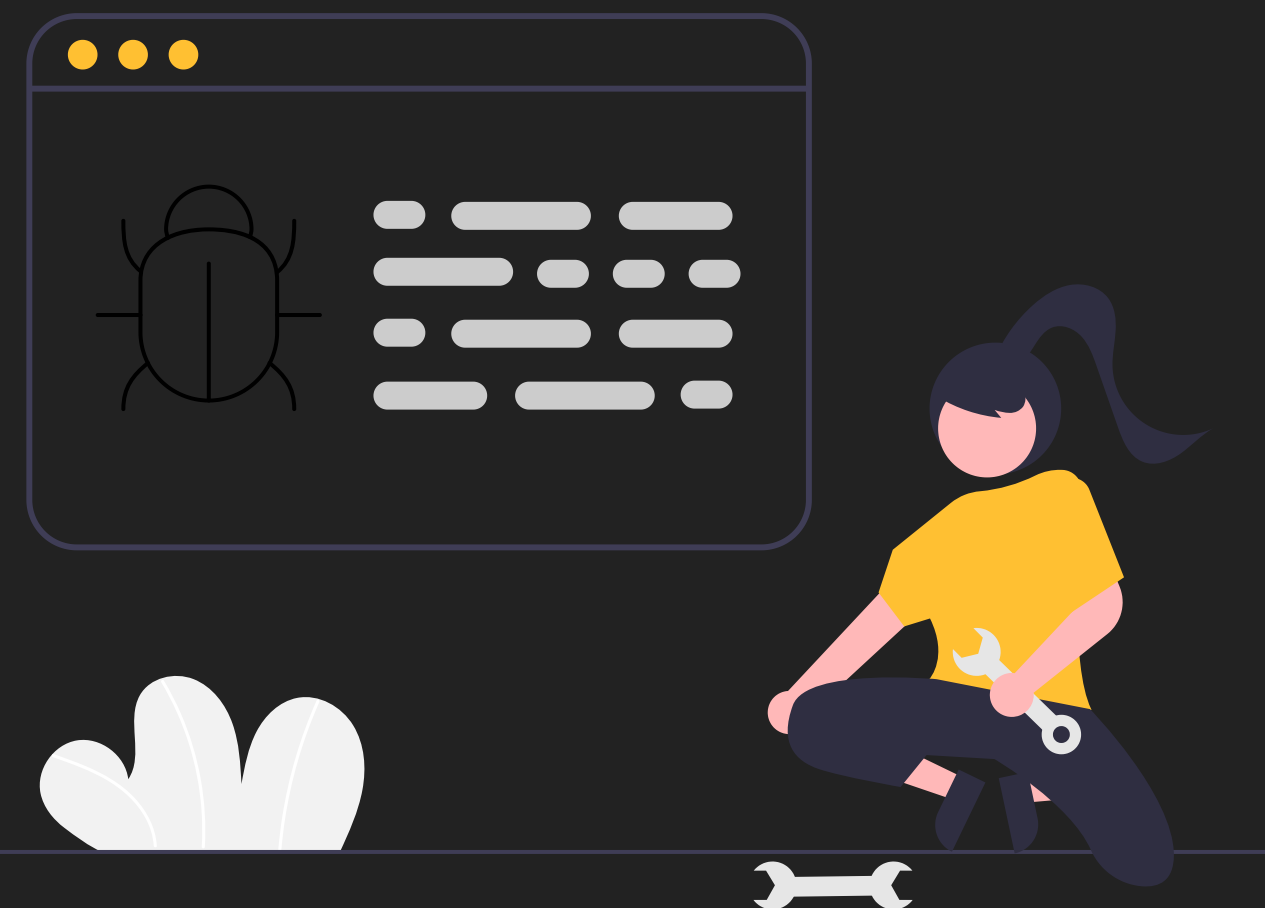
```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   password.txt
        modified:   world.txt

no changes added to commit (use "git add" and/or "git commit -a")
jarvis@jarvis-lenovo:~/Documents/KOSS$ git stash
Saved working directory and index state WIP on master: 6549cb4 initial co
mmmit
jarvis@jarvis-lenovo:~/Documents/KOSS$ git status
On branch master
nothing to commit, working tree clean
jarvis@jarvis-lenovo:~/Documents/KOSS$ git stash list
stash@{0}: WIP on master: 6549cb4 initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$
```

