#90DaysOfDevOps Challenge - Day 10 - Advanced Git & GitHub for DevOps Engineers Part 1

Git Branching

Welcome to Day 10 of the #90DaysOfDevOps challenge. Today, we will explore advanced Git techniques, including branching, merging, and reverting. These techniques are essential for effective collaboration and version control in software development projects. So, let's dive in.

Git Branching

Git branching is a feature in Git that allows you to create separate lines of development within a repository. It enables you to work on different features or fixes simultaneously without affecting the main codebase.

Branches serve as independent workspaces where you can make changes, commit them, and merge them back into the main branch when ready. They provide a way to organize and manage different versions or streams of code within a project.

Git Revert and Reset

`Git revert` is a command that undoes a specific commit by creating a new commit that undoes the changes made in that commit. It is a safe way to undo changes without altering the commit history.

`Git reset` is a command that allows you to move the branch pointer to a different commit. It can be used to reset the branch to a previous state. However, it should be used with caution as it can discard or modify changes in the process.

In simple terms, `git revert` undoes a commit by creating a new one, while `git reset` moves the branch pointer to a different commit.

Git Rebase and Merge

What Is Git Rebase?

`Git rebase` is a command that allows you to update your branch with the latest changes from another branch. It rearranges the commit history by moving your changes on top of the updated branch. This helps create a cleaner and more straightforward history of your changes. It's useful for integrating changes, keeping your branch up to date, and making your commit history more organized before merging.

What Is Git Merge?

`Git merge` is a command that combines changes from different branches into a single branch. It takes the changes made in one branch and integrates them into another branch, creating a new commit that includes the changes from both branches. It's used to incorporate the work done in one branch into another, such as merging a feature branch into a main branch. The merge operation keeps a record of the individual branch histories and combines them into a unified branch history.

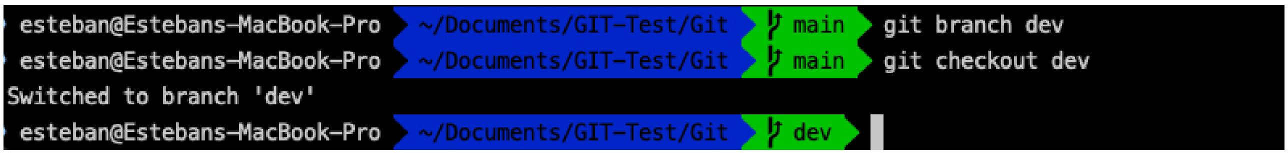
Refer to this article for a better understanding of Git Rebase and Merge Read here

Task 1: Branching, Committing, and Restoring

In this task, we will demonstrate how to create a branch, add commits with different messages, and restore a file to a previous version. Follow the steps below:

1. Create a branch called `dev` from the `main` branch using the command:

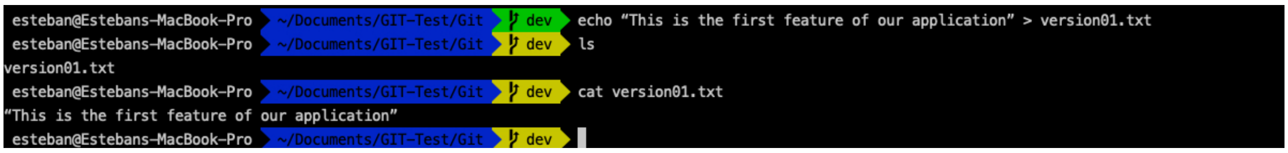
```
git branch dev
git checkout dev
```



```
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git branch dev
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git checkout dev
Switched to branch 'dev'
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev 
```

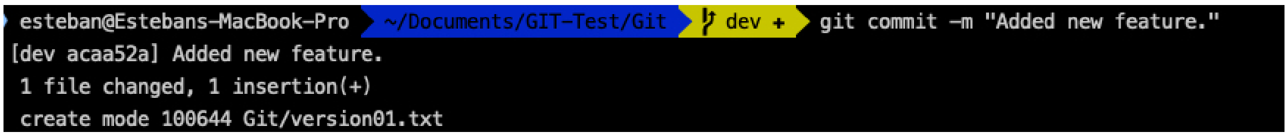
2. Add a text file called `version01.txt` inside the `Devops/Git/` directory. Write the following content inside the file:

```
"This is the first feature of our application."
```



```
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  echo "This is the first feature of our application" > version01.txt
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  ls
version01.txt
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  cat version01.txt
"This is the first feature of our application"
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev 
```

