Name - Vaibhav Dixit

1. Set your username and email in git config

   For setting username, we can use this git command -

   ```
   git config --global user.name "Your Username"
   ```

   For setting email -

   ```
   git config --global user.email "your.email@example.com"
   ```

2. Create a new branch named "feature-branch" and switch to it.

   ```
   git checkout -b feature-branch
   ```

3. List all branches in the repository.

   ```
   git branch
   ```

4. Delete the branch “feature-branch”

   ```
   git branch -d feature-branch
   ```

5. How do you undo the last commit

   ```
   git reset HEAD~1
   ```

6. Create a new branch names “conflict-branch”

   ```
   git checkout -b conflict-branch
   ```

7. Create a another branch named “feature1”

   ```
   git checkout -b feature1
   ```

8. Make some changes in to feature1 branch

   Assuming that we are at branch feature1 as we have checked out to it while creating it in the previous step. I would make some file changes and then

   Git add .

   Lets assume that we are making change to file only -

   Git add file1

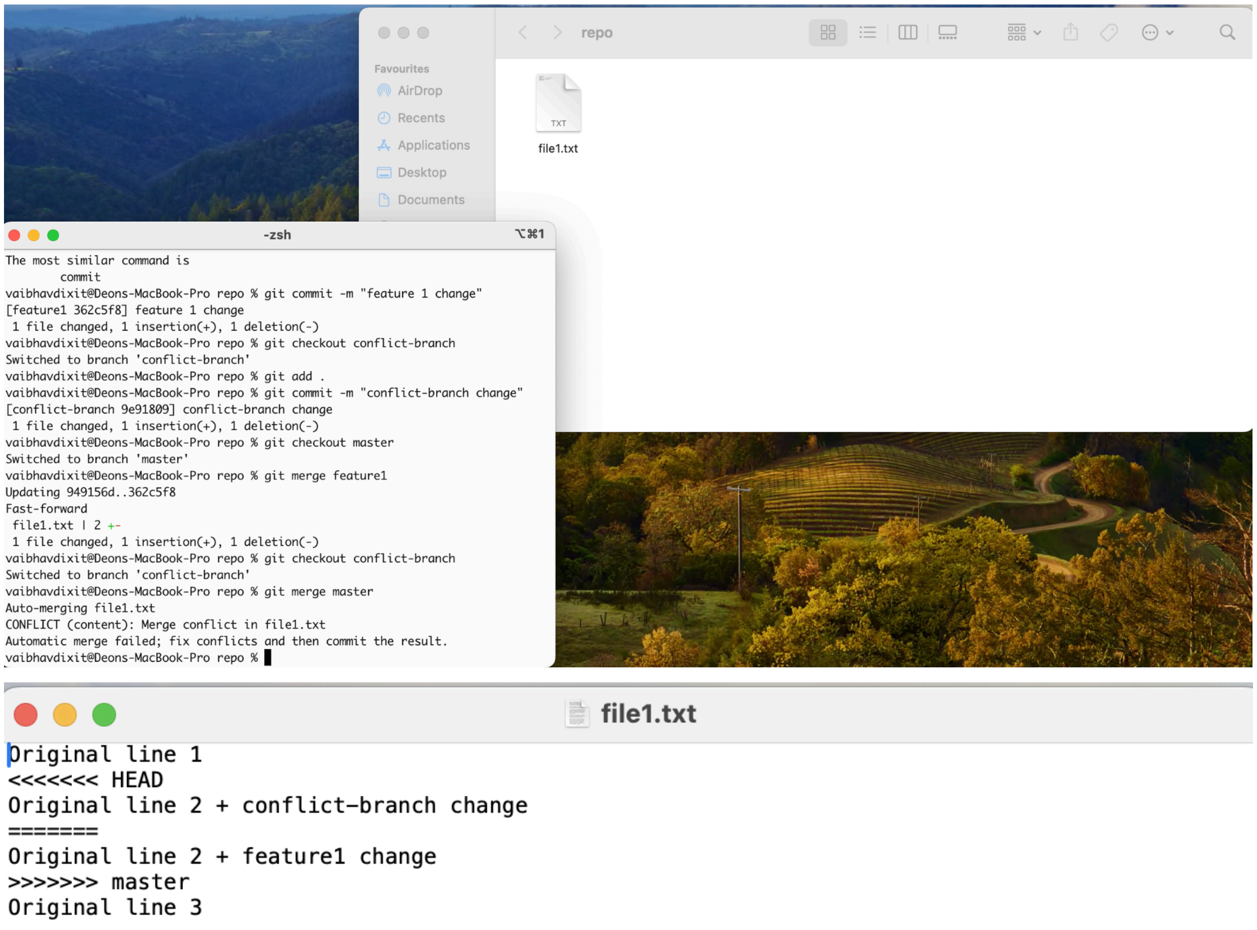
9. Merge “feature1” branch into main branch

