DevOps Lab

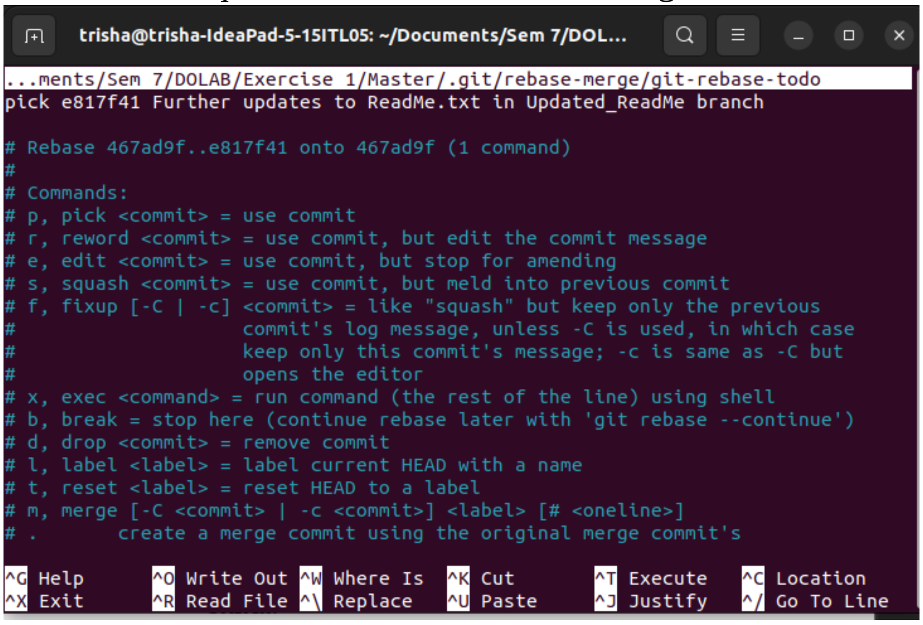
Exercise 4

Trisha Balakrishnan

21011102105

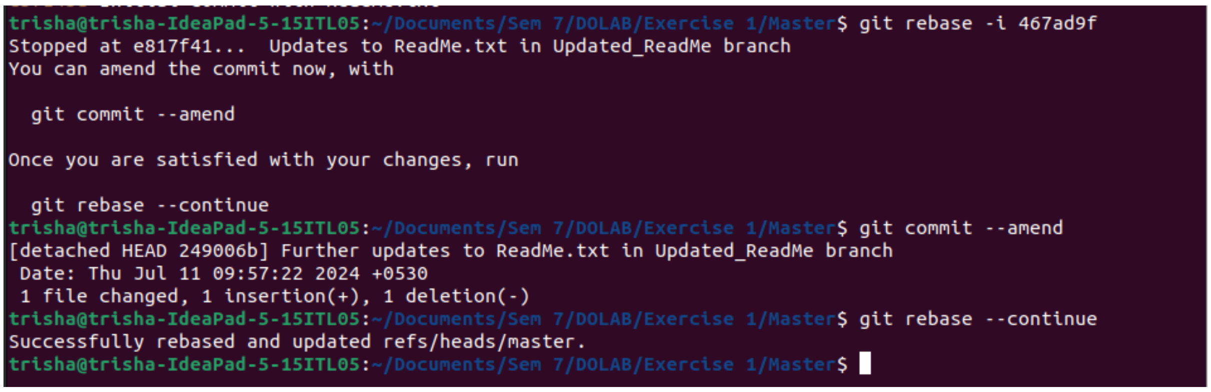
IoT-B

1. **Exploring Git Commands through Collaborative Coding – Advanced Git commands.**
    a. **Use interactive rebase** to combine to multiple commits into one
        - Use git rebase -i to modify the commit history.
        - Interactive Rebase to Edit Multiple Commits
        - In the interactive rebase interface, change pick to edit or squash as needed to edit messages or combine commits.



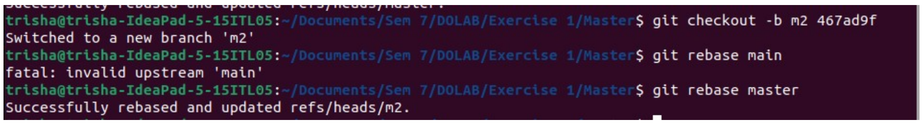
```
trisha@trisha-IdeaPad-5-15ITL05: ~/Documents/Sem 7/DOL...
...ments/Sem 7/DOLAB/Exercise 1/Master/.git/rebase-merge/git-rebase-todo
pick e817f41 Further updates to ReadMe.txt in Updated_ReadMe branch

# Rebase 467ad9f..e817f41 onto 467ad9f (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 [-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

^G Help        ^O Write Out ^W Where Is  ^K Cut        ^T Execute   ^C Location
^X Exit        ^R Read File ^\ Replace   ^U Paste      ^J Justify   ^/ Go To Line
```

