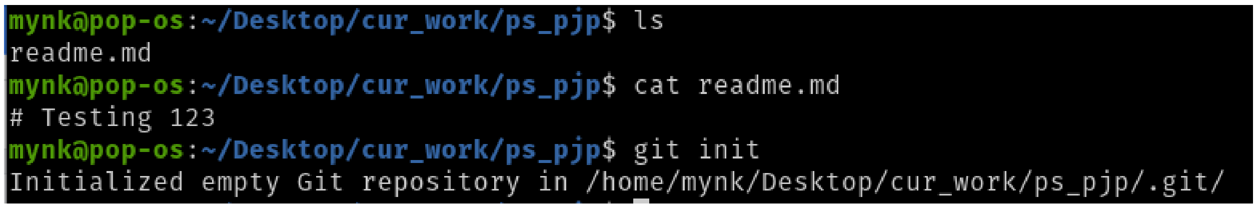
Assignment 1 (GIT / GITHUB)

Mayank Gautam

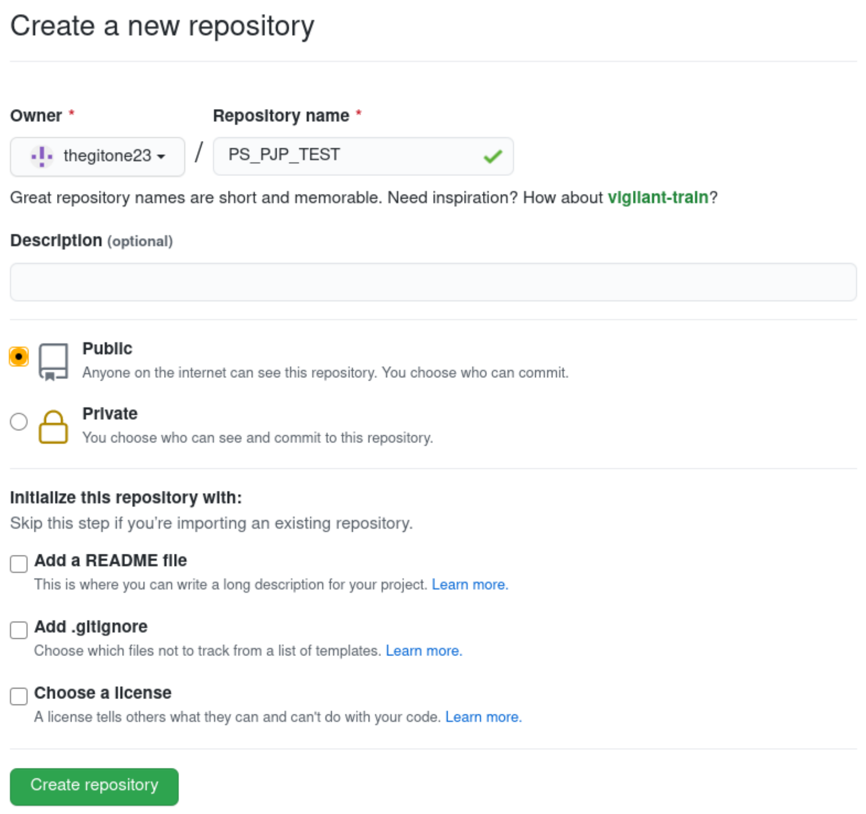
A. Basic CLI usage

1. Creating a local repo



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ ls
readme.md
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ cat readme.md
# Testing 123
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git init
Initialized empty Git repository in /home/mynk/Desktop/cur_work/ps_pjp/.git/
```

2. Setup a remote repo



Create a new repository

Owner * thegitone23 / Repository name * PS_PJP_TEST

Great repository names are short and memorable. Need inspiration? How about vigilant-train?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

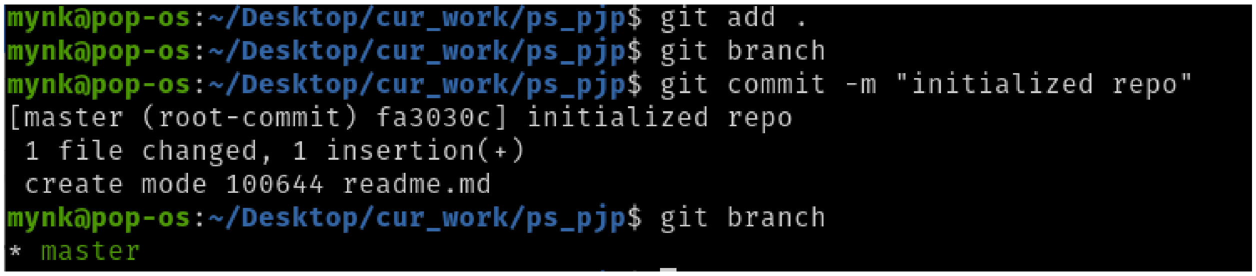
Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

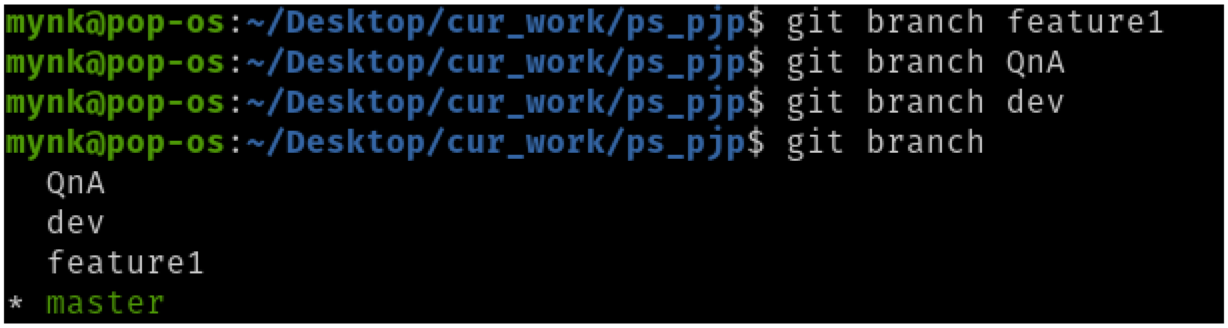
☐ Add .gitignore
Choose which files not to track from a list of templates. Learn more.

☐ Choose a license
A license tells others what they can and can't do with your code. Learn more.

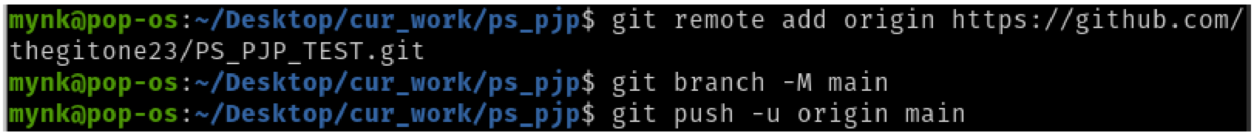
3. Create local branches



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git add .
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git commit -m "initialized repo"
[master (root-commit) fa3030c] initialized repo
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch
* master
```

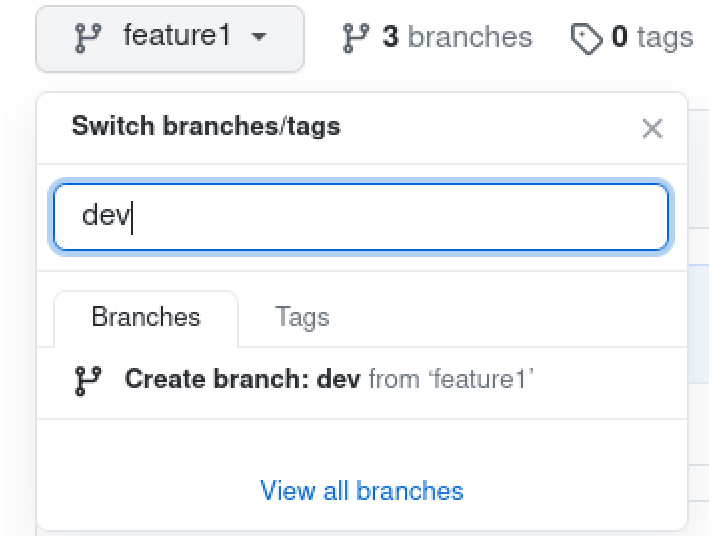


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch feature1
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch QnA
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch dev
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch
  QnA
  dev
  feature1
* master
```

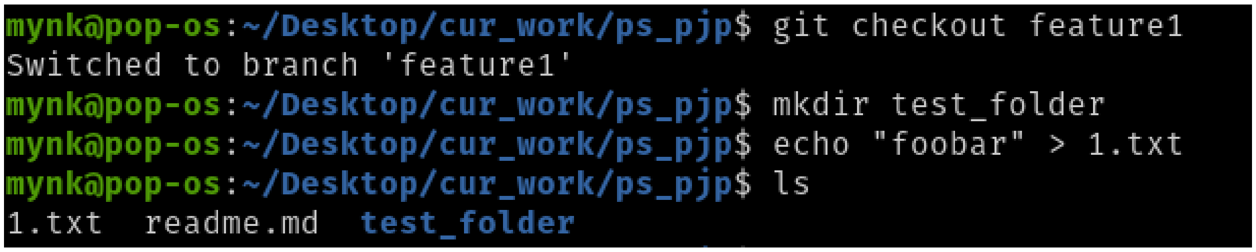


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git remote add origin https://github.com/
thegitone23/PS_PJP_TEST.git
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch -M main
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git push -u origin main
```