   git checkout main
   git merge feature1

10. Make changes in “conflict-branch”, in the same file and line that you had made changes in feature1

    git checkout conflict-branch
    # Making changes in the same file (file1) and line (line 2)) as in feature1
    git add file1

11. Merge master into conflict-branch [Attach screenshot of terminal & file]

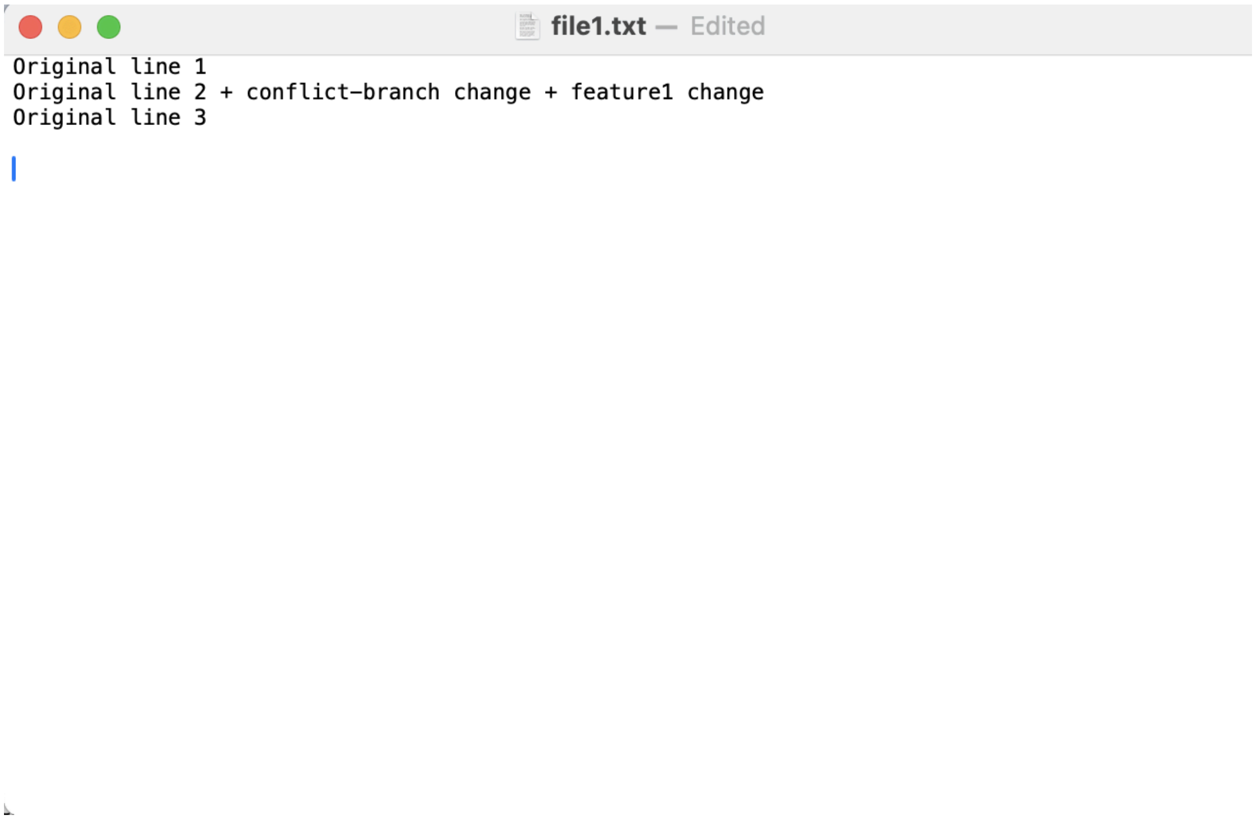


```
The most similar command is
        commit
vaibhavdixit@Deons-MacBook-Pro repo % git commit -m "feature 1 change"
[feature1 362c5f8] feature 1 change
 1 file changed, 1 insertion(+), 1 deletion(-)
vaibhavdixit@Deons-MacBook-Pro repo % git checkout conflict-branch
Switched to branch 'conflict-branch'
vaibhavdixit@Deons-MacBook-Pro repo % git add .
vaibhavdixit@Deons-MacBook-Pro repo % git commit -m "conflict-branch change"
[conflict-branch 9e91809] conflict-branch change
 1 file changed, 1 insertion(+), 1 deletion(-)
vaibhavdixit@Deons-MacBook-Pro repo % git checkout master
Switched to branch 'master'
vaibhavdixit@Deons-MacBook-Pro repo % git merge feature1
Updating 949156d..362c5f8
Fast-forward
 file1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
vaibhavdixit@Deons-MacBook-Pro repo % git checkout conflict-branch
Switched to branch 'conflict-branch'
vaibhavdixit@Deons-MacBook-Pro repo % git merge master
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
vaibhavdixit@Deons-MacBook-Pro repo %
```

