NEW GIT CONCEPTS

KOSS INTERVIEW TASK

Name: Abhishek Kumar
Roll: 21EC30001

CONTENT LAYOUT

- What is Git
- Getting started(creating a repo)
- Making changes
- Viewing project history(logs/diffs)
- branches/switch/merging
- Git stash
- Git bisect
- Git rebase
- Git Cherry-pick

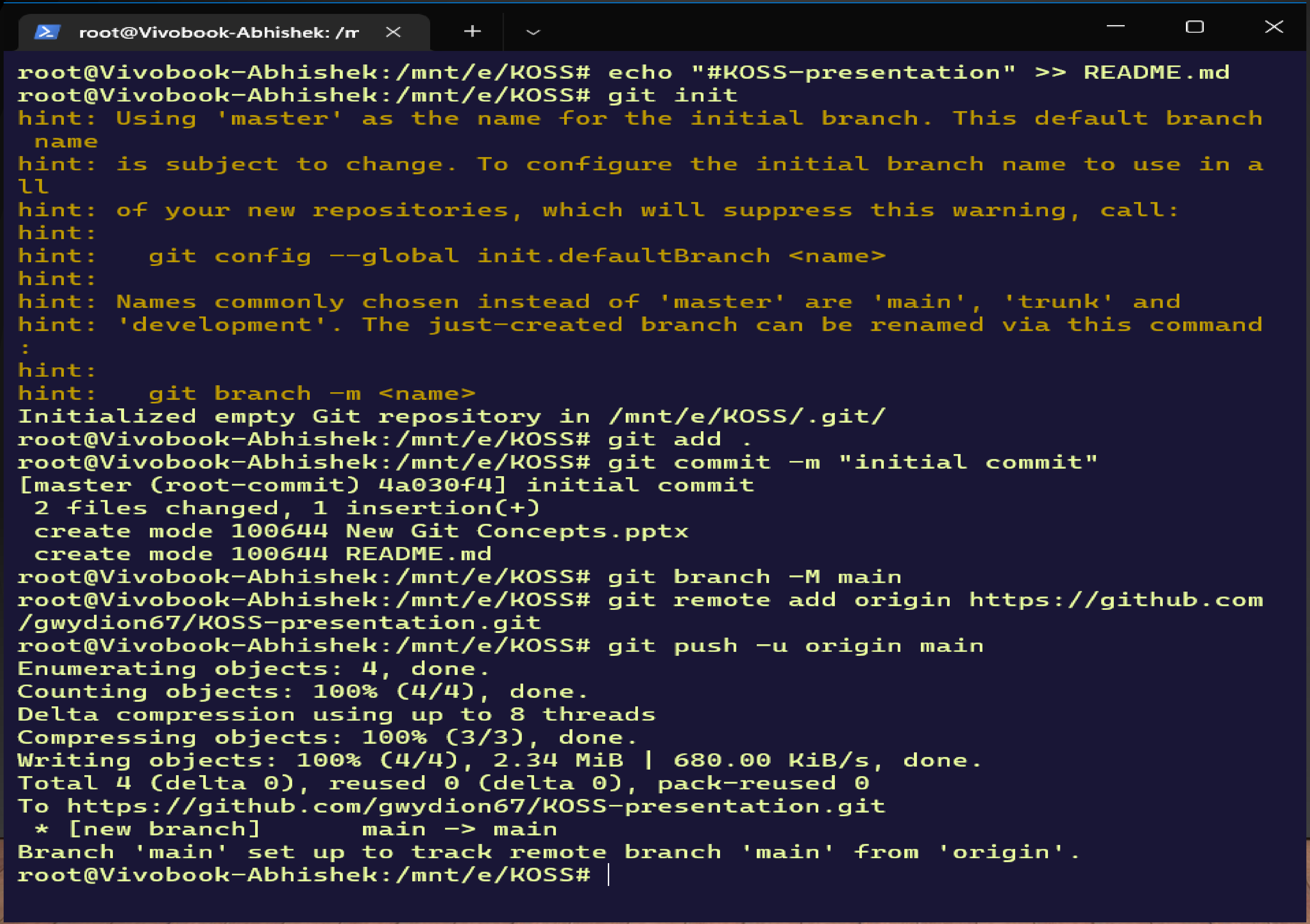
WHAT IS GIT

Git, is a version control system.
It is free, open source and is used by most of the open source software developers to
⇒ Create
⇒ Managing/tracking status of the projects
⇒ Store the code at the local git servers

Basically it is a all in one Swiss knife for programmers

Getting started

$git init	Initialized empty Git repository in .git/
$git add .	take a snapshot of the contents of all files under the current directory
$git commit	permanently store the contents of the index in the repository



root@Vivobook-Abhishek: /m

```
root@Vivobook-Abhishek:/mnt/e/KOSS# echo "#KOSS-presentation" >> README.md
root@Vivobook-Abhishek:/mnt/e/KOSS# git init
hint: Using 'master' as the name for the initial branch. This default branch
 name
hint: is subject to change. To configure the initial branch name to use in a
ll
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command
:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /mnt/e/KOSS/.git/
root@Vivobook-Abhishek:/mnt/e/KOSS# git add .
root@Vivobook-Abhishek:/mnt/e/KOSS# git commit -m "initial commit"
[master (root-commit) 4a030f4] initial commit
 2 files changed, 1 insertion(+)
 create mode 100644 New Git Concepts.pptx
 create mode 100644 README.md
root@Vivobook-Abhishek:/mnt/e/KOSS# git branch -M main
root@Vivobook-Abhishek:/mnt/e/KOSS# git remote add origin https://github.com
/gwydion67/KOSS-presentation.git
root@Vivobook-Abhishek:/mnt/e/KOSS# git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 2.34 MiB | 680.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/gwydion67/KOSS-presentation.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
root@Vivobook-Abhishek:/mnt/e/KOSS# 
```

This is the preview of the said commands . I have used them to create a repo and push them to GitHub

MAKING CHANGES

After modifying some files to add specific files(say file1, file2) to the index

```
$git add file1 file2
```

Using `git status` or `git diff –cached` we can see the changes that will occur after the next commit

`git diff` will show changes that are made but not yet added to the index.

