WORKTREE

What is a Git worktree

- A Git worktree is a linked copy of your Git repository, allowing you to have multiple branches checked out at a time. A worktree has a separate path from your main working copy, but it can be in a different state and on a different branch. The advantage of a new worktree in Git is that you can make a change unrelated to your current task, commit the change, and then merge it at a later date, all without disturbing your current work environment.

Listing active worktrees

- You can get a list of the worktrees and see what branch each has checked out using the git worktree list command:

-
```
$ git worktree list
/home/seth/code/myproject  15fca84 [dev]
/home/seth/code/hotfix     09e585d [master]
```

- You can use this from within either worktree. Worktrees are always linked (unless you manually move them, breaking Git's ability to locate a worktree, and therefore severing the link).

Moving a worktree

- Git tracks the locations and states of a worktree in your project's .git directory:

-
```
$ cat ~/code/myproject/.git/worktrees/hotfix/gitdir
/home/seth/code/hotfix/.git
```

- If you need to relocate a worktree, you must do that using `git worktree move`; otherwise, when Git tries to update the worktree's status, it fails

-
```
$ mkdir ~/Temp
$ git worktree move hotfix ~/Temp
$ git worktree list
/home/seth/code/myproject  15fca84 [dev]
/home/seth/Temp/hotfix     09e585d [master]
```

Removing a worktree

- When you're finished with your work, you can remove it with the remove subcommand:

-
  ```
  $ git worktree remove hotfix
  $ git worktree list
  /home/seth/code/myproject  15fca84 [dev]
  ```

- To ensure your .git directory is clean, use the prune subcommand after removing a worktree:

-
  ```
  $ git worktree prune
  ```

When to use worktrees

- As with many options, whether it's tabs or bookmarks or automatic backups, it's up to you to keep track of the data you generate, or it could get overwhelming. Don't use worktrees so often that you end up with 20 copies of your repo, each in a slightly different state. It is best to create a worktree, do the task that requires it, commit the work, and then remove the tree.
- The important thing is that worktrees provide improved flexibility for how you manage a Git repository. Use them when you need them, and never again scramble to preserve your working state just to check something on another branch.

Group 6 team members

- Sachin Kumar Saini
- Apoorv Gautam
- Shivam Mehta
- Sakshi Sneha
- Mohd Faisal

Thank you