file1.txt

```
Original line 1
<<<<<<< HEAD
Original line 2 + conflict-branch change
=======
Original line 2 + feature1 change
>>>>>>> master
Original line 3
```

12. Resolve merge conflicts

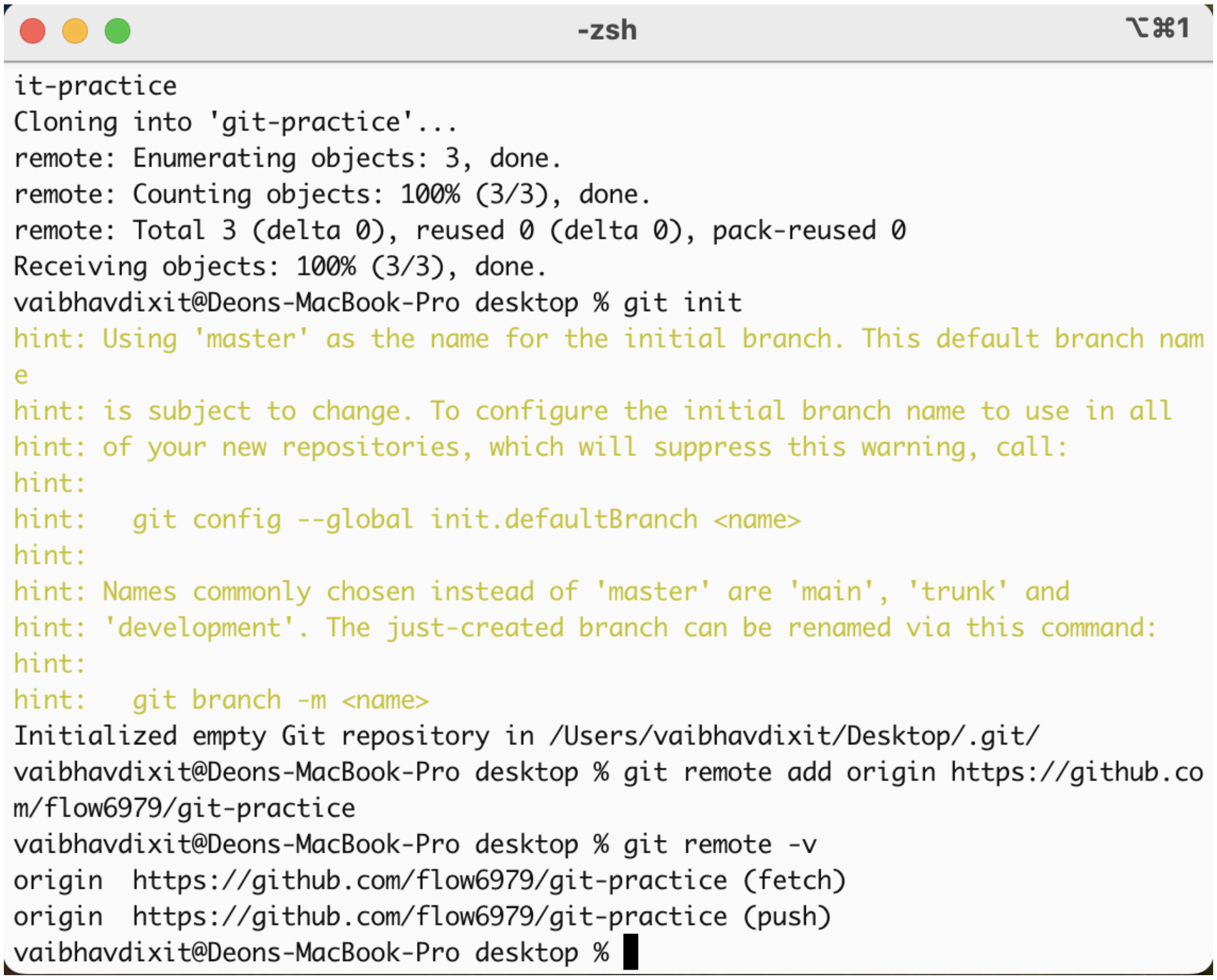


```
Original line 1
Original line 2 + conflict-branch change + feature1 change
Original line 3
```

```
git add .
git commit -m "Resolved merge conflict"
```