`git status` gives a brief summary of the situation of the changes

```
git commit –a
```

Can be used to commit without using git add beforehand

THE PROJECT HISTORY

Logging

➔git log
➔git log --oneline # more succinct output
➔git log --graph #with a visual graph of branches

TO VEIW THE “undo” history

➔git reflog

View your current state + any merge conflicts

➔git status

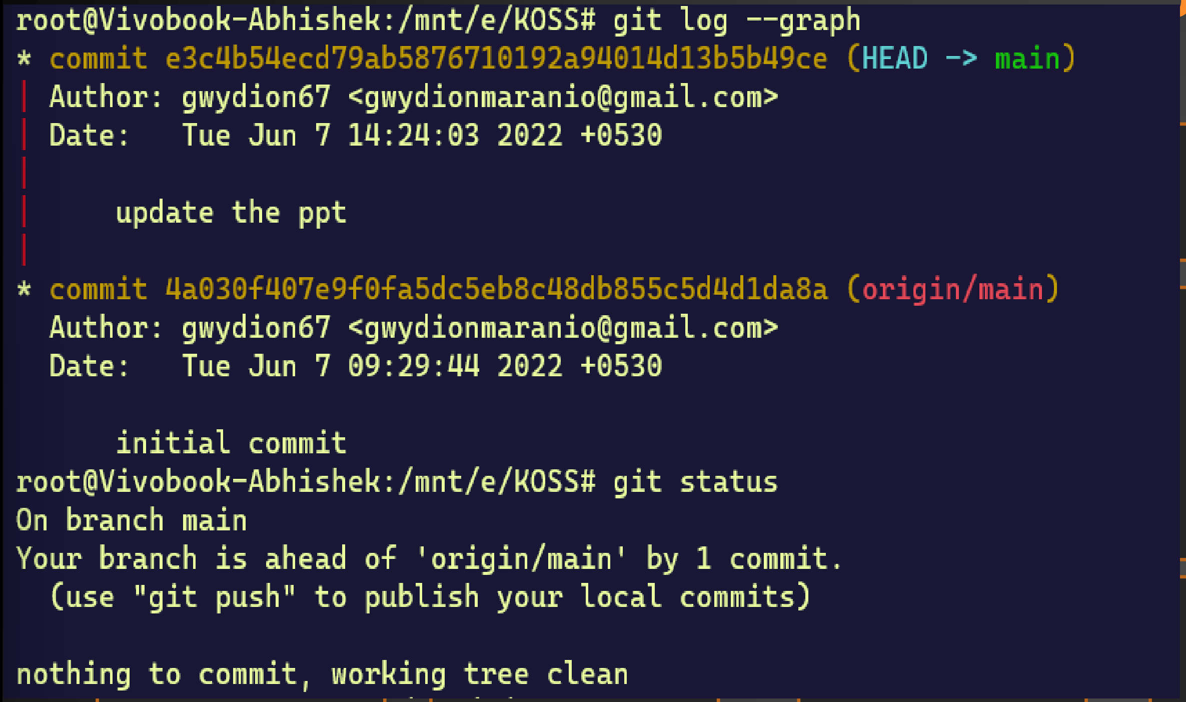
See the differences in your staged (or unstaged) changes

➔git diff--staged # for staged changes
➔git diff #for unstaged changes
➔git diff branch1..branch2 #see differences between two branches

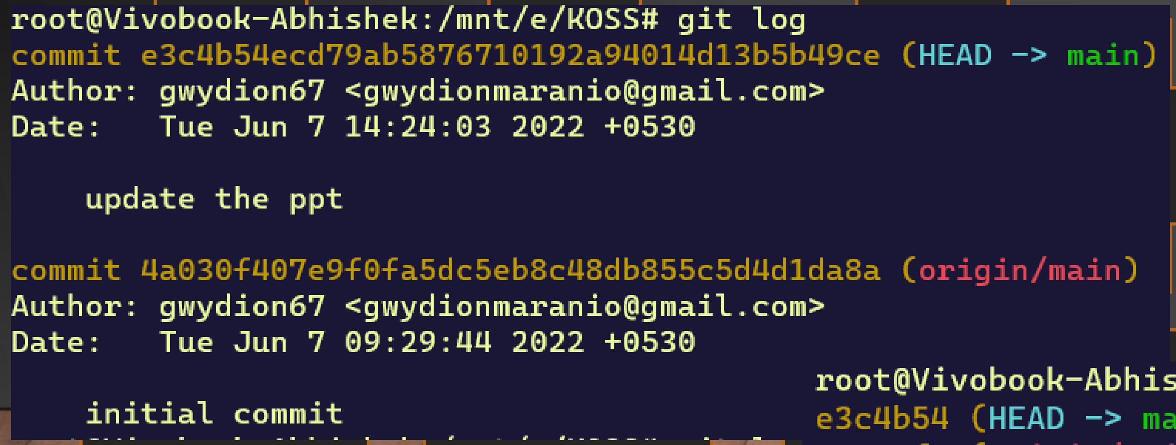
Uncommit the previous changes(this will uncommit and leave the files in the working directory}