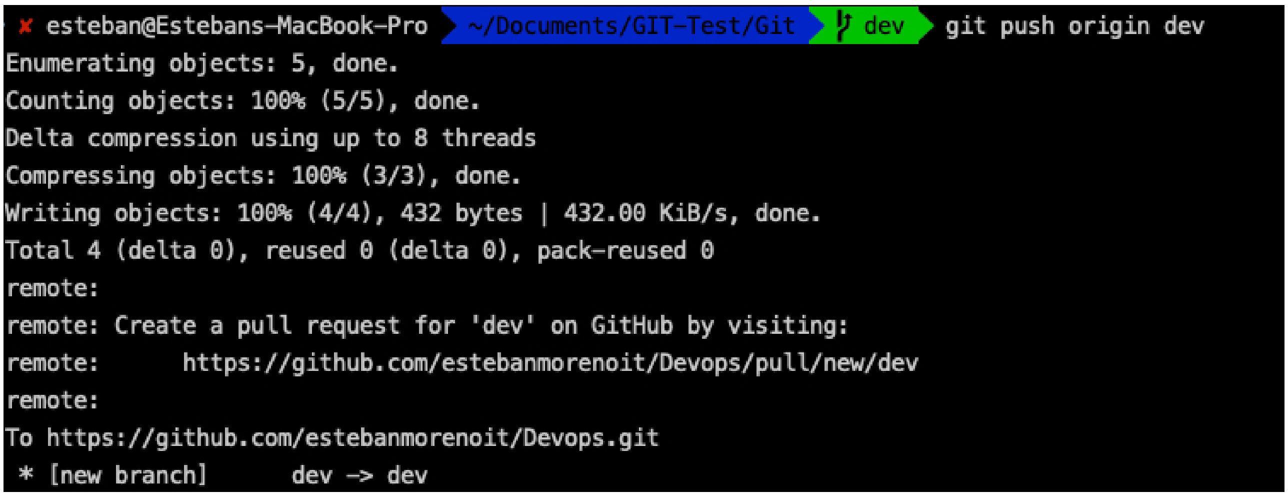
3. Commit this change with the message "Added new feature."



```
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev +  git commit -m "Added new feature."
[dev acaa52a] Added new feature.
 1 file changed, 1 insertion(+)
 create mode 100644 Git/version01.txt
```

4. Push the `dev` branch to the remote repository using the command:

```
git push origin dev
```



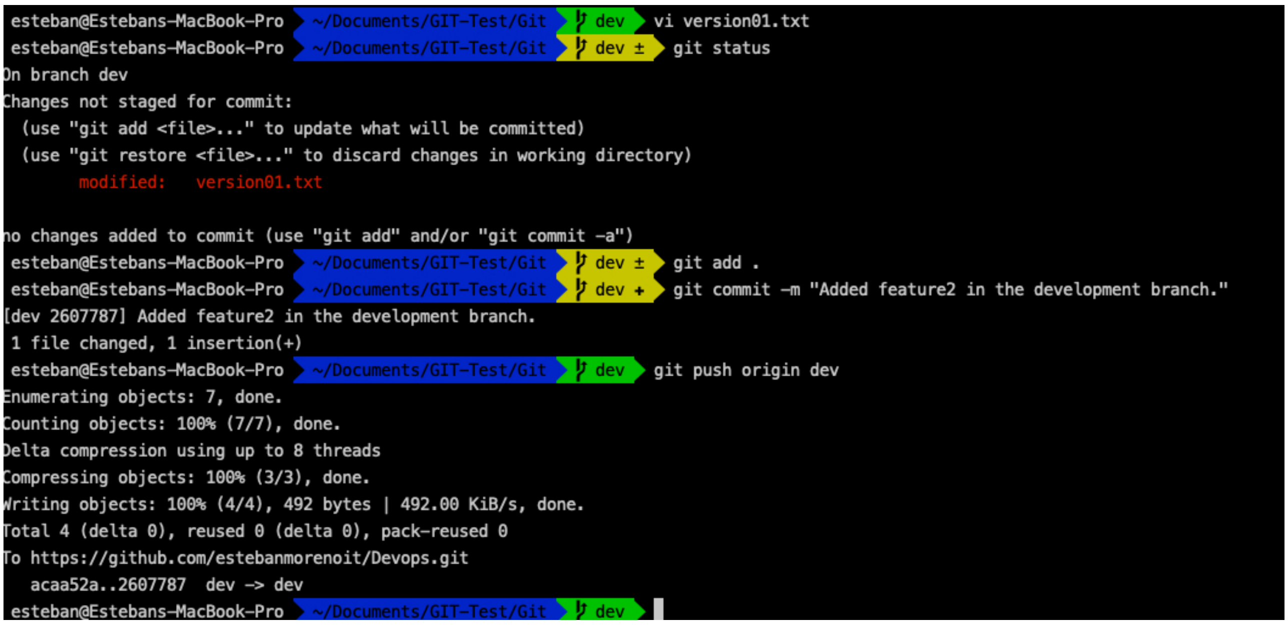
```
✘ esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git push origin dev
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 432 bytes | 432.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/estebanmorenoit/Devops/pull/new/dev
remote:
To https://github.com/estebanmorenoit/Devops.git
 * [new branch]      dev -> dev
```

