

Git Practical Exam

Git Repository Link : https://github.com/Helly-p/Git_Prac

1. Pull and Merge difference

Pull Request

- Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub.
- Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master

compare: feature1

✓ Able to merge. These branches can be automatically merged.

Feature 1 added.

Write

Preview

H B I ≡ < > ↻ ≡ ≡ ≡ @ ↻ ↶

Feature 1 added.

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Add more commits by pushing to the **Feature** branch on **Helly-pl/Git_Prac**.

Require approval from specific reviewers before merging

Branch protection rules ensure specific people approve pull requests before they're merged.

Add rule

×

Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch

Merging can be performed automatically.

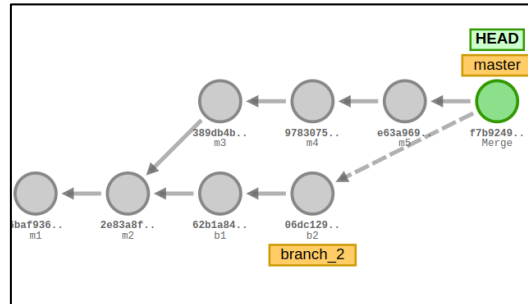
Merge pull request

or view command line instructions.

Merge

- Git merge is used to combine two branches.

```
git merge branch_name
```



Before merge

```
index.html > html > body
6 | <meta name="viewport" content="width=device-width, ini
7 | <title>Document</title>
8 | </head>
9 | <body>
10 | Hello World
11 | Master Branch
12 | </body>
13 | </html>
```

master branch

```
index.html > html > body > Feature1 > Feature2 > Feature3
6 | <meta name="viewport" content="width=device-width, initial-sc
7 | <title>Document</title>
8 | </head>
9 | <body>
10 | Hello World
11 | <Feature1>
12 | <Feature2>
13 | <Feature3>
14 | </body>
15 | </html>
```

Feature branch

After merging master branch into Feature branch : `git merge master`

Merge conflict

```
index.html ! >
index.html > html > body > ? > Feature2 > Feature3
6 | <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 | <title>Document</title>
8 | </head>
9 | <body>
10 | Hello World
11 | <<<<<< HEAD (Current Change)
12 | <Feature1>
13 | <Feature2>
14 | <Feature3>
15 | =====
16 | Master Branch
17 | >>>>>> master (Incoming Change)
18 | </body>
19 | </html>
```

After solving merge conflicts

```
index.html x
index.html > html > body > Feature1 > Feature2 > Feature3
6 | <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 | <title>Document</title>
8 | </head>
9 | <body>
10 |   Hello World
11 |   Master Branch
12 |   <Feature1>
13 |   <Feature2>
14 |   <Feature3>
15 | </body>
16 | </html>
```

Commits on Feb 16, 2023
Master Branch merged in Feature Branch. Helly-p committed 8 minutes ago
Master branch new feature Helly-p committed 20 minutes ago
Feature3 added. Helly-p committed 22 minutes ago
Feature2 added. Helly-p committed 22 minutes ago
Feature1 added. Helly-p committed 26 minutes ago
Initial commit Helly-p committed 28 minutes ago

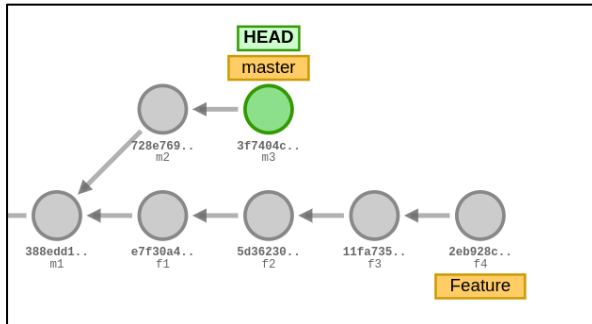
```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline --graph --all
* 21fa3e5 (HEAD -> Feature, origin/Feature) Master Branch merged in Feature Branch.
|
| * 800338f (origin/master, master) Master branch new feature
| * b208da3 Feature3 added.
| * 85c464d Feature2 added.
| * 0c7ec41 Feature1 added.
|/
* 68a7bbd Initial commit
helly@sf-cpu-383:~/Desktop/Git Practical Exam$
```

2. Rebase

Rebasing is the process of moving or combining a sequence of commits to a new base commit.