➔git reset <commit-sha>

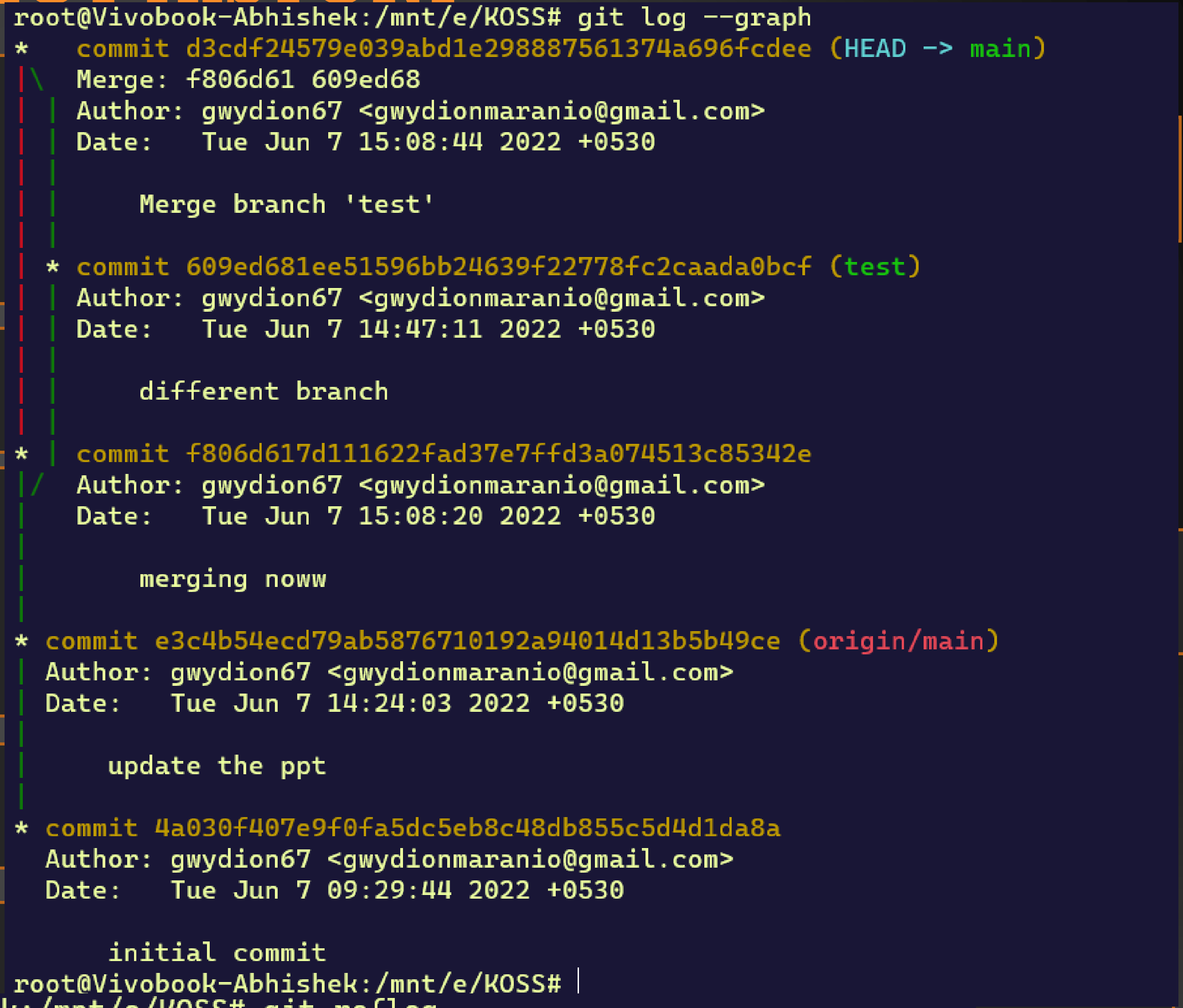
THE PROJECT HISTORY



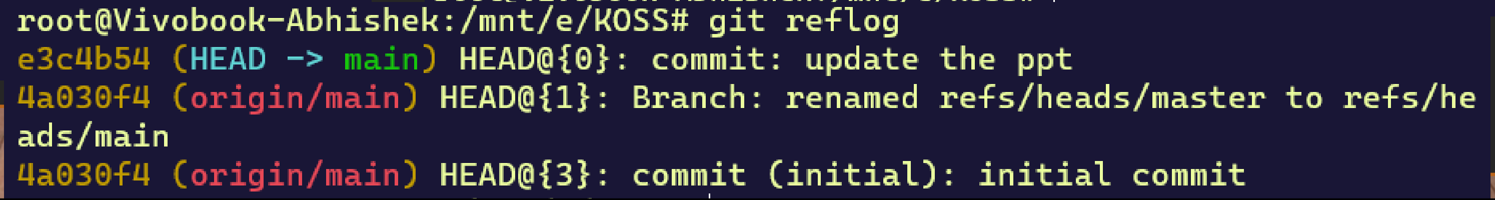
```
root@Vivobook-Abhishek:/mnt/e/KOSS# git log --graph
* commit e3c4b54ecd79ab5876710192a94014d13b5b49ce (HEAD -> main)
| Author: gwydion67 <gwydionmaranio@gmail.com>
| Date:   Tue Jun 7 14:24:03 2022 +0530
|
|     update the ppt
|
* commit 4a030f407e9f0fa5dc5eb8c48db855c5d4d1da8a (origin/main)
  Author: gwydion67 <gwydionmaranio@gmail.com>
  Date:   Tue Jun 7 09:29:44 2022 +0530

      initial commit
root@Vivobook-Abhishek:/mnt/e/KOSS# git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git log
commit e3c4b54ecd79ab5876710192a94014d13b5b49ce (HEAD -> main)
Author: gwydion67 <gwydionmaranio@gmail.com>
Date:   Tue Jun 7 14:24:03 2022 +0530

    update the ppt

commit 4a030f407e9f0fa5dc5eb8c48db855c5d4d1da8a (origin/main)
Author: gwydion67 <gwydionmaranio@gmail.com>
Date:   Tue Jun 7 09:29:44 2022 +0530

    initial commit
```



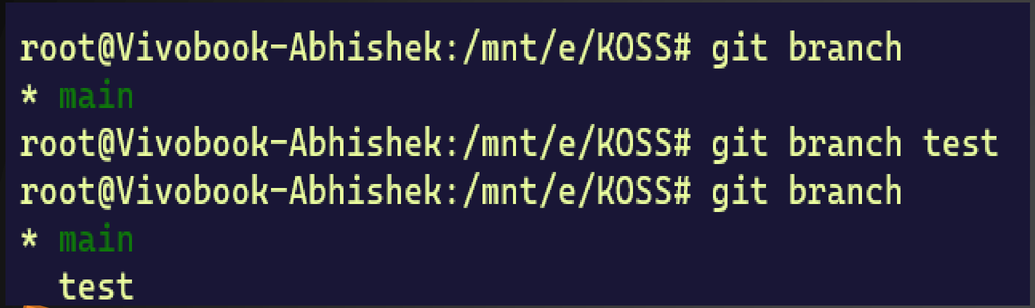
```
root@Vivobook-Abhishek:/mnt/e/KOSS# git log --graph
*   commit d3cdf24579e039abd1e298887561374a696fcdee (HEAD -> main)
|\  Merge: f806d61 609ed68
| | Author: gwydion67 <gwydionmaranio@gmail.com>
| | Date:   Tue Jun 7 15:08:44 2022 +0530
| |
| |     Merge branch 'test'
| |
| * commit 609ed681ee51596bb24639f22778fc2caada0bcf (test)
| | Author: gwydion67 <gwydionmaranio@gmail.com>
| | Date:   Tue Jun 7 14:47:11 2022 +0530
| |
| |     different branch
| |
* | commit f806d617d111622fad37e7ffd3a074513c85342e
|/  Author: gwydion67 <gwydionmaranio@gmail.com>
|   Date:   Tue Jun 7 15:08:20 2022 +0530
|
|       merging noww
|
* commit e3c4b54ecd79ab5876710192a94014d13b5b49ce (origin/main)
| Author: gwydion67 <gwydionmaranio@gmail.com>
| Date:   Tue Jun 7 14:24:03 2022 +0530
|
|     update the ppt
|
* commit 4a030f407e9f0fa5dc5eb8c48db855c5d4d1da8a
  Author: gwydion67 <gwydionmaranio@gmail.com>
  Date:   Tue Jun 7 09:29:44 2022 +0530

      initial commit
root@Vivobook-Abhishek:/mnt/e/KOSS#
```



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git reflog
e3c4b54 (HEAD -> main) HEAD@{0}: commit: update the ppt
4a030f4 (origin/main) HEAD@{1}: Branch: renamed refs/heads/master to refs/he
ads/main
4a030f4 (origin/main) HEAD@{3}: commit (initial): initial commit
```

BRANCHES-SWITCHING/MERGING

Creating a branch:



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git branch
* main
root@Vivobook-Abhishek:/mnt/e/KOSS# git branch test
root@Vivobook-Abhishek:/mnt/e/KOSS# git branch
* main
  test
```