- Rebase onto Another Branch.



```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git rebase -i 467ad9f
Stopped at e817f41...  Updates to ReadMe.txt in Updated_ReadMe branch
You can amend the commit now, with

  git commit --amend

Once you are satisfied with your changes, run

  git rebase --continue
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git commit --amend
[detached HEAD 249006b] Further updates to ReadMe.txt in Updated_ReadMe branch
 Date: Thu Jul 11 09:57:22 2024 +0530
 1 file changed, 1 insertion(+), 1 deletion(-)
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git rebase --continue
Successfully rebased and updated refs/heads/master.
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$
```

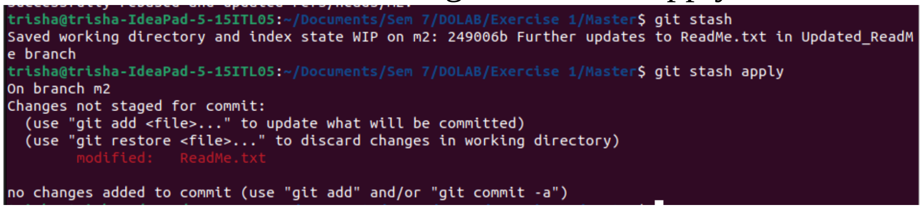
- Create a new branch from an earlier point in your commit history.
- Use rebase to apply the commits from main onto this new branch.



```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git checkout -b m2 467ad9f
Switched to a new branch 'm2'
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git rebase main
fatal: invalid upstream 'main'
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git rebase master
Successfully rebased and updated refs/heads/m2.
```

b. Stash Changes

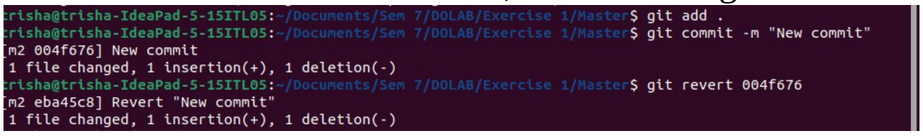
- Make some changes in your working directory.
- Stash those changes, and then apply the stash.



```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git stash
Saved working directory and index state WIP on m2: 249006b Further updates to ReadMe.txt in Updated_ReadM
e branch
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git stash apply
On branch m2
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:   ReadMe.txt

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

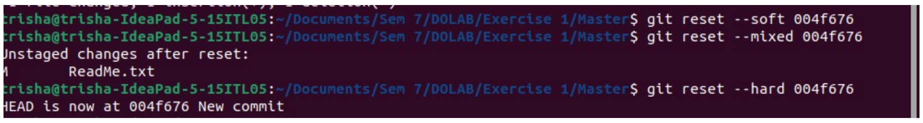
c. Revert and Reset

- Create a new commit, and then use git revert to undo it.



```
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git add .
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git commit -m "New commit"
[m2 004f676] New commit
 1 file changed, 1 insertion(+), 1 deletion(-)
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git revert 004f676
[m2 eba45c8] Revert "New commit"
 1 file changed, 1 insertion(+), 1 deletion(-)
```

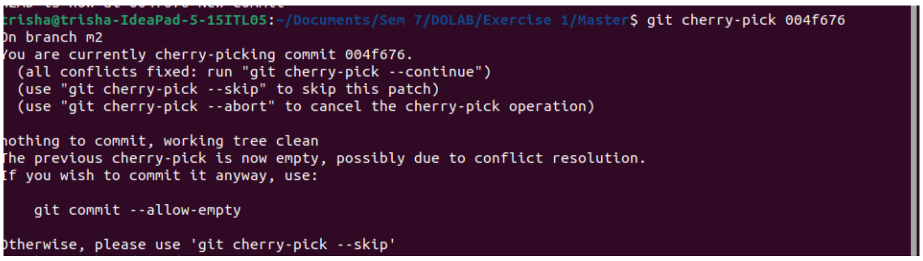
- Experiment with git reset to understand the difference between --soft, --mixed, and –hard.