4. Create remote branches

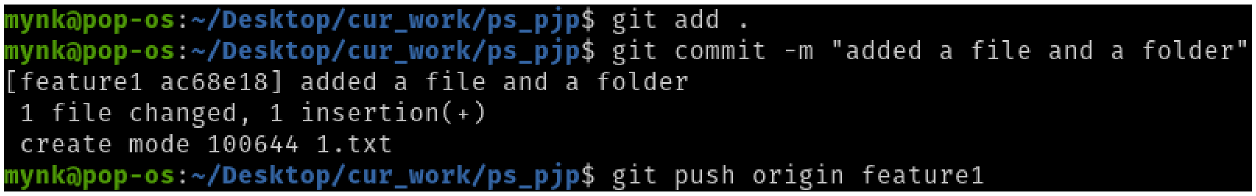


5. Add files / folders



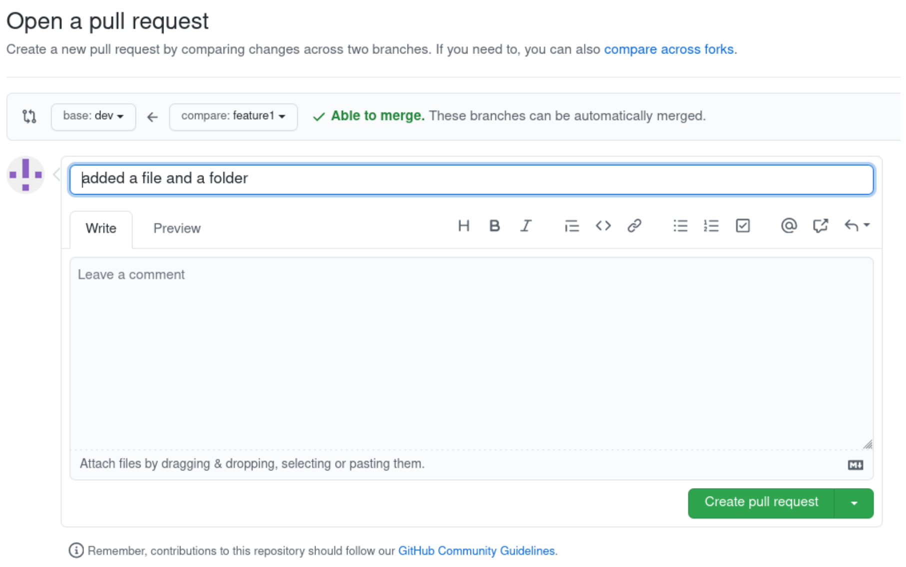
```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout feature1
Switched to branch 'feature1'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ mkdir test_folder
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ echo "foobar" > 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ ls
1.txt  readme.md  test_folder
```

6. Check-in stage commit and push files to feature branch



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git add .
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git commit -m "added a file and a folder"
[feature1 ac68e18] added a file and a folder
 1 file changed, 1 insertion(+)
 create mode 100644 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git push origin feature1
```

7. Promote code from feature branch to dev branch using pull requests



Open a pull request

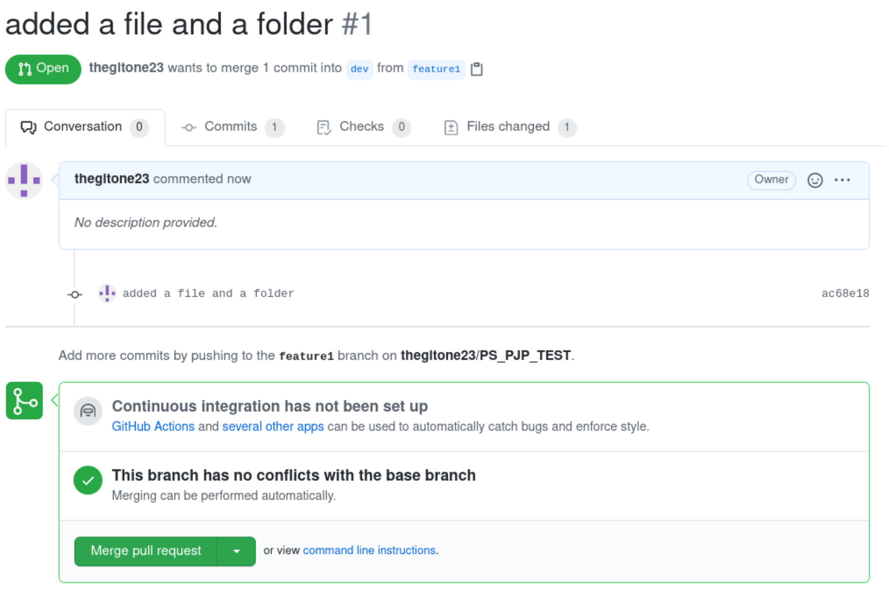
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: dev ← compare: feature1 ✓ **Able to merge.** These branches can be automatically merged.

added a file and a folder

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Create pull request

Remember, contributions to this repository should follow our GitHub Community Guidelines.


added a file and a folder #1

Open thegItone23 wants to merge 1 commit into `dev` from `feature1`

Conversation 0 · Commits 1 · Checks 0 · Files changed 1

thegItone23 commented now · Owner

No description provided.

added a file and a folder · ac68e18

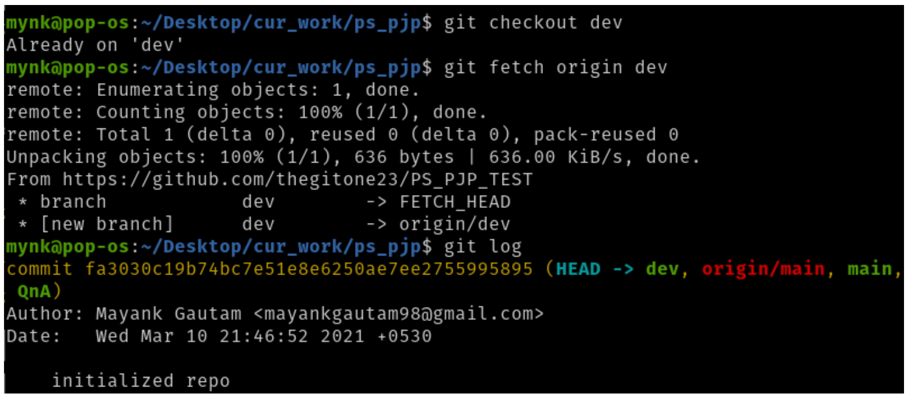
Add more commits by pushing to the **feature1** branch on **thegItone23/PS_PJP_TEST**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

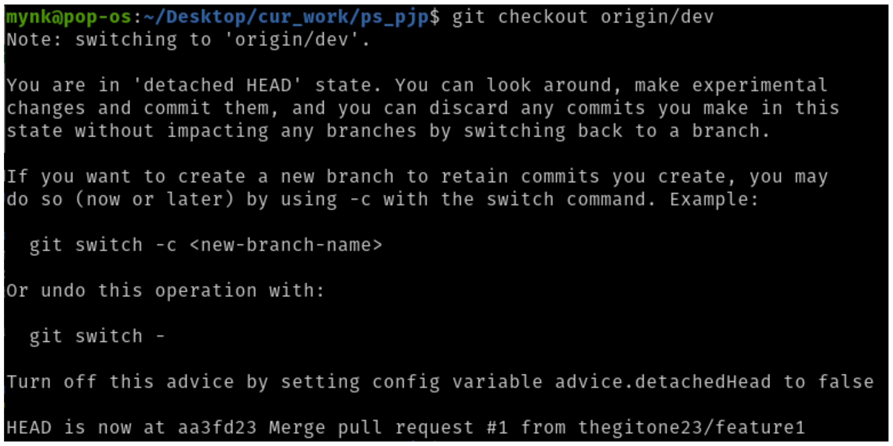
This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request · or view command line instructions.

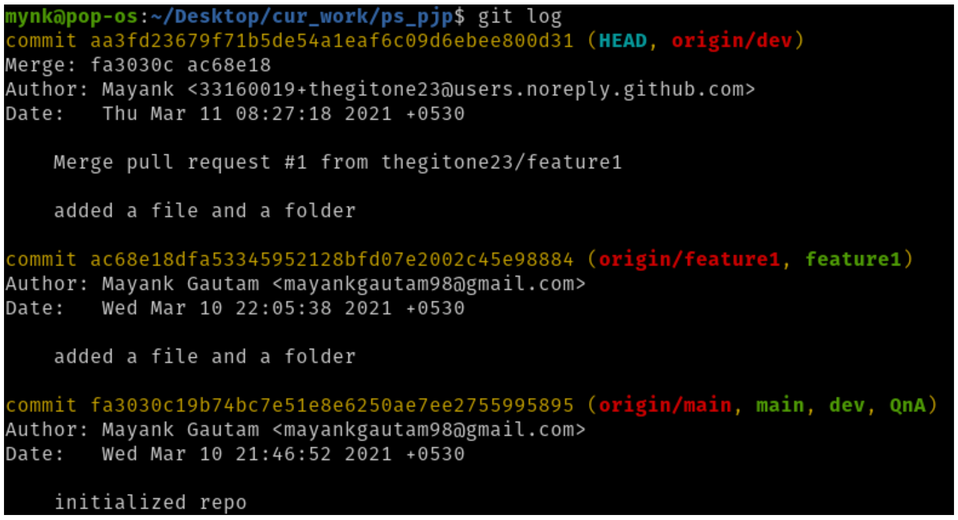
8. Checkout code from remote (dev) branch to local (dev) branch



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout dev
Already on 'dev'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git fetch origin dev
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 636 bytes | 636.00 KiB/s, done.
From https://github.com/thegitone23/PS_PJP_TEST
 * branch            dev        -> FETCH_HEAD
 * [new branch]      dev        -> origin/dev
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git log
commit fa3030c19b74bc7e51e8e6250ae7ee2755995895 (HEAD -> dev, origin/main, main,
 QnA)
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Wed Mar 10 21:46:52 2021 +0530

    initialized repo
```