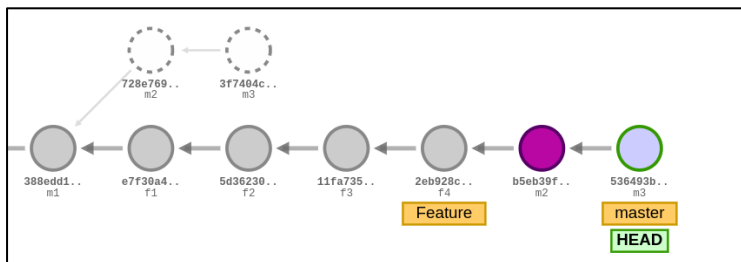
```
git rebase branch_name
```

Before Rebase



```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
94cba43 (HEAD -> master, origin/master) Rebase2
4b31136 Rebase
800338f Master branch new feature
68a7bbd Initial commit
```

After Rebase : `git rebase Feature`



```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
2527439 (HEAD -> master) Rebase2
787411f Rebase
21fa3e5 (origin/Feature, Feature) Master Branch merged in Feature Branch.
800338f Master branch new feature
b208da3 Feature3 added.
85c464d Feature2 added.
0c7ec41 Feature1 added.
68a7bbd Initial commit
```

```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline --graph --all
* 2527439 (HEAD -> master) Rebase2
* 787411f Rebase
* 21fa3e5 (origin/Feature, Feature) Master Branch merged in Feature Branch.
|
| * b208da3 Feature3 added.
| * 85c464d Feature2 added.
| * 0c7ec41 Feature1 added.
| * 94cba43 (origin/master) Rebase2
| * 4b31136 Rebase
|
| * 800338f Master branch new feature
|
| * 68a7bbd Initial commit
```

Here, in **master** branch - 2 commits : 'Rebase', 'Rebase 2'.

After `git rebase Feature` it re-writes the project history and creates brand new commits for each commit in the original branch and make tree in sequential form.

3. Change commit message

Using --amend

The git commit --amend command is a convenient way to modify the most recent commit and to edit commit message.

```
git commit --amend
```

Before

```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git add .
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git commit -m "Msg 1 commit"
[Feature 58b8a75] Msg 1 commit
1 file changed, 1 insertion(+)
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git commit --amend
```

After

```
GNU nano 6.2 /home/helly/Desktop/Git Practical Exam/
Message 1 commit

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Feb 16 14:42:00 2023 +0530
#
# On branch Feature
# Changes to be committed:
#   modified:   index.html
#
# Changes not staged for commit:
#   modified:   index.html
#
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
80b7127 (HEAD -> Feature) Message 1 commit
cb5cbe0 (origin/master, master) Merge branch 'master' of https://github.com/Helly-p/Git_Prac
2527439 Rebase2
```

The commit message of hash **80b7127** is updated from “Msg 1 commit” to “Message 1 commit”.

But this command only changes the message of last commit. For multiple commit messages to change use interactive rebase.

Using Interactive Rebase

Git interactive rebase allows you to change individual commits, squash commits together, drop commits or change the order of the commits.

```
git rebase -i HEAD~5
```

Before

```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
21fa3e5 (HEAD -> Feature, origin/Feature) Master Branch merged in Feature Branch.
800338f Master branch new feature
b208da3 Feature3 added.
85c464d Feature2 added.
0c7ec41 Feature1 added.
68a7bbd Initial commit
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git rebase -i HEAD~4
```