13. Add a remote named "origin" pointing to a GitHub repository.

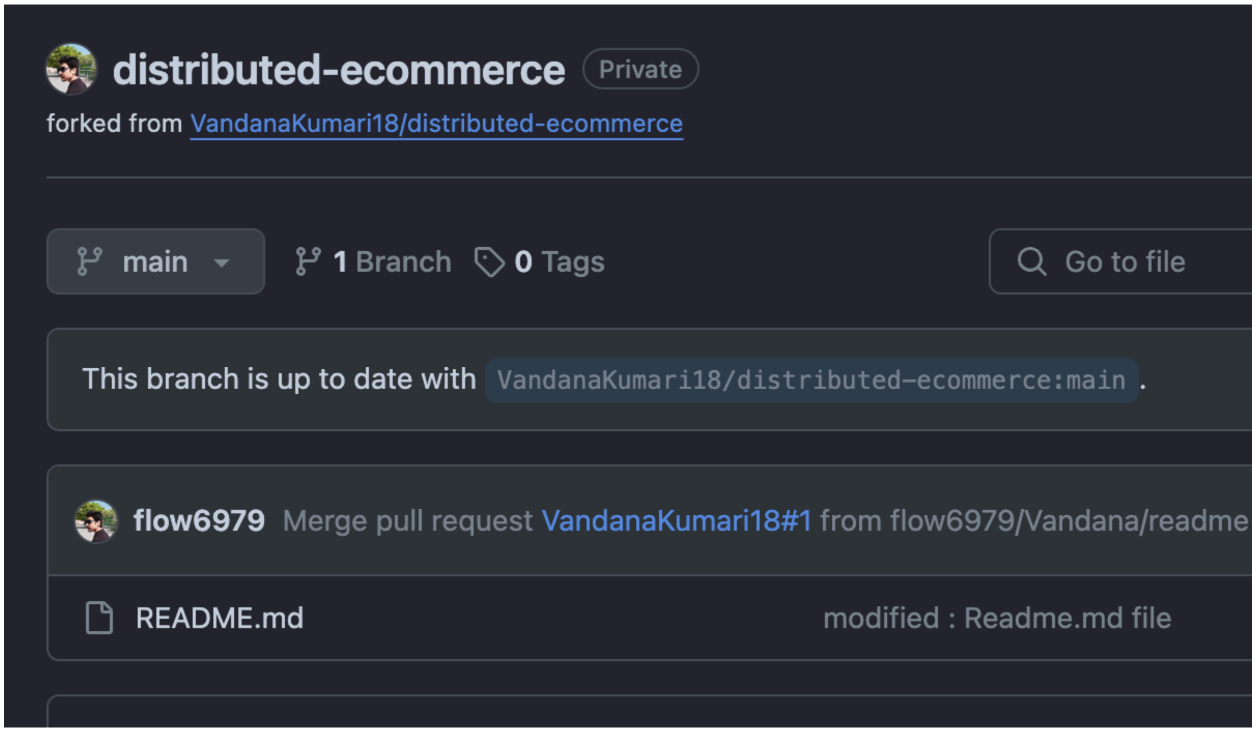
Git remote add origin https://github.com/flow6979/git-practice



```
it-practice
Cloning into 'git-practice'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
vaibhavdixit@Deons-MacBook-Pro desktop % git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
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 /Users/vaibhavdixit/Desktop/.git/
vaibhavdixit@Deons-MacBook-Pro desktop % git remote add origin https://github.co
m/flow6979/git-practice
vaibhavdixit@Deons-MacBook-Pro desktop % git remote -v
origin  https://github.com/flow6979/git-practice (fetch)
origin  https://github.com/flow6979/git-practice (push)
vaibhavdixit@Deons-MacBook-Pro desktop %
```