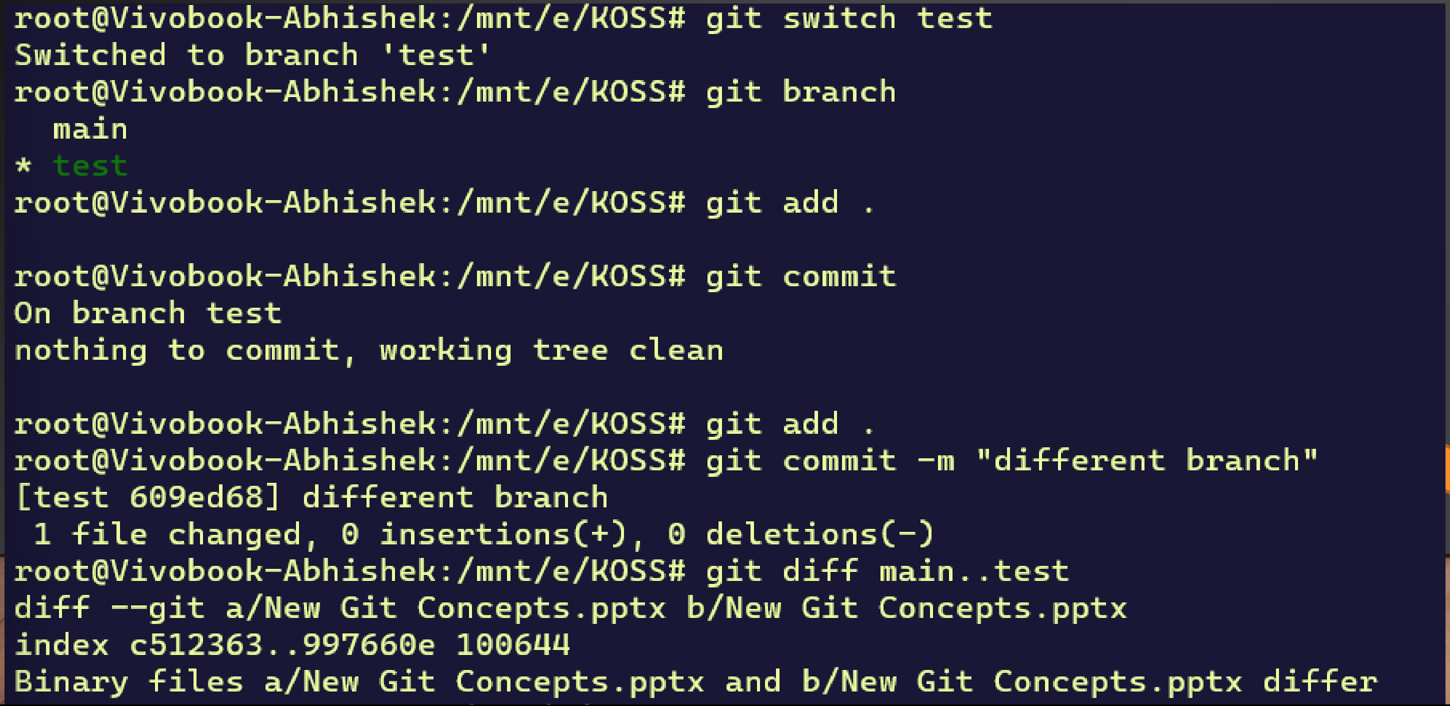
Git branch is used to create a new branch

git switch/git checkout is used to switch between branches

Git merge is used to merge the branch with any other branch

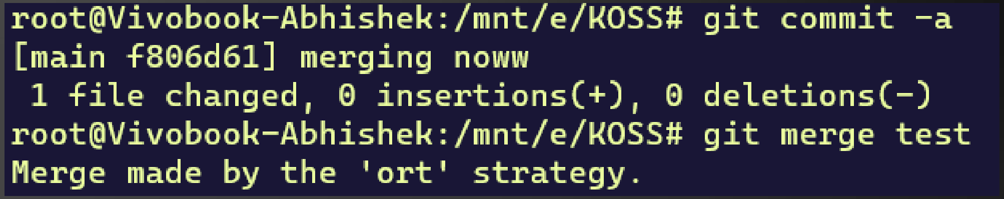
In case of a merge conflict use git diff to check the conflict and merge after resolving the conflict

Switching branch and changing in new branch:

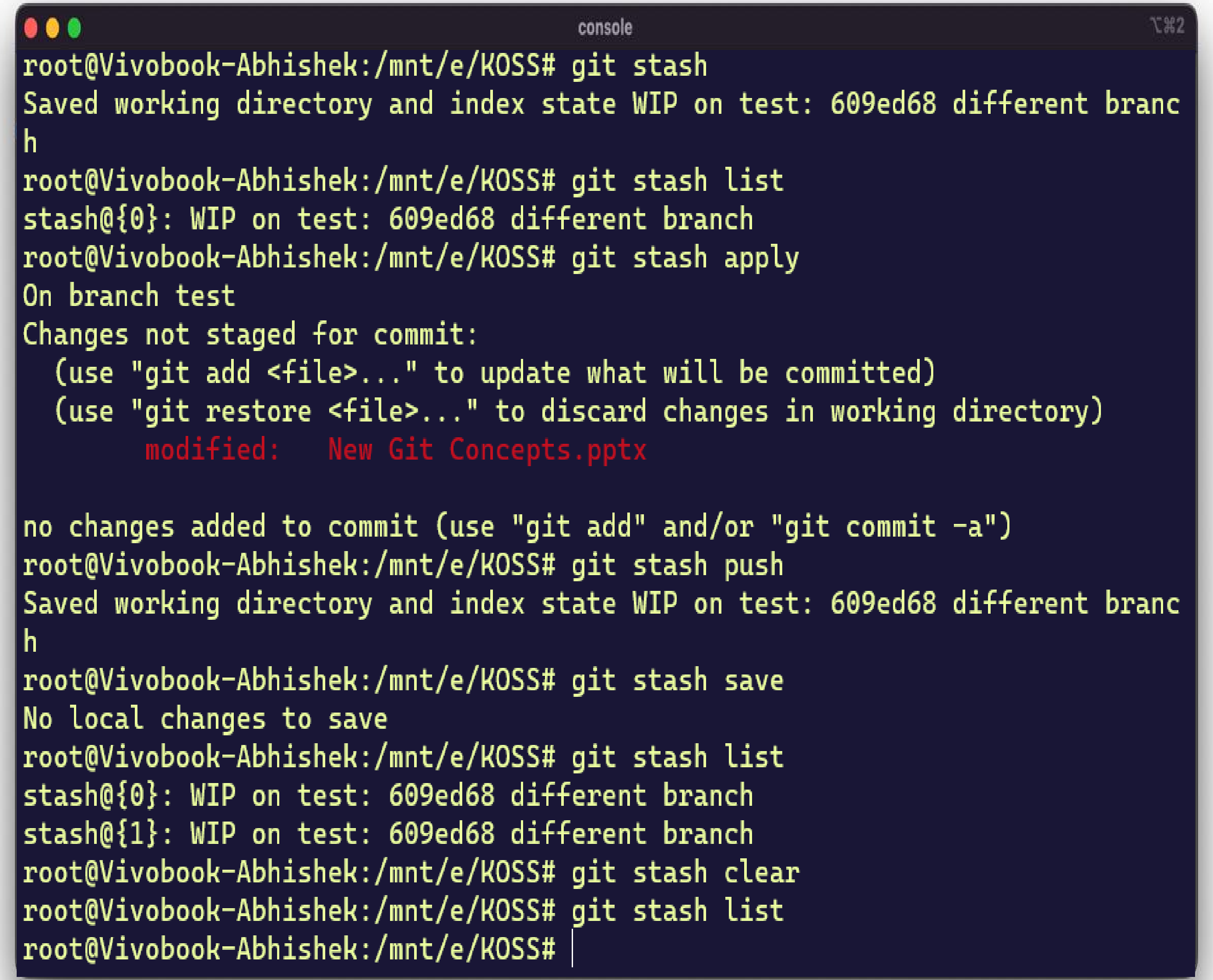


```
root@Vivobook-Abhishek:/mnt/e/KOSS# git switch test
Switched to branch 'test'
root@Vivobook-Abhishek:/mnt/e/KOSS# git branch
  main
* test
root@Vivobook-Abhishek:/mnt/e/KOSS# git add .

root@Vivobook-Abhishek:/mnt/e/KOSS# git commit
On branch test
nothing to commit, working tree clean

root@Vivobook-Abhishek:/mnt/e/KOSS# git add .
root@Vivobook-Abhishek:/mnt/e/KOSS# git commit -m "different branch"
[test 609ed68] different branch
 1 file changed, 0 insertions(+), 0 deletions(-)
root@Vivobook-Abhishek:/mnt/e/KOSS# git diff main..test
diff --git a/New Git Concepts.pptx b/New Git Concepts.pptx
index c512363..997660e 100644
Binary files a/New Git Concepts.pptx and b/New Git Concepts.pptx differ
```