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout origin/dev
Note: switching to 'origin/dev'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  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 aa3fd23 Merge pull request #1 from thegitone23/feature1
```

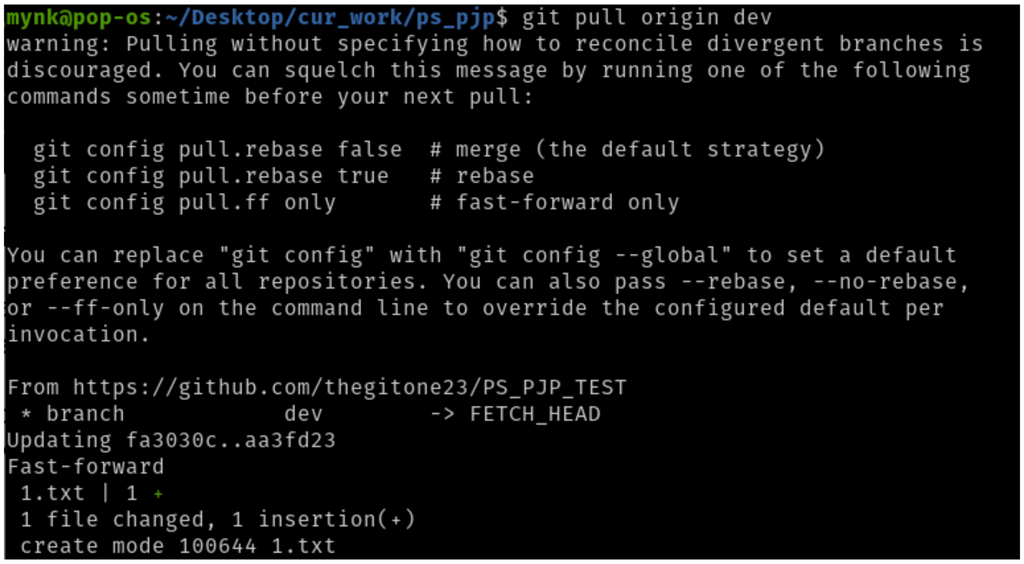


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git log
commit aa3fd23679f71b5de54a1eaf6c09d6ebee800d31 (HEAD, origin/dev)
Merge: fa3030c ac68e18
Author: Mayank <33160019+thegitone23@users.noreply.github.com>
Date:   Thu Mar 11 08:27:18 2021 +0530

    Merge pull request #1 from thegitone23/feature1

    added a file and a folder

commit ac68e18dfa53345952128bfd07e2002c45e98884 (origin/feature1, feature1)
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Wed Mar 10 22:05:38 2021 +0530

    added a file and a folder

commit fa3030c19b74bc7e51e8e6250ae7ee2755995895 (origin/main, main, dev, QnA)
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Wed Mar 10 21:46:52 2021 +0530

    initialized repo
```

9. Explore the difference between checkout and pull

checkout on a remote branch lets the user explore the branch on the local machine. The changes are not merged in the local repo.

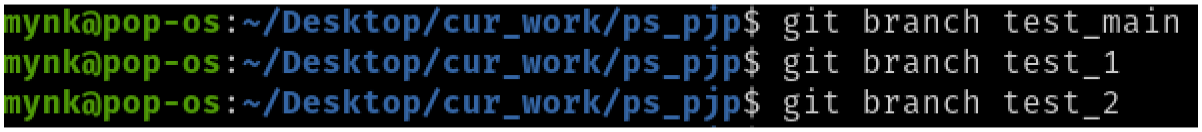
pull fetches the changes from the remote repo and also merges them locally, effectively changing the local repo



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git pull origin dev
warning: Pulling without specifying how to reconcile divergent branches is
discouraged. You can squelch this message by running one of the following
commands sometime before your next pull:

  git config pull.rebase false  # merge (the default strategy)
  git config pull.rebase true   # rebase
  git config pull.ff only       # fast-forward only

You can replace "git config" with "git config --global" to set a default
preference for all repositories. You can also pass --rebase, --no-rebase,
or --ff-only on the command line to override the configured default per
invocation.

From https://github.com/thegitone23/PS_PJP_TEST
 * branch            dev        -> FETCH_HEAD
Updating fa3030c..aa3fd23
Fast-forward
 1.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 1.txt
```

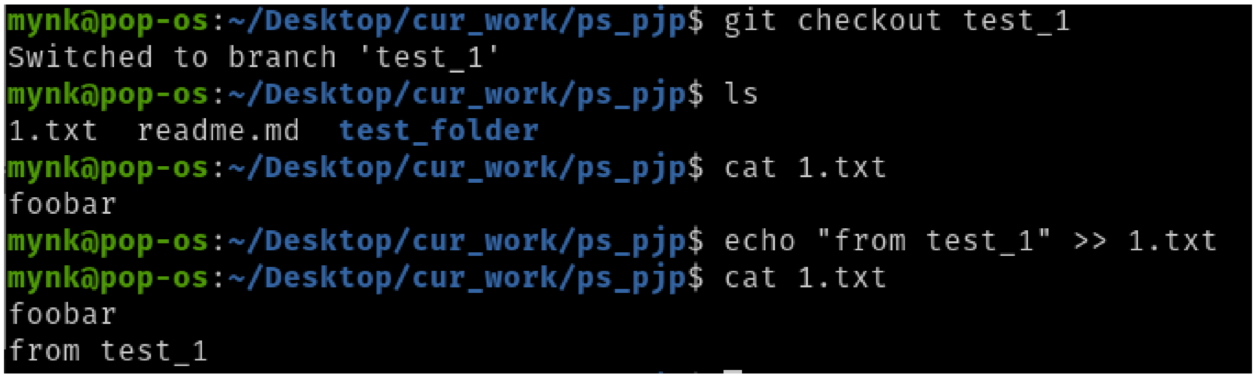
10. Resolve a merge conflict where changes have been made in the same file

Let’s create three new temporary branches

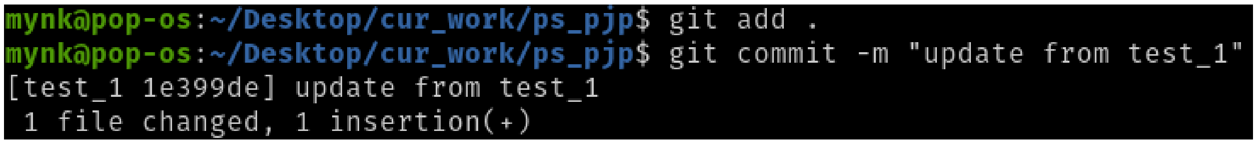


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch test_main
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch test_1
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git branch test_2
```