after stash

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.

```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git bisect start
jarvis@jarvis-lenovo:~/Documents/KOSS$ git bisect bad
jarvis@jarvis-lenovo:~/Documents/KOSS$ git bisect good 6549cb49a2dae6bdba2324c7a8ff93b8112e16ed
Bisecting: 1 revision left to test after this (roughly 1 step)
[24688ffe4f6d603858f6ad48833814293d0c7301] did something 2
jarvis@jarvis-lenovo:~/Documents/KOSS$ git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[c9c8d90b1d8bc0587998d587ec4613f39f03399e] did something 1
jarvis@jarvis-lenovo:~/Documents/KOSS$ git bisect good
24688ffe4f6d603858f6ad48833814293d0c7301 is the first bad commit
commit 24688ffe4f6d603858f6ad48833814293d0c7301
Author: cghosh828049 <cghosh828049@gmail.com>
Date:   Wed Jun 23 01:39:15 2021 +0530

    did something 2

password.txt | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
jarvis@jarvis-lenovo:~/Documents/KOSS$ git bisect reset
Previous HEAD position was c9c8d90 did something 1
Switched to branch 'master'
jarvis@jarvis-lenovo:~/Documents/KOSS$
```

git reflog

07

- Every git repository has a head which specifies 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.

08

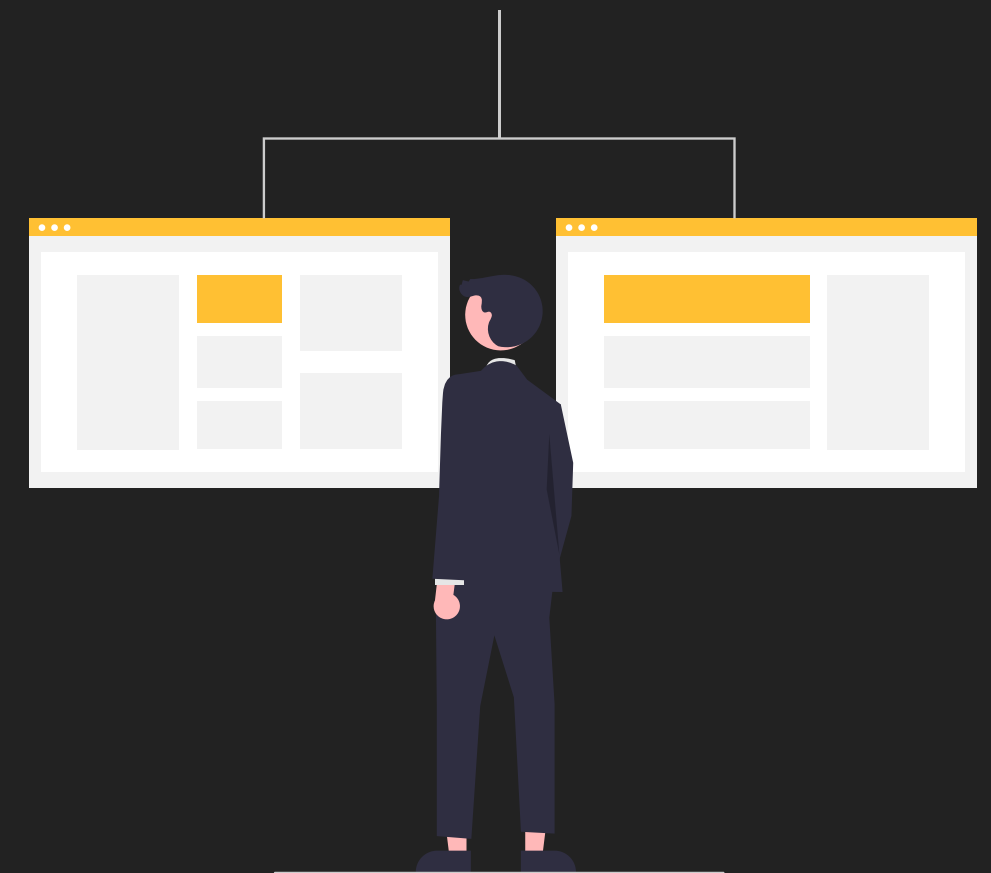
```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git reflog
5ca2a3c (HEAD -> master) HEAD@{0}: checkout: moving from c9c8d90b1d8bc0587998d587ec4613f39f03399e to master
c9c8d90 HEAD@{1}: checkout: moving from 24688ffe4f6d603858f6ad48833814293d0c7301 to c9c8d90b1d8bc0587998d587ec4613f39f03399e
24688ff HEAD@{2}: checkout: moving from master to 24688ffe4f6d603858f6ad48833814293d0c7301
5ca2a3c (HEAD -> master) HEAD@{3}: checkout: moving from c9c8d90b1d8bc0587998d587ec4613f39f03399e to master
c9c8d90 HEAD@{4}: checkout: moving from 24688ffe4f6d603858f6ad48833814293d0c7301 to c9c8d90b1d8bc0587998d587ec4613f39f03399e
24688ff HEAD@{5}: checkout: moving from master to 24688ffe4f6d603858f6ad48833814293d0c7301
5ca2a3c (HEAD -> master) HEAD@{6}: commit: did something 4
4c6fed7 HEAD@{7}: commit: did something 3
24688ff HEAD@{8}: commit: did something 2
c9c8d90 HEAD@{9}: checkout: moving from develop to master
6549cb4 (develop) HEAD@{10}: checkout: moving from master to develop
c9c8d90 HEAD@{11}: commit: did something 1
6549cb4 (develop) HEAD@{12}: reset: moving to HEAD
6549cb4 (develop) HEAD@{13}: commit (initial): initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$
```