After

```
GNU nano 6.2 /home/helly/
pick 0c7ec41 Feature1 added.
reword 85c464d Feature2 added.
pick b208da3 Feature3 added.
pick 800338f Master branch new feature[]

# Rebase 68a7bbd..21fa3e5 onto 68a7bbd (4 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                       commit's log message, unless -C is used, in which case
#                       keep only this commit's message; -c is same as -C but
#                       opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit>] -c <commit>] <label> [# <oneline>]
# . create a merge commit using the original merge commit's
#   message (or the oneline, if no original merge commit was
```

```
GNU nano 6.2 /home/helly/
EDited commit msg of Feature2 added.[]

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Feb 16 13:45:15 2023 +0530
#
# interactive rebase in progress; onto 68a7bbd
# Last commands done (2 commands done):
#   pick 0c7ec41 Feature1 added.
#   reword 85c464d Feature2 added.
# Next commands to do (2 remaining commands):
#   pick b208da3 Feature3 added.
#   pick 800338f Master branch new feature
# You are currently editing a commit while rebasing branch 'Feature' on '68a7bbd'.
#
# Changes to be committed:
#   modified:   index.html
#
```

```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
0ac440c (HEAD) Feature3 added.
9cd2e2d EDited commit msg of Feature2 added.
0c7ec41 Feature1 added.
68a7bbd Initial commit
helly@sf-cpu-383:~/Desktop/Git Practical Exam$
```

Here, we use **reword** (r for short) for editing commit messages. Commit message is changed from “Feature2 added.” to “EDited commit msg of Feature2 added.”

Other use of Interactive rebase

squash (s for short), which melds the commit into the previous one (the one in the line before)

drop (d for short), remove commit

4. Cherry-pick

Cherry-picking in Git stands for applying some commit from one branch into another branch.

```
git cherry-pick commit_hash
```

Before

```
index.html x
index.html > html > body > Feature1 > Feature2 > Feature3
6 | <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 | <title>Document</title>
8 | </head>
9 | <body>
10 |   Hello World
11 |   Master Branch
12 |   <Feature1>
13 |   <Feature2>
14 |   <Feature3>
15 | </body>
16 | </html>
```

```
... index.html ! x
index.html > html
A 6 | <meta name="viewport" content="width=device-width, initial-scale=1.0">
! 7 | <title>Document</title>
8 | </head>
9 | <body>
10 |   Hello World
11 |   Master Branch
12 |   <Feature1>
13 |   <Feature2>
14 |   <Feature3>
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
15 | <<<<< HEAD (Current Change)
16 | =====
17 | | New Commit
18 | | New Commit 2
19 | >>>>> 8189721 (Line 1 added.) (Incoming Change)
20 | </body>
21 | </html>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

git switch -c <new-branch-name>

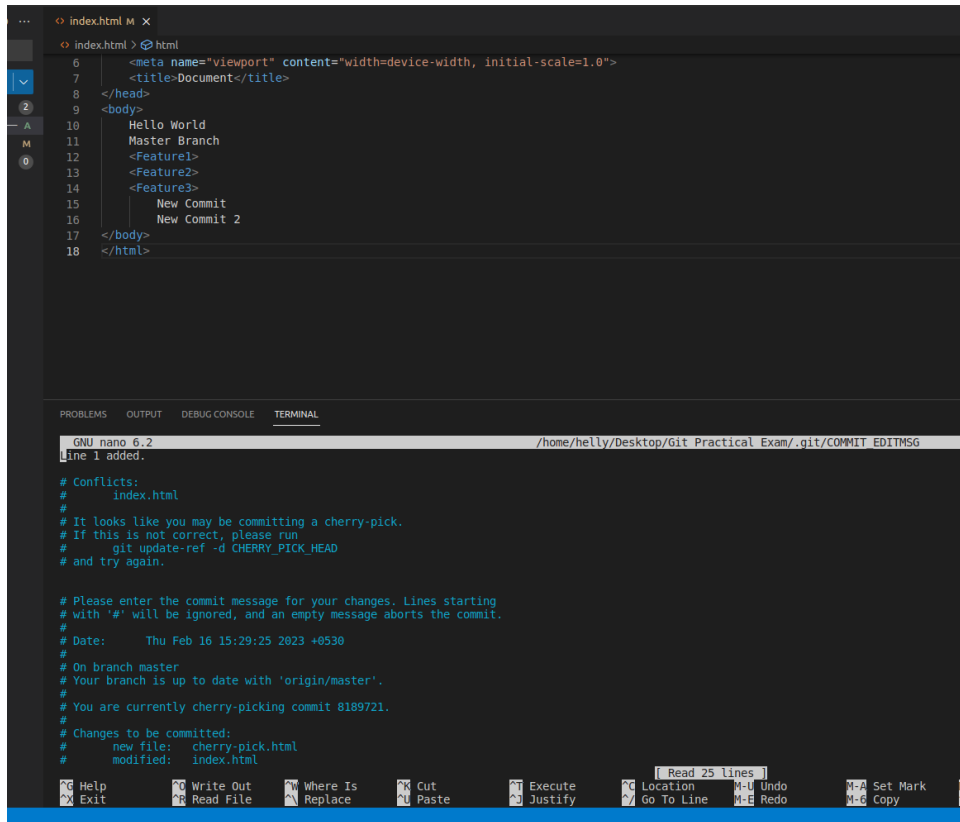
Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 8189721 Line 1 added.
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git checkout 80b7127
Previous HEAD position was 8189721 Line 1 added.
HEAD is now at 80b7127 Message 1 commit
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git checkout master
Previous HEAD position was 80b7127 Message 1 commit
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git cherry-pick 8189721
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
error: could not apply 8189721... Line 1 added.
hint: After resolving the conflicts, mark them with
hint: "git add/rm <paths>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
helly@sf-cpu-383:~/Desktop/Git Practical Exam$
```