Merging the two branches :



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git commit -a
[main f806d61] merging noww
 1 file changed, 0 insertions(+), 0 deletions(-)
root@Vivobook-Abhishek:/mnt/e/KOSS# git merge test
Merge made by the 'ort' strategy.
```



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash
Saved working directory and index state WIP on test: 609ed68 different branc
h
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash list
stash@{0}: WIP on test: 609ed68 different branch
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash apply
On branch test
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:   New Git Concepts.pptx

no changes added to commit (use "git add" and/or "git commit -a")
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash push
Saved working directory and index state WIP on test: 609ed68 different branc
h
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash save
No local changes to save
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash list
stash@{0}: WIP on test: 609ed68 different branch
stash@{1}: WIP on test: 609ed68 different branch
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash clear
root@Vivobook-Abhishek:/mnt/e/KOSS# git stash list
root@Vivobook-Abhishek:/mnt/e/KOSS#
```

GIT STASH

It allows one to save a copy of the code without making a commit.

$git stash list #Enable one to see the list of the stashes

$git stash apply #To reapply the stashed code

$git stash push #Creates a new stash and rolls back the state of all modified files

$git stash pop # Takes the files in a stash, places them back into the development workspace and deletes the stash from history

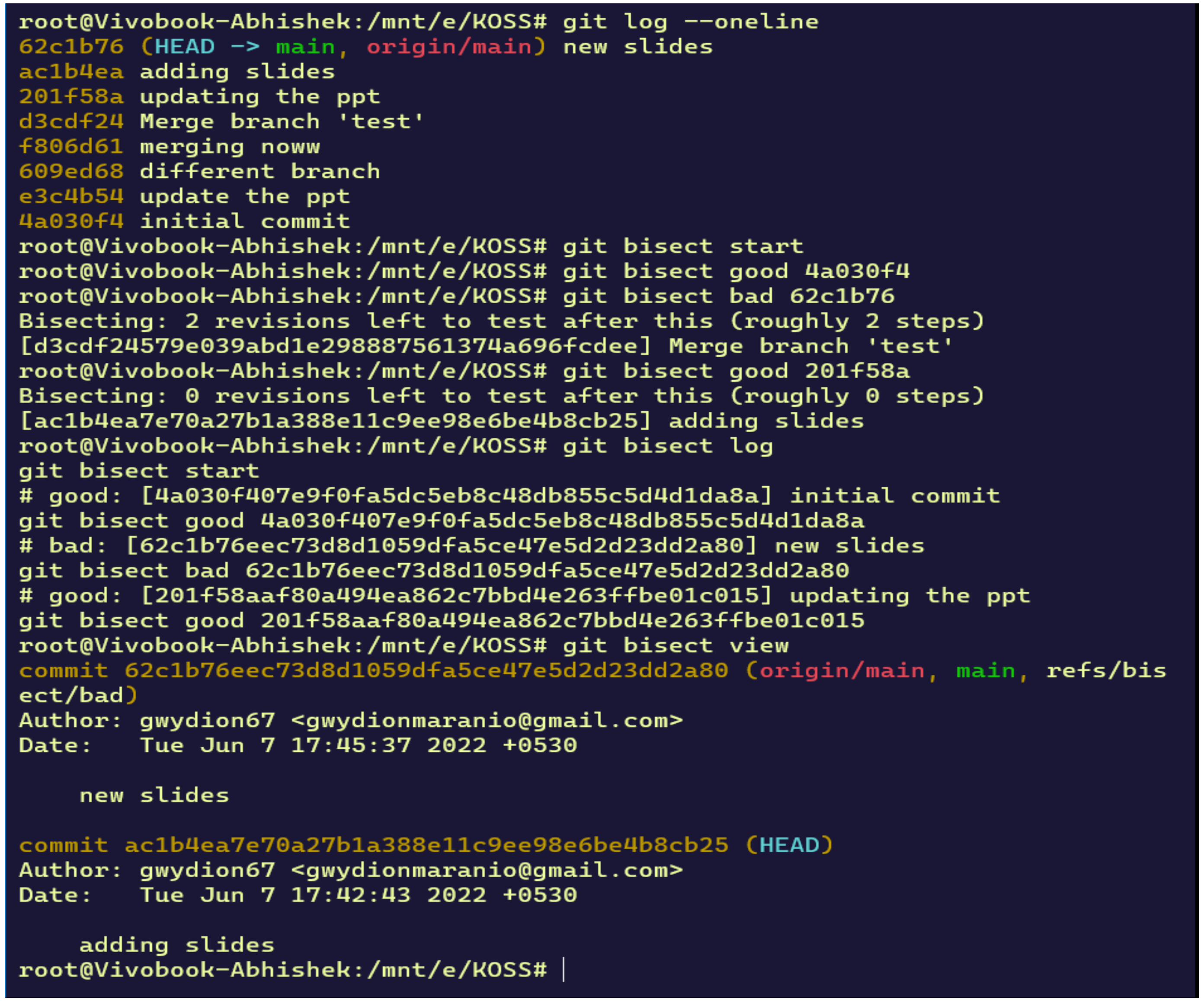
$git stash clear #Removes all entries in the git stash history