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.

```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt

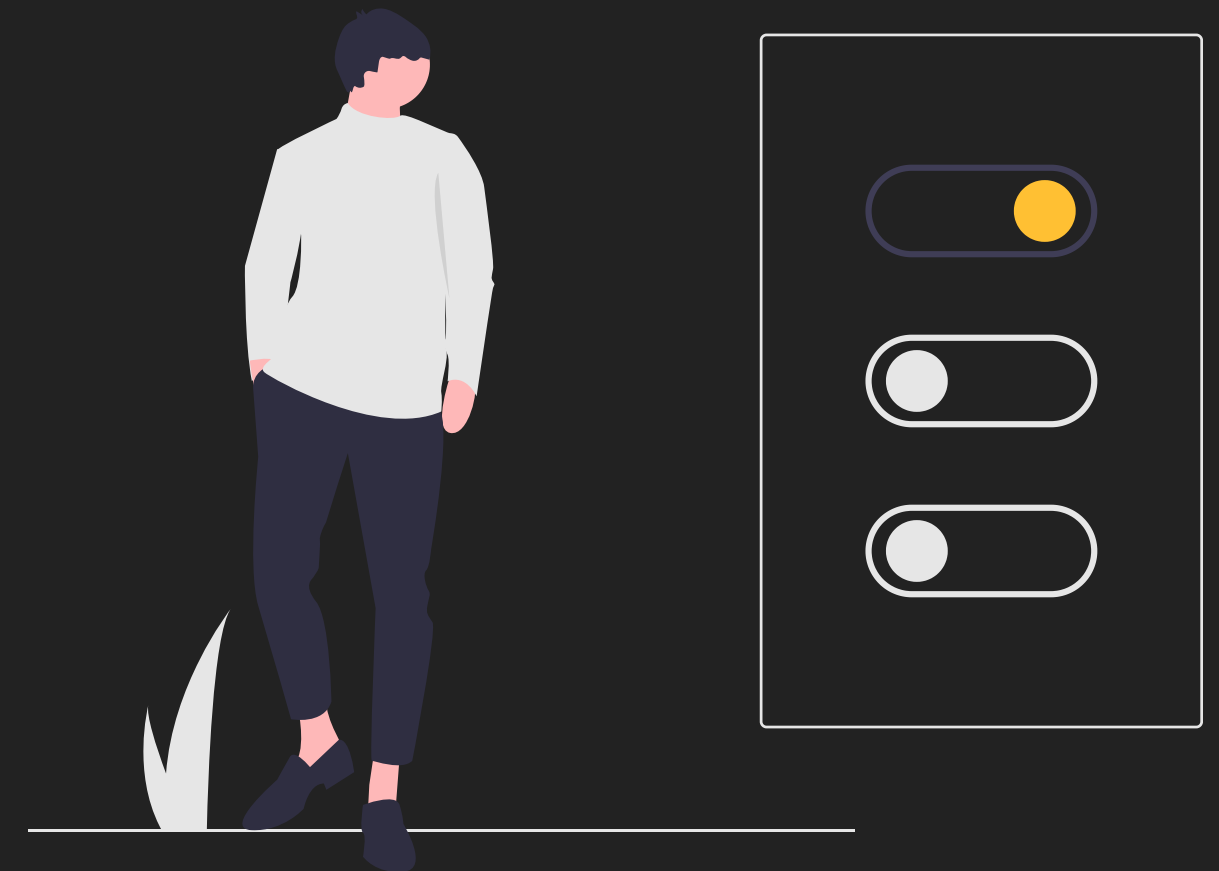
no changes added to commit (use "git add" and/or "git commit -a")
jarvis@jarvis-lenovo:~/Documents/KOSS$ git diff
diff --git a/hello.txt b/hello.txt
index 980a0d5..d750bff 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,1 @@
-Hello World!
+Hello World! (2)
jarvis@jarvis-lenovo:~/Documents/KOSS$ git diff 4c6fed7f6a8907078aa4031b30e4fc04d2091f5f
diff --git a/hello.txt b/hello.txt
index 980a0d5..d750bff 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,1 @@
-Hello World!
+Hello World! (2)
diff --git a/world.txt b/world.txt
index a066b0a..16ad73f 100644
--- a/world.txt
+++ b/world.txt
@@ -1,1 @@
-this is a demo message! hi hi hi
+this is a demo message! hi
jarvis@jarvis-lenovo:~/Documents/KOSS$
```