After



The screenshot shows a code editor with a file named `index.html` open. The file contains the following HTML code:

```
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Document</title>
8 </head>
9 <body>
10   Hello World
11   Master Branch
12   <Feature1>
13   <Feature2>
14   <Feature3>
15     New Commit
16     New Commit 2
17 </body>
18 </html>
```

Below the code editor, the `TERMINAL` tab is active, showing the output of a `git commit` command. The terminal text is as follows:

```
GNU nano 6.2 /home/helly/Desktop/Git Practical Exam/.git/COMMIT_EDITMSG
line 1 added.

# Conflicts:
#   index.html
#
# It looks like you may be committing a cherry-pick.
# If this is not correct, please run
#   git update-ref -d CHERRY_PICK_HEAD
# and try again.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Thu Feb 16 15:29:25 2023 +0530
#
# On branch master
# Your branch is up to date with 'origin/master'.
#
# You are currently cherry-picking commit 8189721.
#
# Changes to be committed:
#   new file:   cherry-pick.html
#   modified:   index.html
```

At the bottom of the terminal window, there is a row of keyboard shortcuts: `Help`, `Exit`, `Write Out`, `Read File`, `Where Is`, `Replace`, `Cut`, `Paste`, `Execute`, `Justify`, `Location`, `Go To Line`, `Undo`, `M-U`, `Redo`, `M-A`, `Set Mark`, `M-G`, `Copy`.

5. Drop commit

It resets the current branch tip, and also deletes any changes in the working directory and staging area.

```
git reset --hard
```

Before

```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
cb5cbe0 (HEAD -> Feature_2, origin/Feature_2) Merge branch 'master' of https://github.com/Helly-p/Git_Prac
2527439 Rebase2
787411f Rebase
94cba43 Rebase2
4b31136 Rebase
21fa3e5 Master Branch merged in Feature Branch.
800338f Master branch new feature
b208da3 Feature3 added.
85c464d Feature2 added.
0c7ec41 Feature1 added.
68a7bbd Initial commit
```

After

```
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git reset --hard HEAD~1
HEAD is now at 2527439 Rebase2
helly@sf-cpu-383:~/Desktop/Git Practical Exam$ git log --oneline
2527439 (HEAD -> Feature_2) Rebase2
787411f Rebase
21fa3e5 Master Branch merged in Feature Branch.
800338f Master branch new feature
b208da3 Feature3 added.
85c464d Feature2 added.
0c7ec41 Feature1 added.
68a7bbd Initial commit
```

Here, at first the HEAD pointer points to commit **cb5cbe0** but after git reset --hard HEAD~1, HEAD pointer points to commit **2527439** and the first commit is dropped.

We can also use interactive rebase drop (d for short) for dropping commit.