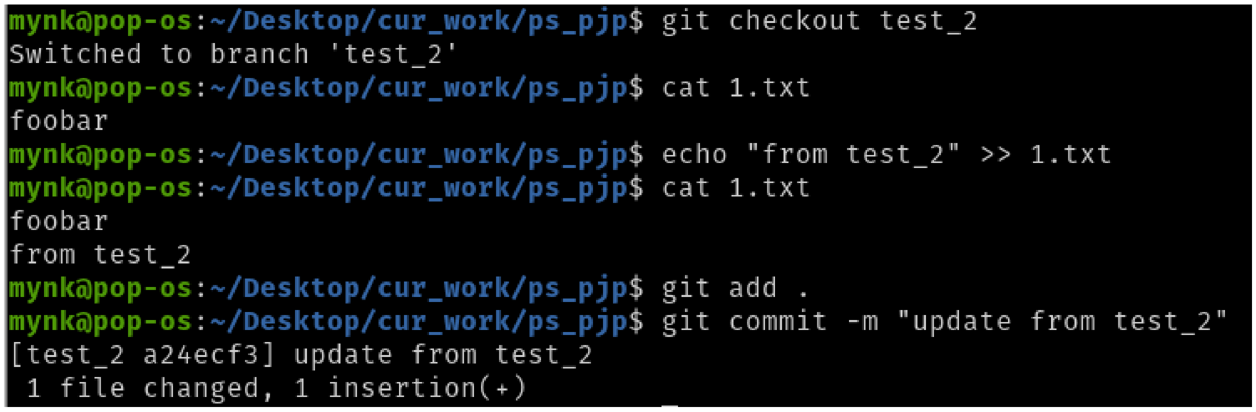
Now lets edit and commit a common file in test_1 and test_2 respectively



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout test_1
Switched to branch 'test_1'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ ls
1.txt  readme.md  test_folder
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ cat 1.txt
foobar
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ echo "from test_1" >> 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ cat 1.txt
foobar
from test_1
```

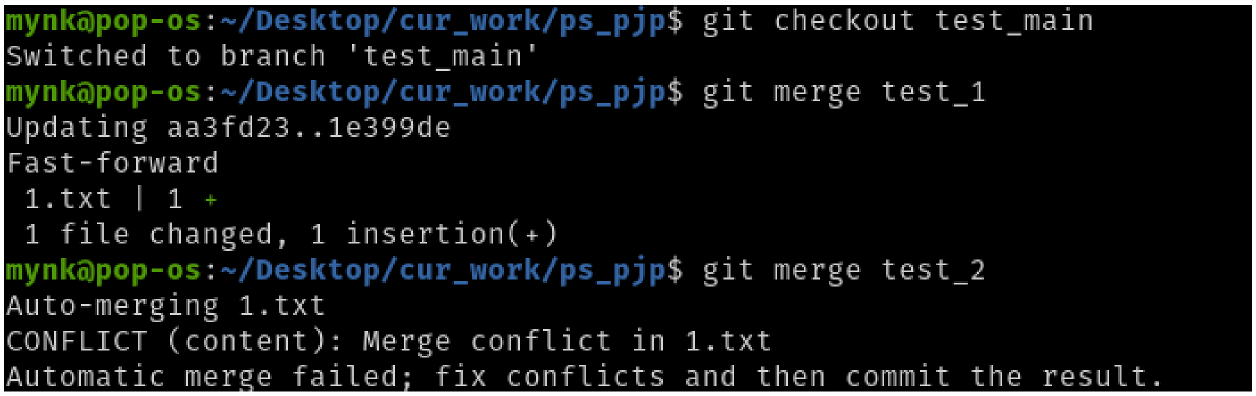


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git add .
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git commit -m "update from test_1"
[test_1 1e399de] update from test_1
 1 file changed, 1 insertion(+)
```

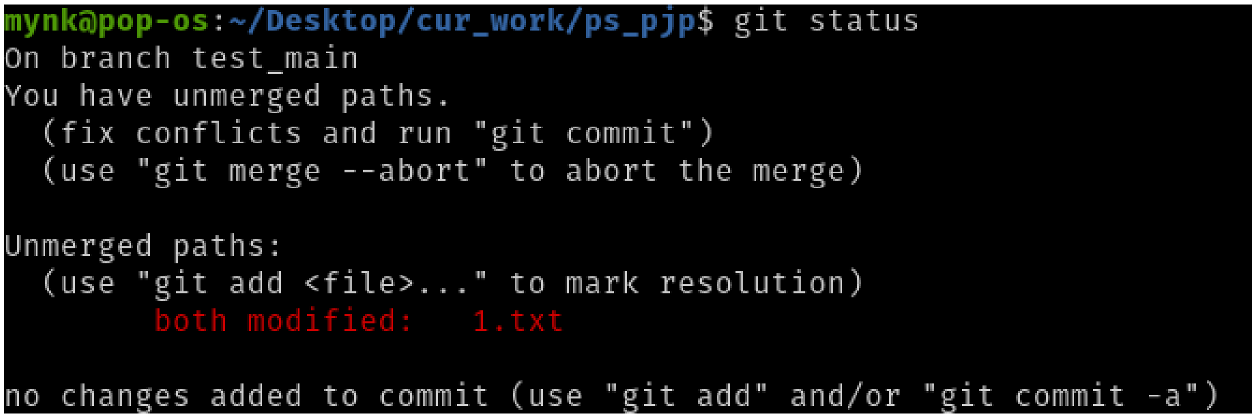


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout test_2
Switched to branch 'test_2'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ cat 1.txt
foobar
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ echo "from test_2" >> 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ cat 1.txt
foobar
from test_2
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git add .
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git commit -m "update from test_2"
[test_2 a24ecf3] update from test_2
 1 file changed, 1 insertion(+)
```

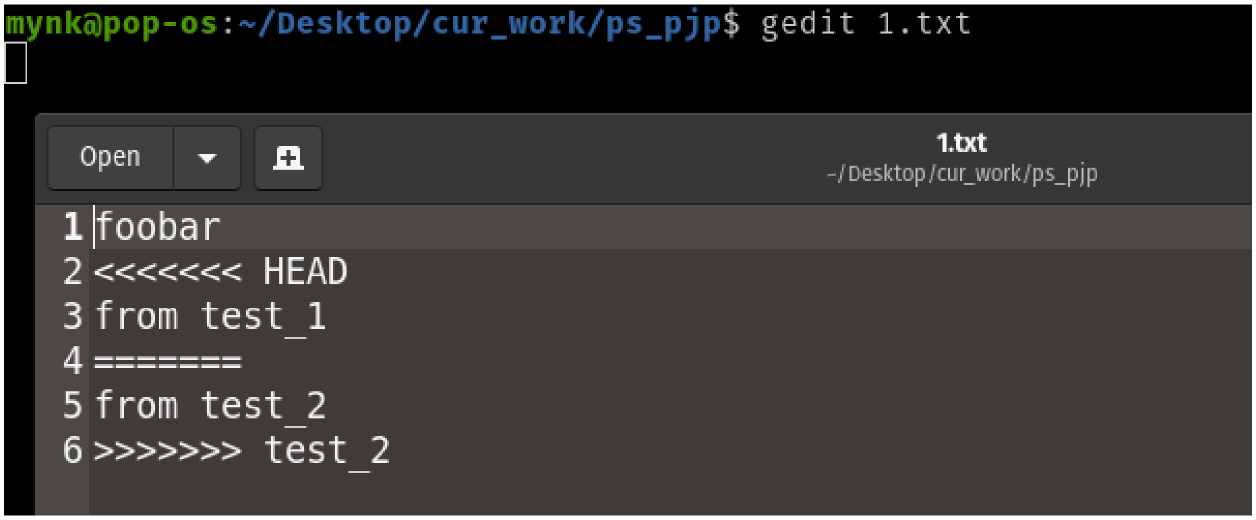
Now lets try merging test_1 and test_2 in test_main



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout test_main
Switched to branch 'test_main'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git merge test_1
Updating aa3fd23..1e399de
Fast-forward
 1.txt | 1 +
 1 file changed, 1 insertion(+)
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git merge test_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Automatic merge failed; fix conflicts and then commit the result.
```