git switch

11

- 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.





```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git branch
develop
* master
jarvis@jarvis-lenovo:~/Documents/KOSS$ git switch develop
Switched to branch 'develop'
jarvis@jarvis-lenovo:~/Documents/KOSS$ git branch
* develop
master
jarvis@jarvis-lenovo:~/Documents/KOSS$ git branch "new-feature"
jarvis@jarvis-lenovo:~/Documents/KOSS$ git branch
* develop
master
new-feature
jarvis@jarvis-lenovo:~/Documents/KOSS$ git switch new-feature
Switched to branch 'new-feature'
jarvis@jarvis-lenovo:~/Documents/KOSS$ git status
On branch new-feature
nothing to commit, working tree clean
jarvis@jarvis-lenovo:~/Documents/KOSS$ git branch
develop
master
* new-feature
jarvis@jarvis-lenovo:~/Documents/KOSS$
```

- The syntax can be seen here.

git rebase

13

- You have a master branch. You are currently working in a different branch for a brand new feature.
- Now after successfully 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.

14

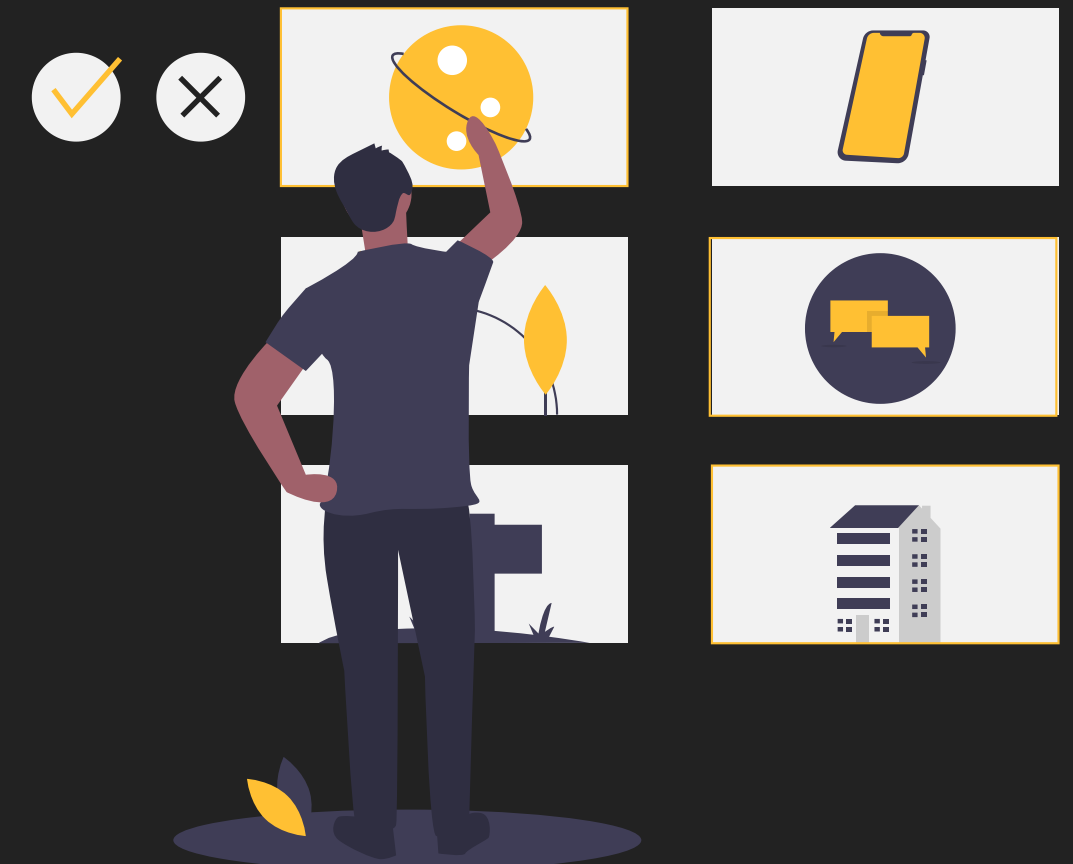
- The syntax can be seen here.
- I am on the master branch and I rebased the new-feature commits on top of it.

```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git log --oneline
c2817aa (HEAD -> new-feature) bugs fixed in new feature
64a8e08 new feature added
5ca2a3c (master) did something 4
4c6fed7 did something 3
24688ff did something 2
c9c8d90 did something 1
6549cb4 (develop) initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$ git switch master
Switched to branch 'master'
jarvis@jarvis-lenovo:~/Documents/KOSS$ git log --oneline
5ca2a3c (HEAD -> master) did something 4
4c6fed7 did something 3
24688ff did something 2
c9c8d90 did something 1
6549cb4 (develop) initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$ git rebase new-feature