$git stash save **"message to go along with changes"**

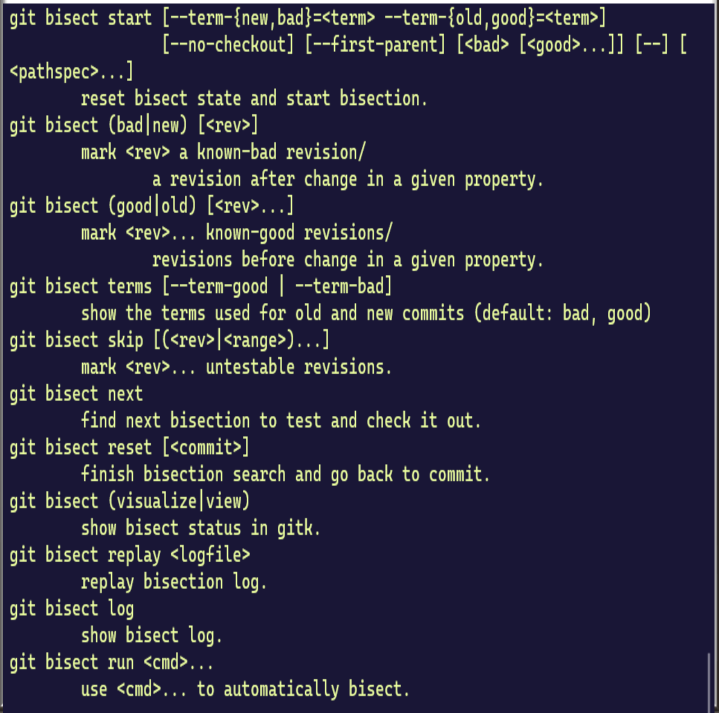
$git stash –u **#stash untracked files as well**

Git bisect

The git bisect command implements a binary search algorithm to track which commit caused the bug



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git log --oneline
62c1b76 (HEAD -> main, origin/main) new slides
ac1b4ea adding slides
201f58a updating the ppt
d3cdf24 Merge branch 'test'
f806d61 merging noww
609ed68 different branch
e3c4b54 update the ppt
4a030f4 initial commit
root@Vivobook-Abhishek:/mnt/e/KOSS# git bisect start
root@Vivobook-Abhishek:/mnt/e/KOSS# git bisect good 4a030f4
root@Vivobook-Abhishek:/mnt/e/KOSS# git bisect bad 62c1b76
Bisecting: 2 revisions left to test after this (roughly 2 steps)
[d3cdf24579e039abd1e29888756137 4a696fcdee] Merge branch 'test'
root@Vivobook-Abhishek:/mnt/e/KOSS# git bisect good 201f58a
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[ac1b4ea7e70a27b1a388e11c9ee98e6be4b8cb25] adding slides
root@Vivobook-Abhishek:/mnt/e/KOSS# git bisect log
git bisect start
# good: [4a030f407e9f0fa5dc5eb8c48db855c5d4d1da8a] initial commit
git bisect good 4a030f407e9f0fa5dc5eb8c48db855c5d4d1da8a
# bad: [62c1b76eec73d8d1059dfa5ce47e5d2d23dd2a80] new slides
git bisect bad 62c1b76eec73d8d1059dfa5ce47e5d2d23dd2a80
# good: [201f58aaf80a494ea862c7bbd4e263ffbe01c015] updating the ppt
git bisect good 201f58aaf80a494ea862c7bbd4e263ffbe01c015
root@Vivobook-Abhishek:/mnt/e/KOSS# git bisect view
commit 62c1b76eec73d8d1059dfa5ce47e5d2d23dd2a80 (origin/main, main, refs/bis
ect/bad)
Author: gwydion67 <gwydionmaranio@gmail.com>
Date:   Tue Jun 7 17:45:37 2022 +0530

    new slides

commit ac1b4ea7e70a27b1a388e11c9ee98e6be4b8cb25 (HEAD)
Author: gwydion67 <gwydionmaranio@gmail.com>
Date:   Tue Jun 7 17:42:43 2022 +0530

    adding slides
root@Vivobook-Abhishek:/mnt/e/KOSS# |
```



```
git bisect start [--term-{new,bad}=<term> --term-{old,good}=<term>]
                 [--no-checkout] [--first-parent] [<bad> [<good>...]] [--] [
<pathspec>...]
        reset bisect state and start bisection.
git bisect (bad|new) [<rev>]
        mark <rev> a known-bad revision/
                a revision after change in a given property.
git bisect (good|old) [<rev>...]
        mark <rev>... known-good revisions/
                revisions before change in a given property.
git bisect terms [--term-good | --term-bad]
        show the terms used for old and new commits (default: bad, good)
git bisect skip [(<rev>|<range>)...]
        mark <rev>... untestable revisions.
git bisect next
        find next bisection to test and check it out.
git bisect reset [<commit>]
        finish bisection search and go back to commit.
git bisect (visualize|view)
        show bisect status in gitk.
git bisect replay <logfile>
        replay bisection log.
git bisect log
        show bisect log.
git bisect run <cmd>...
        use <cmd>... to automatically bisect.
```

Assume the following history exists and the current branch is "topic":