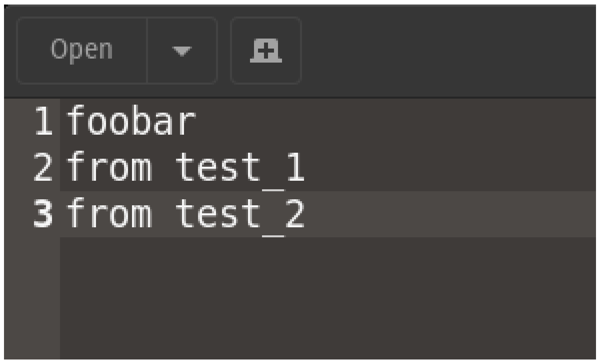
```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   1.txt

no changes added to commit (use "git add" and/or "git commit -a")
```



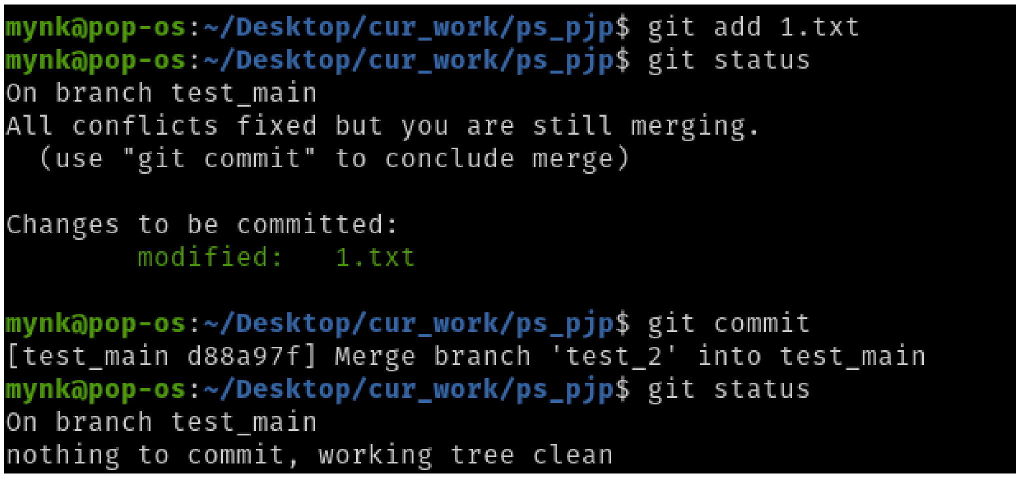
```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ gedit 1.txt
```

```
foobar
<<<<<<< HEAD
from test_1
=======
from test_2
>>>>>>> test_2
```

Now let’s manually edit 1.txt and finish the merge

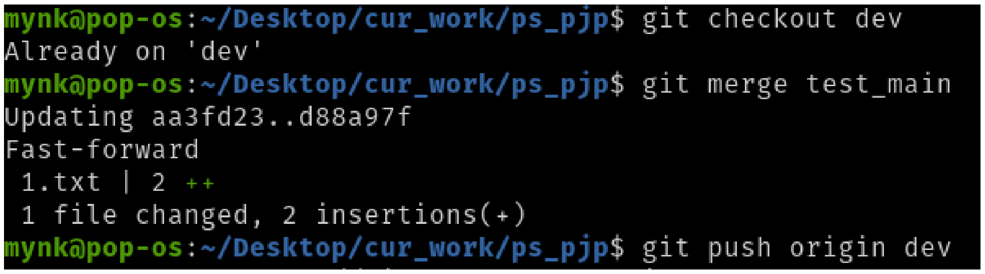


```
foobar
from test_1
from test_2
```

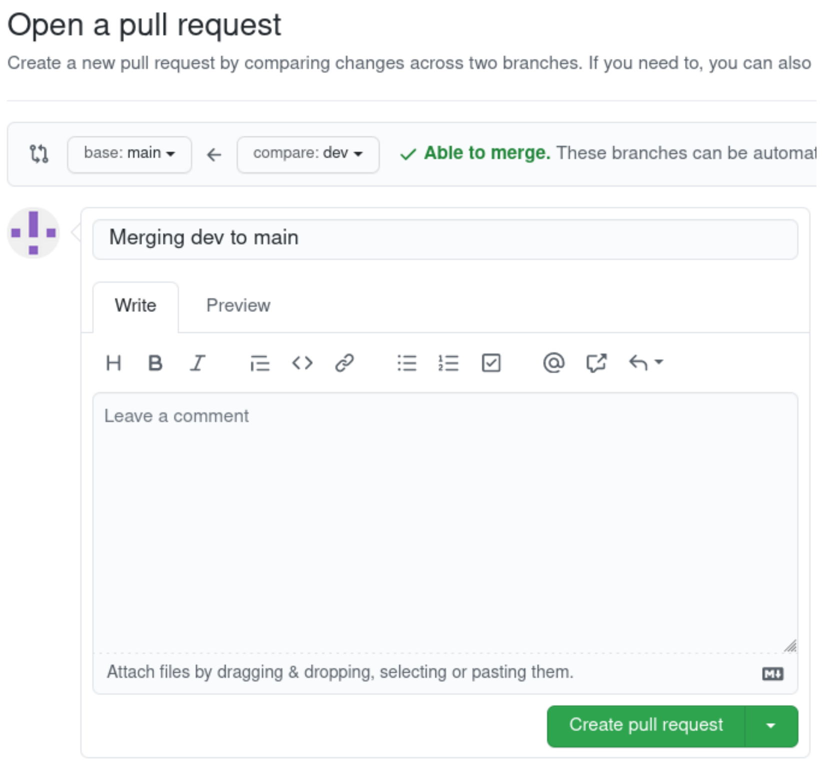


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git add 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   1.txt

mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git commit
[test_main d88a97f] Merge branch 'test_2' into test_main
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
nothing to commit, working tree clean
```

11. Syncing the latest changes



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout dev
Already on 'dev'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git merge test_main
Updating aa3fd23..d88a97f
Fast-forward
 1.txt | 2 ++
 1 file changed, 2 insertions(+)
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git push origin dev
```