```
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git reset --soft 004f676
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git reset --mixed 004f676
nstaged changes after reset:
       ReadMe.txt
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git reset --hard 004f676
HEAD is now at 004f676 New commit
```

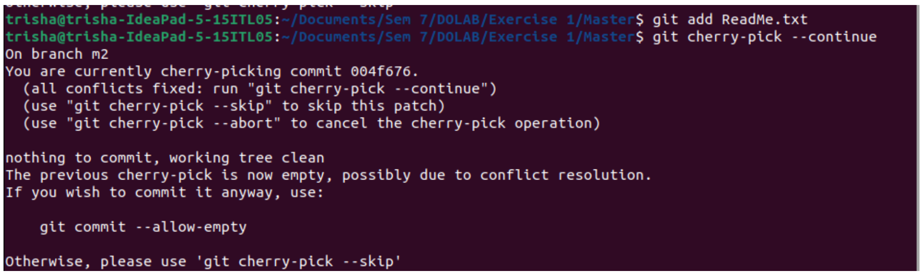
d. Cherry-Pick a Commit

- Resolve Conflicts During Cherry-Picking



```
risha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git cherry-pick 004f676
n branch m2
ou are currently cherry-picking commit 004f676.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

othing to commit, working tree clean
he previous cherry-pick is now empty, possibly due to conflict resolution.
f you wish to commit it anyway, use:

    git commit --allow-empty

therwise, please use 'git cherry-pick --skip'
```

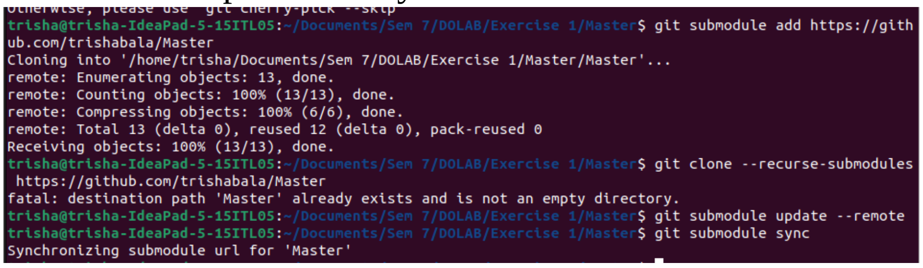
- Resolve Conflicts During Cherry-Picking



```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git add ReadMe.txt
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git cherry-pick --continue
On branch m2
You are currently cherry-picking commit 004f676.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'
```

e. Working with Submodules

- In your repository, add another repository as a submodule.
- Clone a Repository with Submodules
- Update and Synchronize Submodules

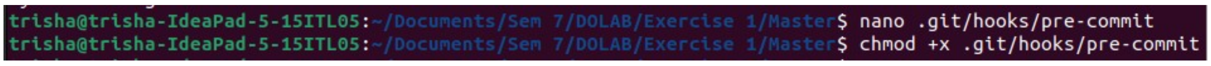


```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git submodule add https://gith
ub.com/trishabala/Master
Cloning into '/home/trisha/Documents/Sem 7/DOLAB/Exercise 1/Master/Master'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 13 (delta 0), reused 12 (delta 0), pack-reused 0
Receiving objects: 100% (13/13), done.
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git clone --recurse-submodules
 https://github.com/trishabala/Master
fatal: destination path 'Master' already exists and is not an empty directory.
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git submodule update --remote
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git submodule sync
Synchronizing submodule url for 'Master'
```

f. Git Hooks

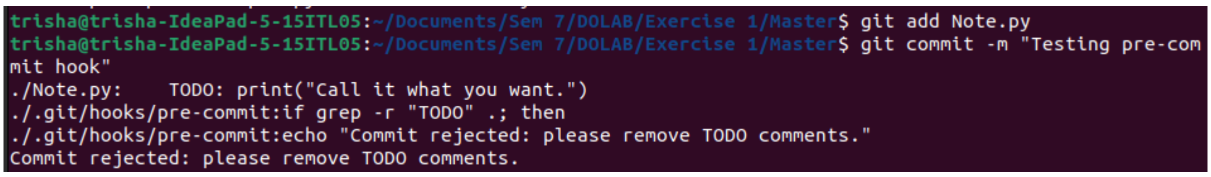
- Create a Pre-Commit Hook that prevents commits with “TODO” comments
- Use the below script

```
#!/bin/sh
if grep -r "TODO" .; then
    echo "Commit rejected: please remove TODO
comments."
    exit 1

fi
```



```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ nano .git/hooks/pre-commit
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ chmod +x .git/hooks/pre-commit
```

- Make the script executable and try committing a file with a TODO comment.



```
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git add Note.py
trisha@trisha-IdeaPad-5-15ITL05:~/Documents/Sem 7/DOLAB/Exercise 1/Master$ git commit -m "Testing pre-com
mit hook"
./Note.py:    TODO: print("Call it what you want.")
./.git/hooks/pre-commit:if grep -r "TODO" .; then
./.git/hooks/pre-commit:echo "Commit rejected: please remove TODO comments."
Commit rejected: please remove TODO comments.
```

- Explore other types of hooks like post-merge, pre-push, or post-checkout, and create simple scripts for them.