```
          A---B---C topic
         /
    D---E---F---G main
```

From this point, the result of either of the following commands:

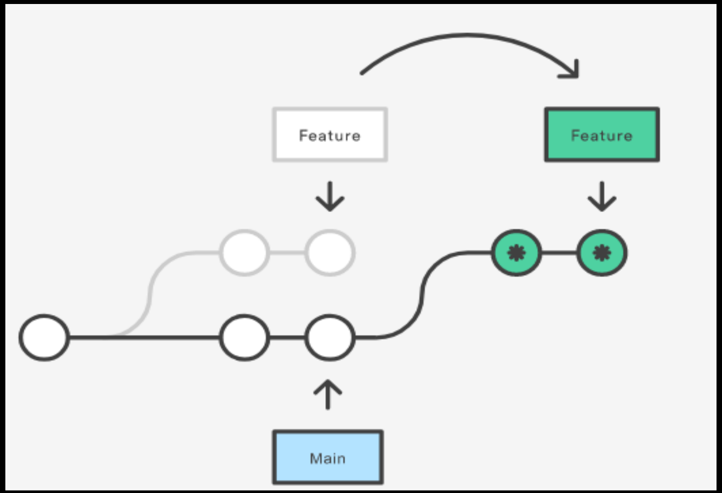
```
git rebase main
git rebase main topic
```

would be:

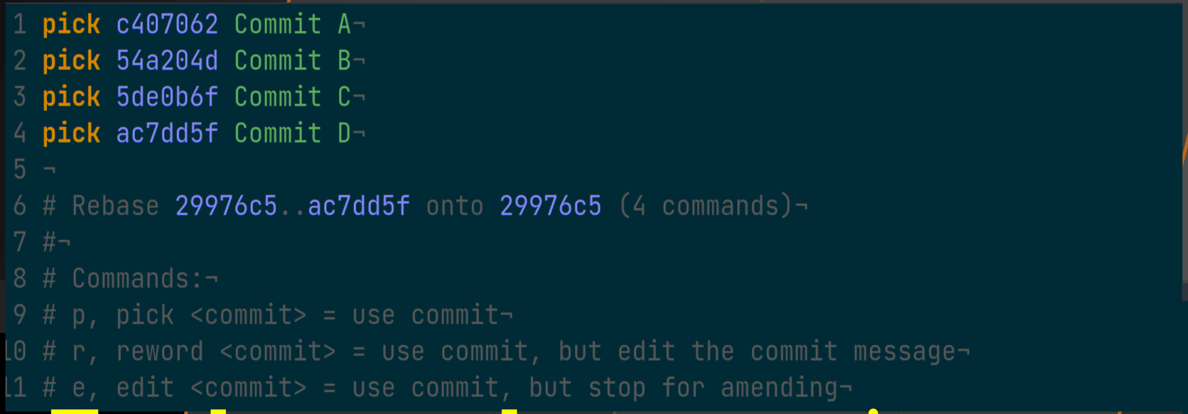
```
              A'--B'--C' topic
             /
    D---E---F---G main
```

GIT REBASE

Rebasing is the process of moving or combining a sequence of commits to a new base commit. rebasing is changing the base of your branch from one commit to another making it appear as if you'd created your branch from a different commit.

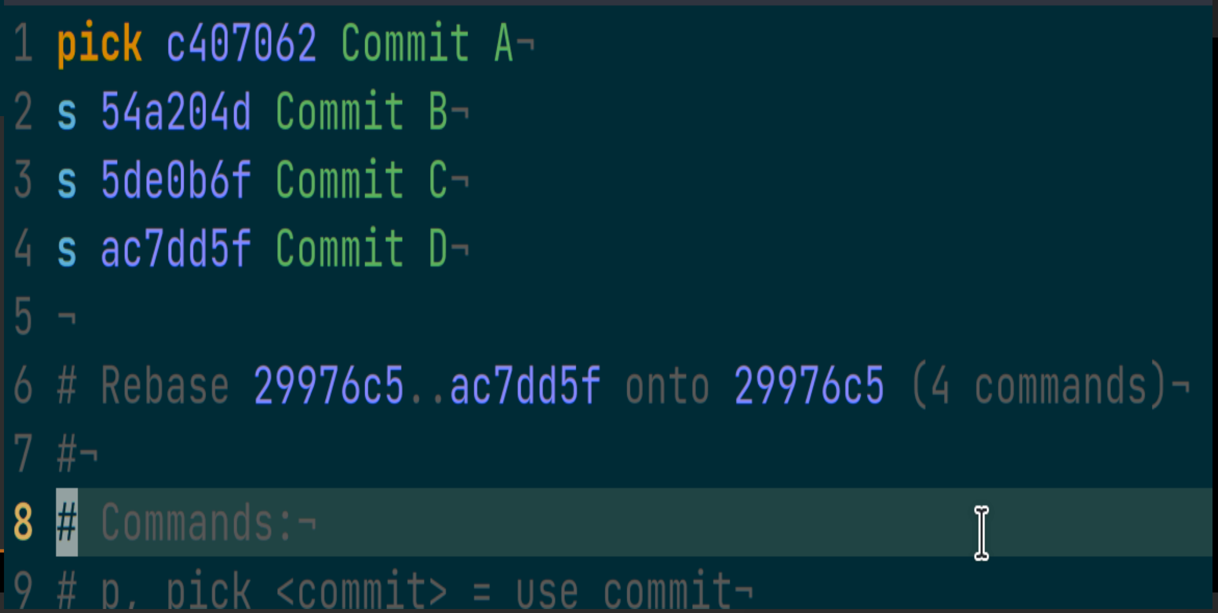


Git rebase can also be used to squash two or more commits into one as well to rename a commit



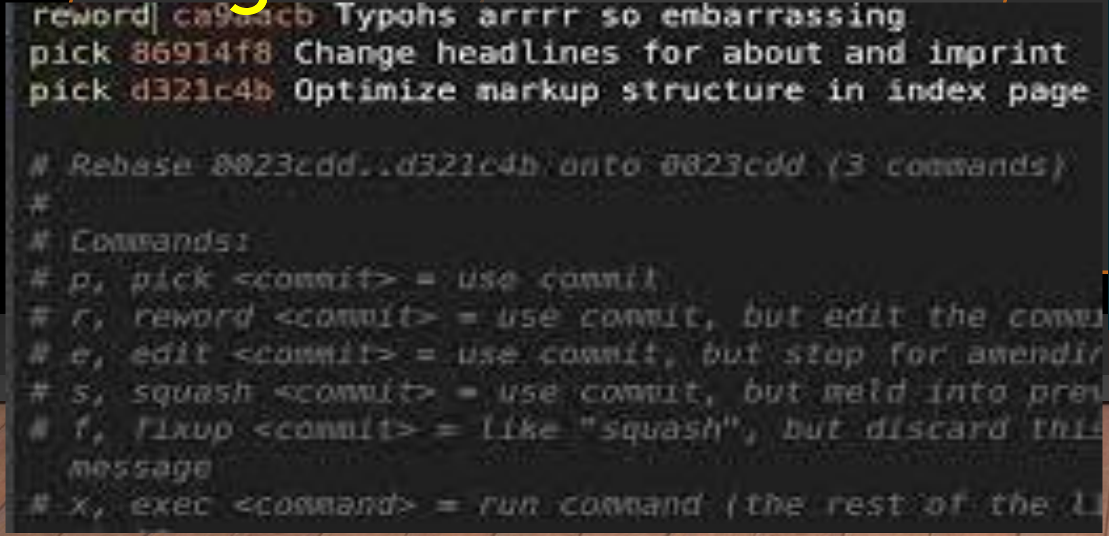
```
pick c407062 Commit A¬
pick 54a204d Commit B¬
pick 5de0b6f Commit C¬
pick ac7dd5f Commit D¬
¬
# Rebase 29976c5..ac7dd5f onto 29976c5 (4 commands)¬
#¬
# Commands:¬
# p, pick <commit> = use commit¬
# r, reword <commit> = use commit, but edit the commit message¬
# e, edit <commit> = use commit, but stop for amending¬
```

This will squash the commits B,C,D into a single commit i.e. A

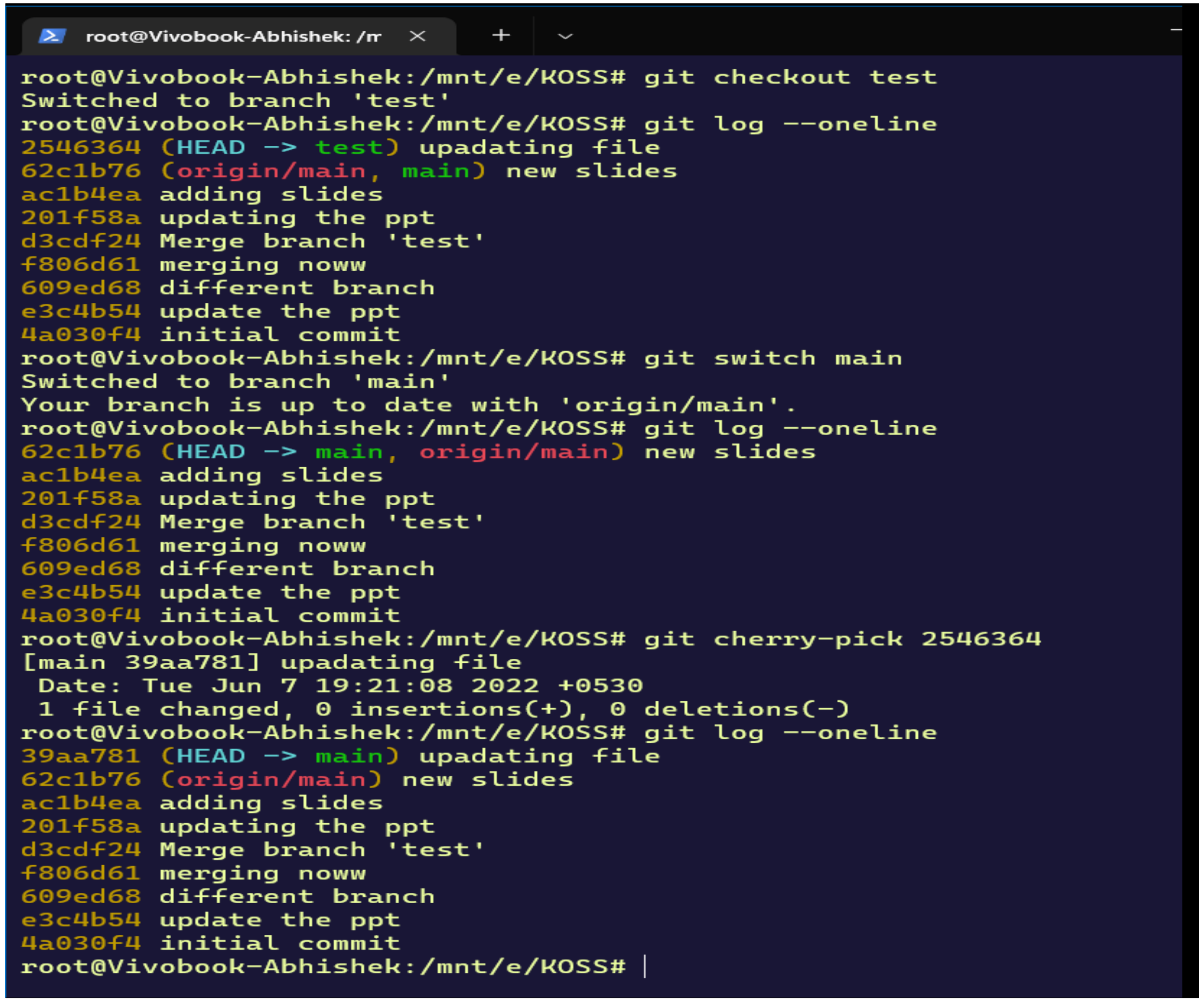