Successfully rebased and updated refs/heads/master.
jarvis@jarvis-lenovo:~/Documents/KOSS$ git log --oneline
c2817aa (HEAD -> master, new-feature) bugs fixed in new feature
64a8e08 new feature added
5ca2a3c did something 4
4c6fed7 did something 3
24688ff did something 2
c9c8d90 did something 1
6549cb4 (develop) initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$
```

git cherry-pick

15

- 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.

```
jarvis@jarvis-lenovo: ~/Documents/KOSS
jarvis@jarvis-lenovo:~/Documents/KOSS$ git log --oneline
6ac9175 (HEAD -> new-feature) added malware 00
f51f38b added fantastic feature
c2817aa (master) bugs fixed in new feature
64a8e08 new feature added
5ca2a3c did something 4
4c6fed7 did something 3
24688ff did something 2
c9c8d90 did something 1
6549cb4 (develop) initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$ git switch master
Switched to branch 'master'
jarvis@jarvis-lenovo:~/Documents/KOSS$ git log --oneline
c2817aa (HEAD -> master) bugs fixed in new feature
64a8e08 new feature added
5ca2a3c did something 4
4c6fed7 did something 3
24688ff did something 2
c9c8d90 did something 1
6549cb4 (develop) initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$ git cherry-pick f51f38b
[master 9237d44] added fantastic feature
Date: Wed Jun 23 20:47:45 2021 +0530
1 file changed, 1 insertion(+)
jarvis@jarvis-lenovo:~/Documents/KOSS$ git log --oneline
9237d44 (HEAD -> master) added fantastic feature
c2817aa bugs fixed in new feature
64a8e08 new feature added
5ca2a3c did something 4
4c6fed7 did something 3
24688ff did something 2
c9c8d90 did something 1
6549cb4 (develop) initial commit
jarvis@jarvis-lenovo:~/Documents/KOSS$
```



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