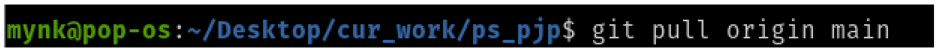


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also

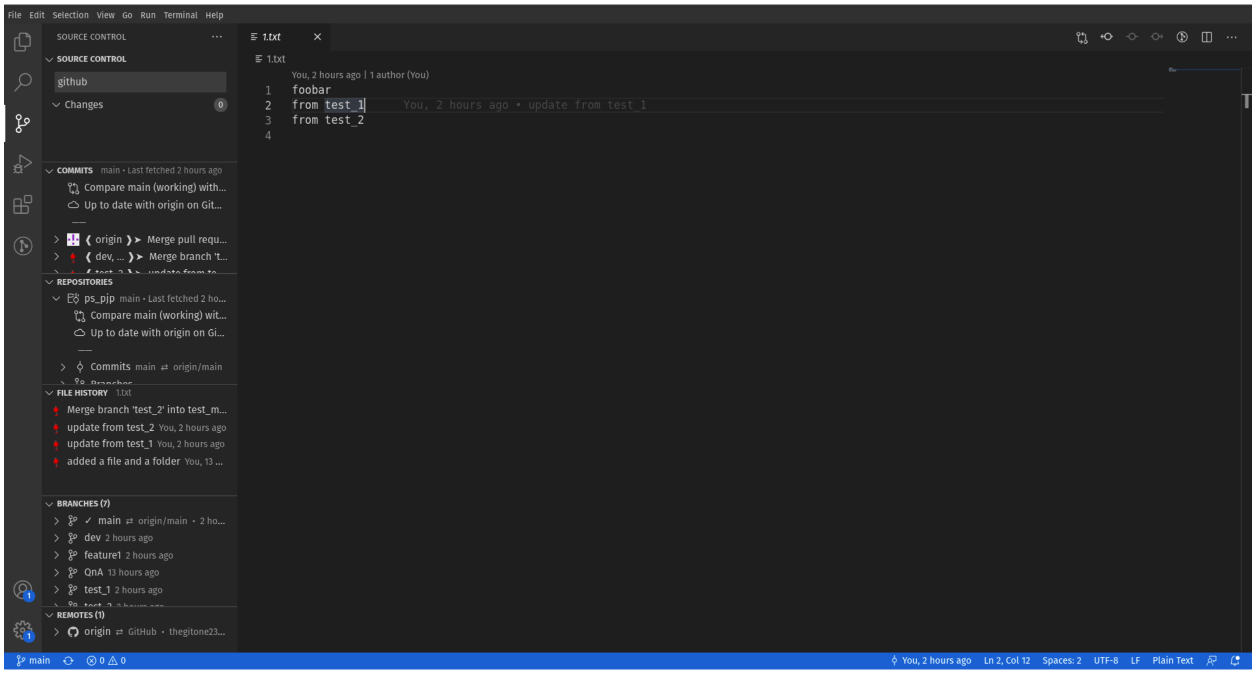
Able to merge. These branches can be automat

Merging dev to main



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git pull origin main
```

B. Using GUI interfaces with git


File Edit Selection View Go Run Terminal Help
SOURCE CONTROL
SOURCE CONTROL
github
Changes 0
COMMITS main • Last fetched 2 hours ago
Compare main (working) with...
Up to date with origin on Git...
❬ origin ❭➤ Merge pull requ...
❬ dev, ... ❭➤ Merge branch 't...
REPOSITORIES
ps_pjp main • Last fetched 2 ho...
Compare main (working) wit...
Up to date with origin on Gi...
Commits main ⇄ origin/main
FILE HISTORY 1.txt
BRANCHES (7)
REMOTES (1)
origin ⇄ GitHub • thegitone23...
1.txt
main
0 0
You, 2 hours ago
Ln 2, Col 12
Spaces: 2
UTF-8
LF
Plain Text

Merge branch 'test_2' into test_m...
update from test_2 You, 2 hours ago
update from test_1 You, 2 hours ago
added a file and a folder You, 13 ...

✓ main ⇄ origin/main • 2 ho...
dev 2 hours ago
feature1 2 hours ago
QnA 13 hours ago
test_1 2 hours ago

You, 2 hours ago | 1 author (You)

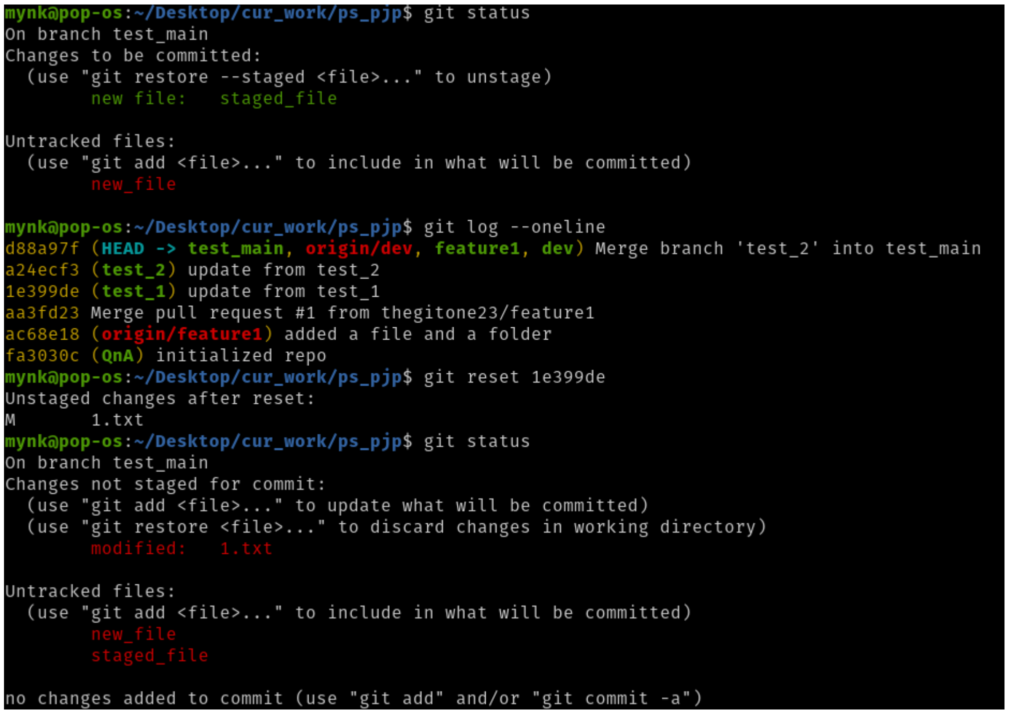
```
foobar
from test_1      You, 2 hours ago • update from test_1
from test_2
```

VS Code along with any suitable extension (git lens in this case) can provide an intuitive GUI to perform all the basic git operations like branching, forking, checking in/out, merging, pushing / pulling code and many more.

C. Other common day to day, git tasks

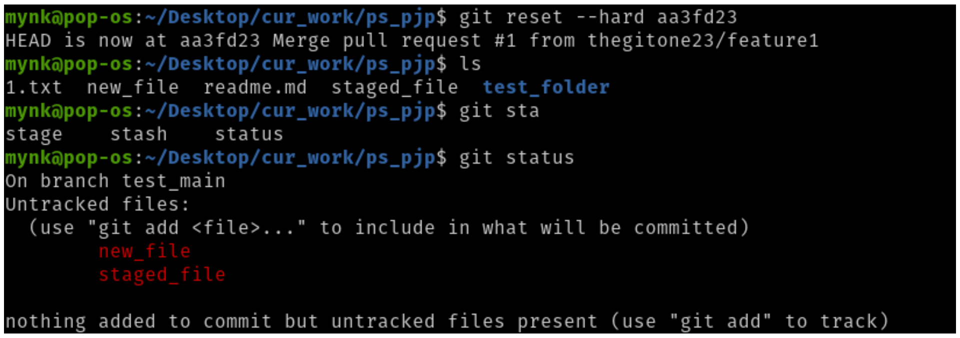
1. reset / revert files to previous state and ignore the local changes (soft reset and hard reset)

reset without --soft (default behaviour) keeps all the changes and staged / untracked files all as unstaged changes



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   staged_file

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new_file

mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git log --oneline
d88a97f (HEAD -> test_main, origin/dev, feature1, dev) Merge branch 'test_2' into test_main
a24ecf3 (test_2) update from test_2
1e399de (test_1) update from test_1
aa3fd23 Merge pull request #1 from thegitone23/feature1
ac68e18 (origin/feature1) added a file and a folder
fa3030c (QnA) initialized repo
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git reset 1e399de
Unstaged changes after reset:
M       1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
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:   1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new_file
        staged_file

no changes added to commit (use "git add" and/or "git commit -a")
```