git remote -v   (for verifying)

14. Fork a repository on GitHub and clone it to your local machine.


distributed-ecommerce Private

forked from VandanaKumari18/distributed-ecommerce

main 1 Branch 0 Tags Go to file

This branch is up to date with `VandanaKumari18/distributed-ecommerce:main`.

flow6979 Merge pull request VandanaKumari18#1 from flow6979/Vandana/readme

README.md modified : Readme.md file

Git clone https://github.com/flow6979/distributed-ecommerce.git

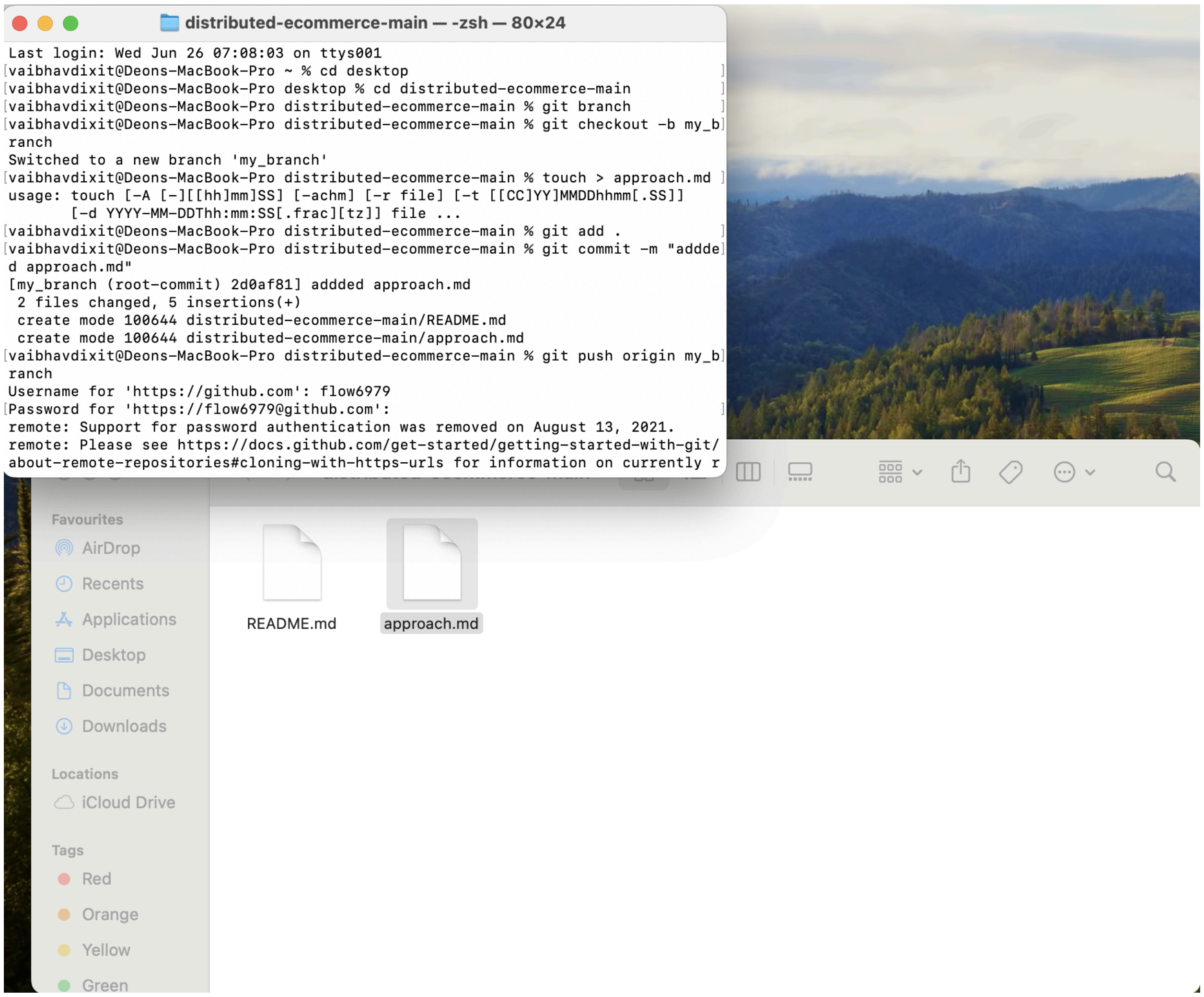
15. Create a new branch on your fork, make changes, and open a pull request to the original repository.

    git checkout -b my_branch

    Added approach.md

    git add .
    git commit -m "Added approach.md"

    git push origin my_branch

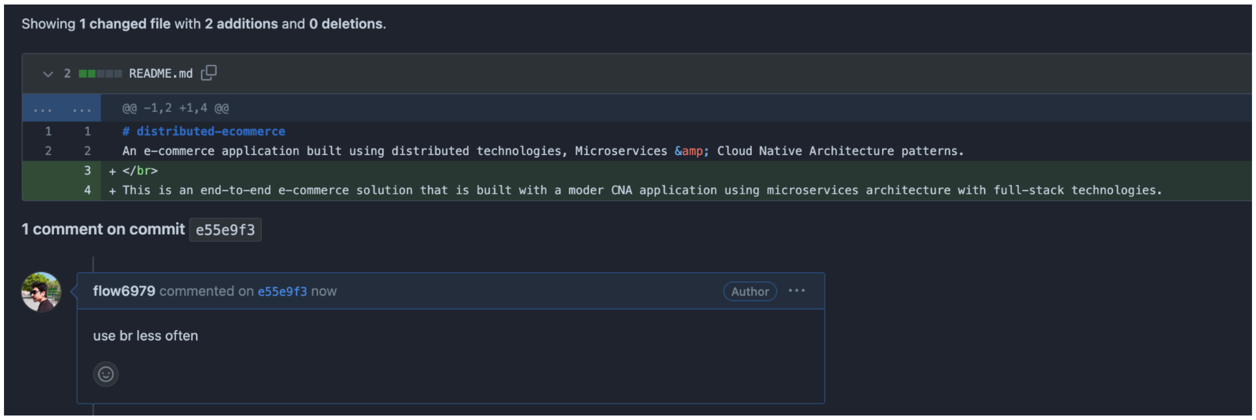


```
Last login: Wed Jun 26 07:08:03 on ttys001
vaibhavdixit@Deons-MacBook-Pro ~ % cd desktop
vaibhavdixit@Deons-MacBook-Pro desktop % cd distributed-ecommerce-main
vaibhavdixit@Deons-MacBook-Pro distributed-ecommerce-main % git branch
vaibhavdixit@Deons-MacBook-Pro distributed-ecommerce-main % git checkout -b my_b
ranch
Switched to a new branch 'my_branch'
vaibhavdixit@Deons-MacBook-Pro distributed-ecommerce-main % touch > approach.md
usage: touch [-A [-][[hh]mm]SS] [-achm] [-r file] [-t [[CC]YY]MMDDhhmm[.SS]]
       [-d YYYY-MM-DDThh:mm:SS[.frac][tz]] file ...
vaibhavdixit@Deons-MacBook-Pro distributed-ecommerce-main % git add .
vaibhavdixit@Deons-MacBook-Pro distributed-ecommerce-main % git commit -m "adde
d approach.md"
[my_branch (root-commit) 2d0af81] addded approach.md
 2 files changed, 5 insertions(+)
 create mode 100644 distributed-ecommerce-main/README.md
 create mode 100644 distributed-ecommerce-main/approach.md
vaibhavdixit@Deons-MacBook-Pro distributed-ecommerce-main % git push origin my_b
ranch
Username for 'https://github.com': flow6979
Password for 'https://flow6979@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/
about-remote-repositories#cloning-with-https-urls for information on currently r
```