```
pick c407062 Commit A¬
s 54a204d Commit B¬
s 5de0b6f Commit C¬
s ac7dd5f Commit D¬
¬
# Rebase 29976c5..ac7dd5f onto 29976c5 (4 commands)¬
#¬
# Commands:¬
# p, pick <commit> = use commit¬
```

To change the commit message we use reword



```
reword ca9[illegible]cb Typohs arrrr so embarrassing
pick 86914f8 Change headlines for about and imprint
pick d321c4b Optimize markup structure in index page

# Rebase 0023cdd..d321c4b onto 0023cdd (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commi
# e, edit <commit> = use commit, but stop for amendin
# s, squash <commit> = use commit, but meld into prev
# f, fixup <commit> = like "squash", but discard this
  message
# x, exec <command> = run command (the rest of the li
```



```
root@Vivobook-Abhishek:/mnt/e/KOSS# git checkout test
Switched to branch 'test'
root@Vivobook-Abhishek:/mnt/e/KOSS# git log --oneline
2546364 (HEAD -> test) upadating file
62c1b76 (origin/main, main) new slides
ac1b4ea adding slides
201f58a updating the ppt
d3cdf24 Merge branch 'test'
f806d61 merging noww
609ed68 different branch
e3c4b54 update the ppt
4a030f4 initial commit
root@Vivobook-Abhishek:/mnt/e/KOSS# git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
root@Vivobook-Abhishek:/mnt/e/KOSS# git log --oneline
62c1b76 (HEAD -> main, origin/main) new slides
ac1b4ea adding slides
201f58a updating the ppt
d3cdf24 Merge branch 'test'
f806d61 merging noww
609ed68 different branch
e3c4b54 update the ppt
4a030f4 initial commit
root@Vivobook-Abhishek:/mnt/e/KOSS# git cherry-pick 2546364
[main 39aa781] upadating file
 Date: Tue Jun 7 19:21:08 2022 +0530
 1 file changed, 0 insertions(+), 0 deletions(-)
root@Vivobook-Abhishek:/mnt/e/KOSS# git log --oneline
39aa781 (HEAD -> main) upadating file
62c1b76 (origin/main) new slides
ac1b4ea adding slides
201f58a updating the ppt
d3cdf24 Merge branch 'test'
f806d61 merging noww
609ed68 different branch
e3c4b54 update the ppt
4a030f4 initial commit
root@Vivobook-Abhishek:/mnt/e/KOSS# |
```

GIT CHERRY-PICK

Git cherry-pick is used to merge specific commits from one branch to other

git cherry-pick <commit-sha>

This merges the particular commit to the current branch

**cherry-pick is rarely used as it can easily arise merge conflicts

Arigatogozaimashita!!

For further contacts:

- Telegram--@maranio67
- GitHub--gwydion67

Sources:

- ➔dev.to
- ➔toptal
- ➔Atlassian
- ➔git