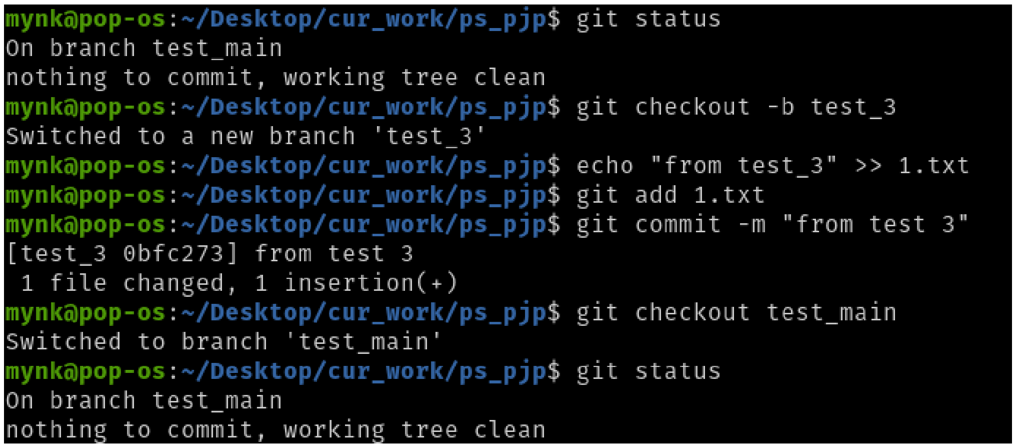
with --hard all uncommitted changes are **lost**, while the staged/untracked files and folders all exist as untracked



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git reset --hard aa3fd23
HEAD is now at aa3fd23 Merge pull request #1 from thegitone23/feature1
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ ls
1.txt  new_file  readme.md  staged_file  test_folder
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git sta
stage    stash    status
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new_file
        staged_file

nothing added to commit but untracked files present (use "git add" to track)
```

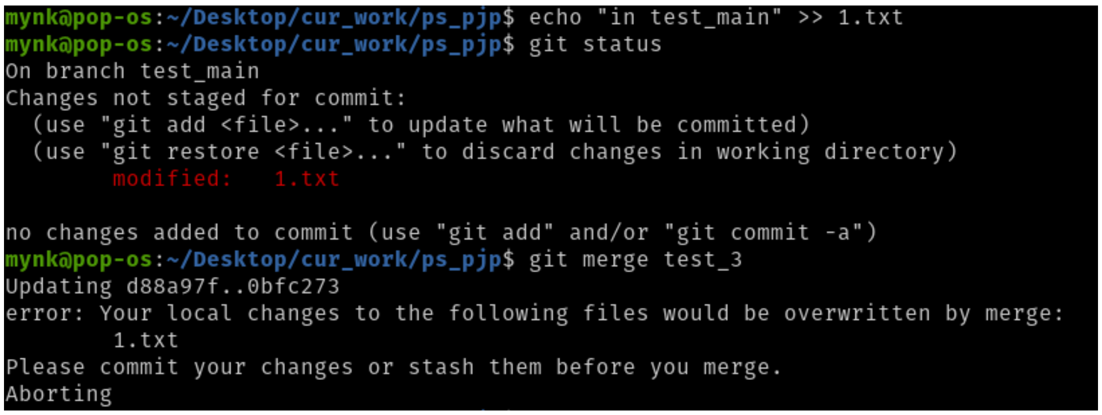
2. Stash changes during merging

Let’s create a new branch and modify and commit a file



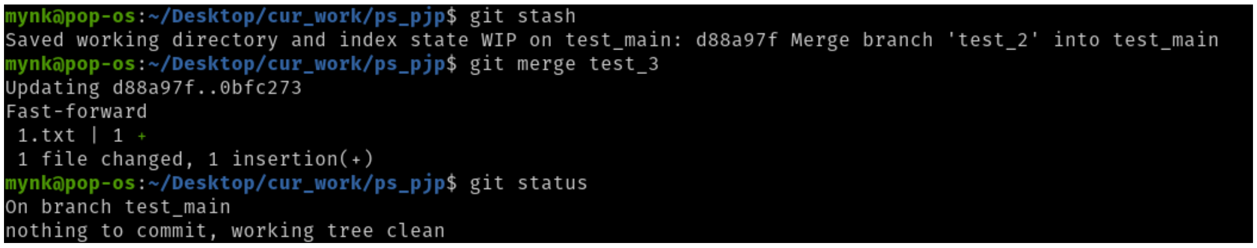
```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
nothing to commit, working tree clean
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout -b test_3
Switched to a new branch 'test_3'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ echo "from test_3" >> 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git add 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git commit -m "from test 3"
[test_3 0bfc273] from test 3
 1 file changed, 1 insertion(+)
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git checkout test_main
Switched to branch 'test_main'
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
nothing to commit, working tree clean
```

Now in the original   branch modify the same file without commiting and trying to merge the newly created branch



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ echo "in test_main" >> 1.txt
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
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:   1.txt

no changes added to commit (use "git add" and/or "git commit -a")
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git merge test_3
Updating d88a97f..0bfc273
error: Your local changes to the following files would be overwritten by merge:
        1.txt
Please commit your changes or stash them before you merge.
Aborting
```

Now let’s stash the current changes and merge the branch



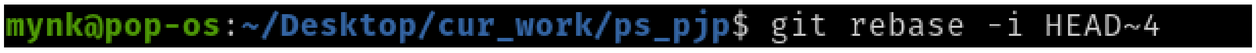
```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git stash
Saved working directory and index state WIP on test_main: d88a97f Merge branch 'test_2' into test_main
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git merge test_3
Updating d88a97f..0bfc273
Fast-forward
 1.txt | 1 +
 1 file changed, 1 insertion(+)
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
nothing to commit, working tree clean
```

Further we can can use **git stash pop**
But that would lead to a merge conflict which we will need to resolve

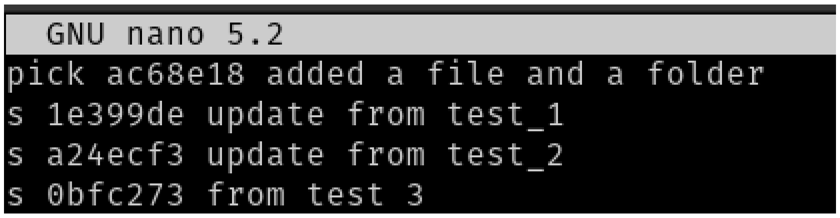
3. Rebasing with different options

rebasing branch x from y basically makes it as if x is a fork of y, git will write new commits (which may look the same but are actually different internally) to achieve this task, and if a common file has been changed in the history of x and y both then the conflict needs to be resolved by the use

Using rebase to squash history

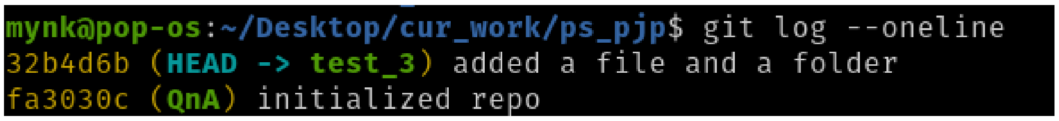


```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git rebase -i HEAD~4
```



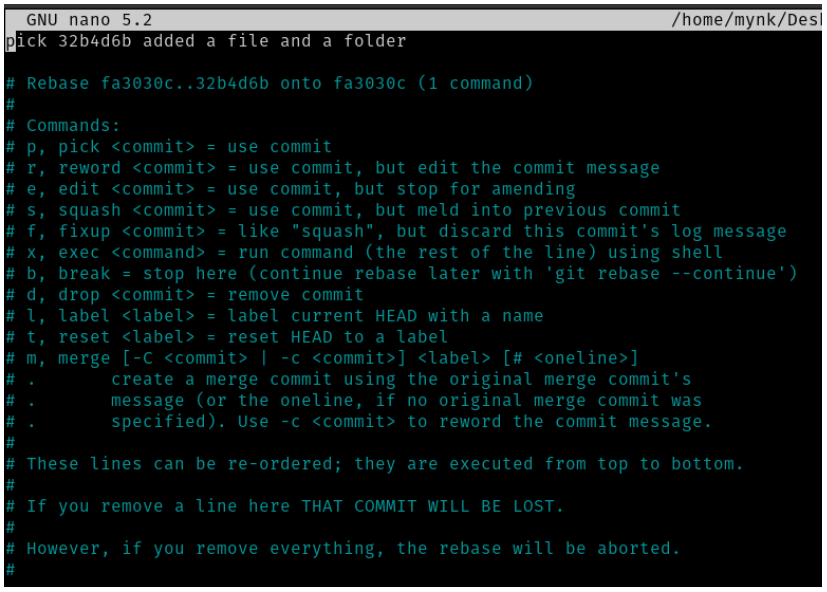
```
  GNU nano 5.2
pick ac68e18 added a file and a folder
s 1e399de update from test_1
s a24ecf3 update from test_2
s 0bfc273 from test 3
```