There is some authentication issue while cloning and pushing

16. Comment on a PR and suggest improvements

ß


Showing **1 changed file** with **2 additions** and **0 deletions**.

```
2 README.md
@@ -1,2 +1,4 @@
   # distributed-ecommerce
   An e-commerce application built using distributed technologies, Microservices &amp; Cloud Native Architecture patterns.
 + </br>
 + This is an end-to-end e-commerce solution that is built with a moder CNA application using microservices architecture with full-stack technologies.
```

1 comment on commit e55e9f3

flow6979 commented on e55e9f3 now — Author

use br less often

17. Create a Git alias for the command `git log --oneline` named `gitlol`.

    git config --global alias.gitlol "log --oneline"

    Now, git lol & git log --oneline should do same thing

18. Create a pre-commit hook

    Steps -
    1. cd .git/hooks (inside our repo)
    2. touch pre-commit (creating new file)
    3. chmod +x pre-commit (making it executable)
    4. nano pre-commit (editing our file using text editor)
    5. Add pre-commit hook script, then save and exit the editor

    Pre-commit Hook: This script runs before each commit is processed. It allows you to enforce policies or checks (like code linting, tests, etc.) before allowing a commit to proceed.

19. You have made local changes in your branch, but you need to switch to another branch urgently without committing. How would you handle this situation?

    Stash

    git stash
    git checkout branch2 (going from branch1 to branch2)
    git stash apply

    We can also create other worktree for other branch

20. You accidentally deleted a file in your local repository. How do you restore it using Git?

    git checkout HEAD -- deleted_file_path

    retrieves the specified file from the most recent commit (HEAD) in your Git history and restores it to your working directory.

21. You have committed changes to your branch but forgot to include a file. How do you add the file to the last commit without creating a new commit?

    Git add file_path
    Git commit –amend –no-edit

    –amend: Incorporates changes into the last commit instead of creating a new one.
    –no-edit: Prevents Git from opening the commit message editor, preserving the existing commit message.

22. You want to discard all changes in your working directory and revert to the last commit. What Git command would you use?

    Git reset

    or

    git reset --hard HEAD

    (--hard -> reset both stagin area and working directory)

23. You need to view a specific commit's changes. What Git command can be used to show the changes introduced by a particular commit?

    Git show commit-hash

    commit -hash -> unique alphanumeric string assigned to each commit in git.
24. You want to change a commit message, after you have already committed, how do you do so?

    git commit --amend -m "New commit message"

    Or

git commit --amend

Edit in editor -> save and close the editor

25. Your colleague has made changes in their branch, and you want to incorporate those changes into your branch without merging. How do you achieve this?