After resolving the merge conflicts



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git log --oneline
32b4d6b (HEAD -> test_3) added a file and a folder
fa3030c (QnA) initialized repo
```

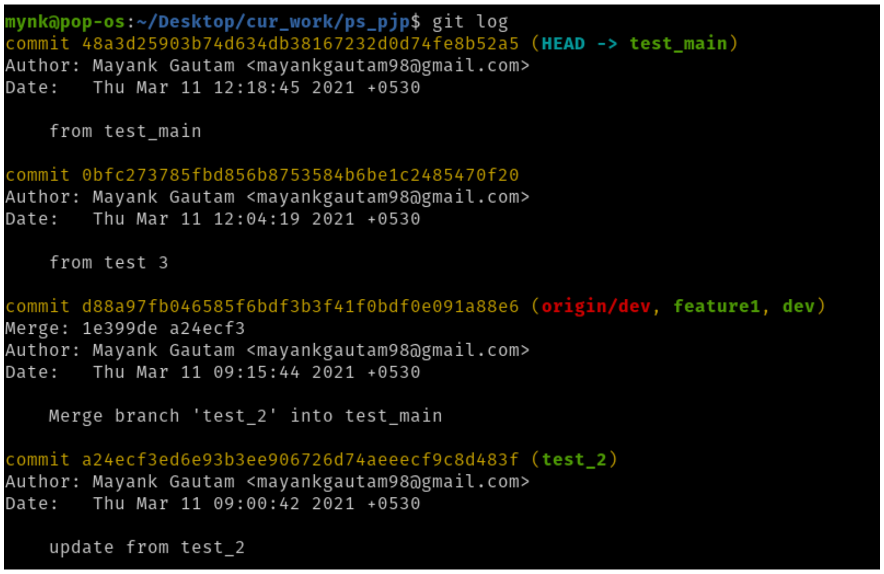
Various commands can be scene and used with running rebase wit -i (interactive) flag



```
  GNU nano 5.2                                              /home/mynk/Desk
pick 32b4d6b added a file and a folder

# Rebase fa3030c..32b4d6b onto fa3030c (1 command)
#
# 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 <commit> = like "squash", but discard this commit's log message
# 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
# .       specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
```

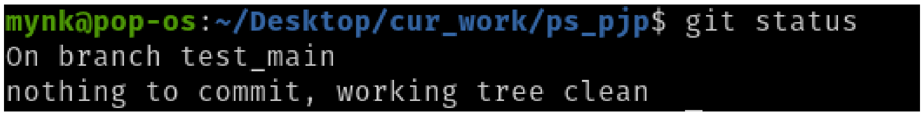
4. log , status and reflog

git log lets us see the entire commit history from the perspective of current head



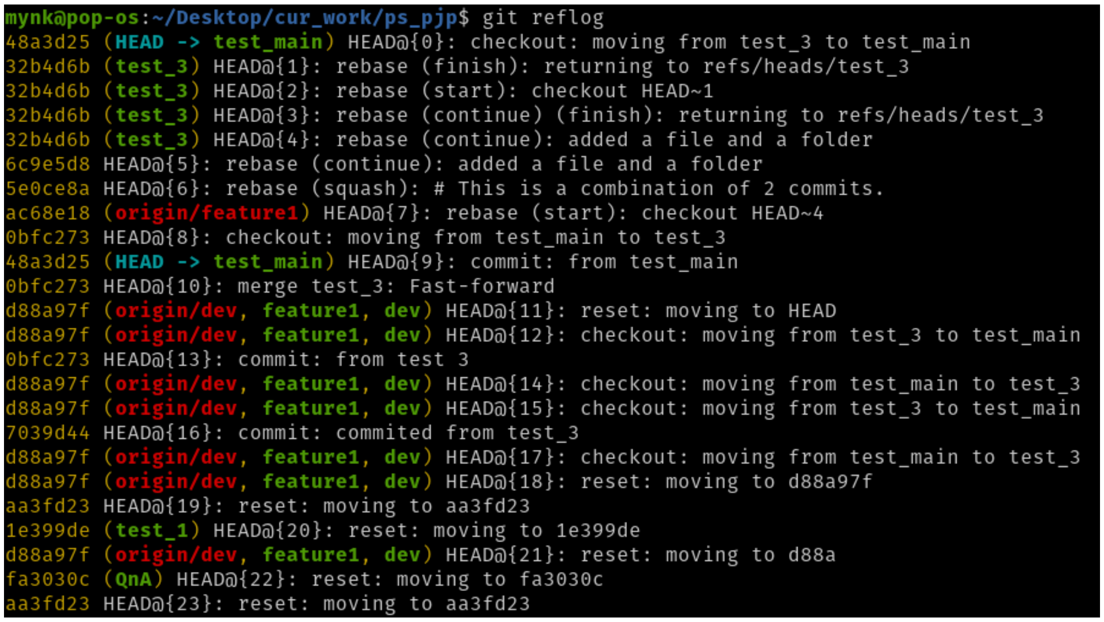
```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git log
commit 48a3d25903b74d634db38167232d0d74fe8b52a5 (HEAD -> test_main)
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Thu Mar 11 12:18:45 2021 +0530

    from test_main

commit 0bfc273785fbd856b8753584b6be1c2485470f20
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Thu Mar 11 12:04:19 2021 +0530

    from test 3

commit d88a97fb046585f6bdf3b3f41f0bdf0e091a88e6 (origin/dev, feature1, dev)
Merge: 1e399de a24ecf3
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Thu Mar 11 09:15:44 2021 +0530

    Merge branch 'test_2' into test_main

commit a24ecf3ed6e93b3ee906726d74aeeecf9c8d483f (test_2)
Author: Mayank Gautam <mayankgautam98@gmail.com>
Date:   Thu Mar 11 09:00:42 2021 +0530

    update from test_2
```

git status shows the status of current working tree, including tracked / untracked files



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git status
On branch test_main
nothing to commit, working tree clean
```

git reflog records basically every action ever performed on a git repository



```
mynk@pop-os:~/Desktop/cur_work/ps_pjp$ git reflog
48a3d25 (HEAD -> test_main) HEAD@{0}: checkout: moving from test_3 to test_main
32b4d6b (test_3) HEAD@{1}: rebase (finish): returning to refs/heads/test_3
32b4d6b (test_3) HEAD@{2}: rebase (start): checkout HEAD~1
32b4d6b (test_3) HEAD@{3}: rebase (continue) (finish): returning to refs/heads/test_3
32b4d6b (test_3) HEAD@{4}: rebase (continue): added a file and a folder
6c9e5d8 HEAD@{5}: rebase (continue): added a file and a folder
5e0ce8a HEAD@{6}: rebase (squash): # This is a combination of 2 commits.
ac68e18 (origin/feature1) HEAD@{7}: rebase (start): checkout HEAD~4
0bfc273 HEAD@{8}: checkout: moving from test_main to test_3
48a3d25 (HEAD -> test_main) HEAD@{9}: commit: from test_main
0bfc273 HEAD@{10}: merge test_3: Fast-forward
d88a97f (origin/dev, feature1, dev) HEAD@{11}: reset: moving to HEAD
d88a97f (origin/dev, feature1, dev) HEAD@{12}: checkout: moving from test_3 to test_main
0bfc273 HEAD@{13}: commit: from test 3
d88a97f (origin/dev, feature1, dev) HEAD@{14}: checkout: moving from test_main to test_3
d88a97f (origin/dev, feature1, dev) HEAD@{15}: checkout: moving from test_3 to test_main
7039d44 HEAD@{16}: commit: commited from test_3
d88a97f (origin/dev, feature1, dev) HEAD@{17}: checkout: moving from test_main to test_3
d88a97f (origin/dev, feature1, dev) HEAD@{18}: reset: moving to d88a97f
aa3fd23 HEAD@{19}: reset: moving to aa3fd23
1e399de (test_1) HEAD@{20}: reset: moving to 1e399de
d88a97f (origin/dev, feature1, dev) HEAD@{21}: reset: moving to d88a
fa3030c (QnA) HEAD@{22}: reset: moving to fa3030c
aa3fd23 HEAD@{23}: reset: moving to aa3fd23
```