    Steps -
    1. Identify commit hashes from colleague's branch
       git checkout colleague-branch
       git log
    2. git checkout your-branch
    3. git cherry-pick <commit-hash>
       Or
       git cherry-pick <commit-hash1> <commit-hash2> ..   (for specific hashes)
       Or
       git cherry-pick <commit-hash1>..<commit-hash2>   (for range)

26. You've made several commits on a branch, but you want to club them into a single commit before pushing to the remote repository. How would you do that?

    Commit squashing -
    1. Start an interactive rebase
       git rebase -i HEAD~n   (n = number of commits to merge)
    2. An editor will open with our commits prefixed with pick, except for first replace all picks with squash. Then save and close editor. Another editor will open to edit combined commit message.

       A force push may require to rewrite the commit history (git push origin feature-branch --force)

27. You accidentally staged a file that you don't want to commit. How do you unstage it?

    Git reset

    Or

    Git reset HEAD file_name

28. You don't want to commit files that have .yml in the end, and also files inside folder config. How do you do that?

Use gitignore

touch .gitignore
Add rules in .gitignore, like -
```
*.yml
config/
```

29. You want to see a list of all the files changed in the last commit. What Git command would you use?

    git show --name-only HEAD

    Or

    git diff --name-only HEAD~1 HEAD

30. You realize that your local branch is outdated, and you want to fetch the latest changes from the remote repository. How do you do this without merging?

    git fetch

31. You accidentally deleted a branch. How do you recover it?

    1. git reflog (to find commit hash of last commit on deleted branch)
    2. git checkout -b new_branch <commit-hash>

32. You want to remove untracked files and directories from your working directory. What Git command would you use?

    1. Dry run: git clean -n
    2. Removing untracked files: git clean -f
    3. Removing untracked directories: git clean -fd
    4. Removing ignored files: git clean -fdx

33. You have a commit from a feature branch that you want to apply to the main branch without merging the entire feature branch.

    1. git log --oneline feature-branch
    2. git checkout main

3. git cherry-pick <commit-hash>
4. Resolve conflicts

   git add resolved_file

   git cherry-pick --continue

34. You mistakenly committed a change to the wrong branch and need to apply that commit to the correct branch.

    1. git log --oneline (identify commit hash)
    2. git checkout correct_branch
    3. git cherry-pick <commit-hash>
    4. Resolve conflicts

       git add resolved_file

       git cherry-pick --continue

35. There is a series of commits on a feature branch, but you only want to cherry-pick a specific range of commits.

1. Identify commit hashes

   git log --oneline feature_branch

2. git checkout our_branch
3. git cherry-pick <start-commit>^..<end-commit>
4. Resolve conflicts

   git add resolved_file

   git cherry-pick --continue

36. You want to clone a GitHub repository onto your local machine, but you only need a specific branch. How can you achieve this?

    git clone -b branch_name repository_url

37. You've made changes to your local repository and want to push them to your fork on GitHub. What Git commands would you use?

    git add .

    git commit -m "commit message"

    git push origin branch_name

38. You want to create a new branch both locally and on GitHub to work on a new feature. What commands would you use?

Creating locally: git checkout -b branch_name
Pushing to github: git push -u origin branch_name

39. You want to see the commit history of a GitHub repository. How can you do this using Git commands?

    git log

40. You've accidentally committed sensitive information and want to remove the commit from both your local and remote repositories on GitHub. What commands would you use?

    1. git reset --hard HEAD~1 (resetting branch head to point before commit, maybe just before current head)
    2. git push --force origin branch_name (force push the updated branch to github repo to overwrite remote history)

41. You want to delete a remote branch on GitHub. What Git command would you use?

    1. git branch -d branch-name (deleting locally)
    2. git push origin --delete branch-name (deleting on github)

42. Create a git repository for all your assignments and upload them in it. Ask your peers to code review it, and you need to code review your peers assignments
43. Create a pull request on any open source library on github, attach the pull request link to the readme file of this project's repository