5. Add new commits to the `dev` branch by modifying the `version01.txt` file with the following content:

```
This is the bug fix in the development branch
```

6. Commit this change with the message

```
"Added feature2 in the development branch."
```



```
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  vi version01.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev ±  git status
On branch dev
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:   version01.txt

no changes added to commit (use "git add" and/or "git commit -a")
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev ±  git add .
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev ✚  git commit -m "Added feature2 in the development branch."
[dev 2607787] Added feature2 in the development branch.
 1 file changed, 1 insertion(+)
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git push origin dev
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 492 bytes | 492.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/estebanmorenoit/Devops.git
   acaa52a..2607787  dev -> dev
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev 
```

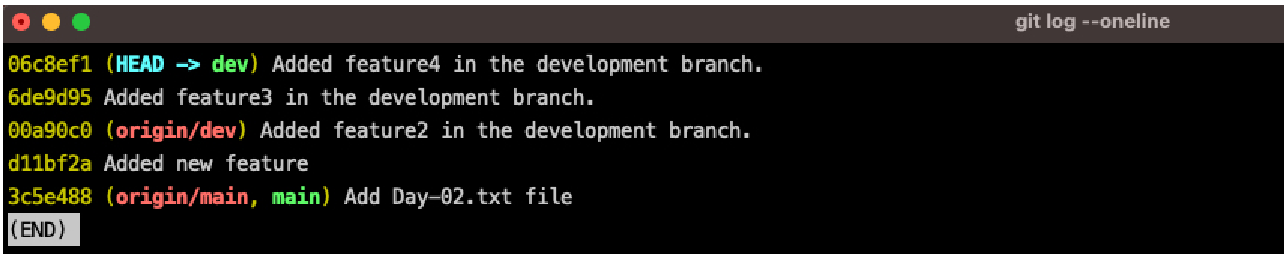
7. Repeat this step two more times, adding the following content and committing with appropriate messages:

```
This is gadbad code
```

```
This feature will gadbad everything from now.
```

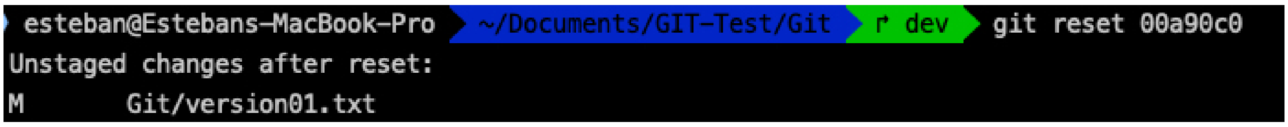
8. Restore the `version01.txt` file to a previous version where the content should be "This is the bug fix in the development branch."

9. Using the `git log --oneline` command, we can find the `<commit>` information and identity the commit you want to reset to.

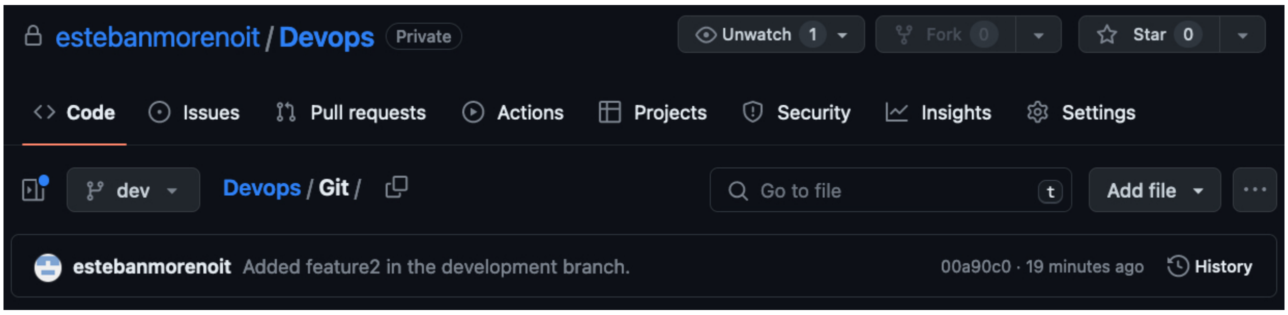


```
git log --oneline
06c8ef1 (HEAD -> dev) Added feature4 in the development branch.
6de9d95 Added feature3 in the development branch.
00a90c0 (origin/dev) Added feature2 in the development branch.
d11bf2a Added new feature
3c5e488 (origin/main, main) Add Day-02.txt file
(END)
```

10. We can use the `git reset <commit>` command to remove the last two commits where we added the second and third lines and move the changes to the unstaged area.



```
esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git reset 00a90c0
Unstaged changes after reset:
M       Git/version01.txt
```



estebanmorenoit / Devops Private

dev Devops / Git /

estebanmorenoit Added feature2 in the development branch. 00a90c0 · 19 minutes ago History

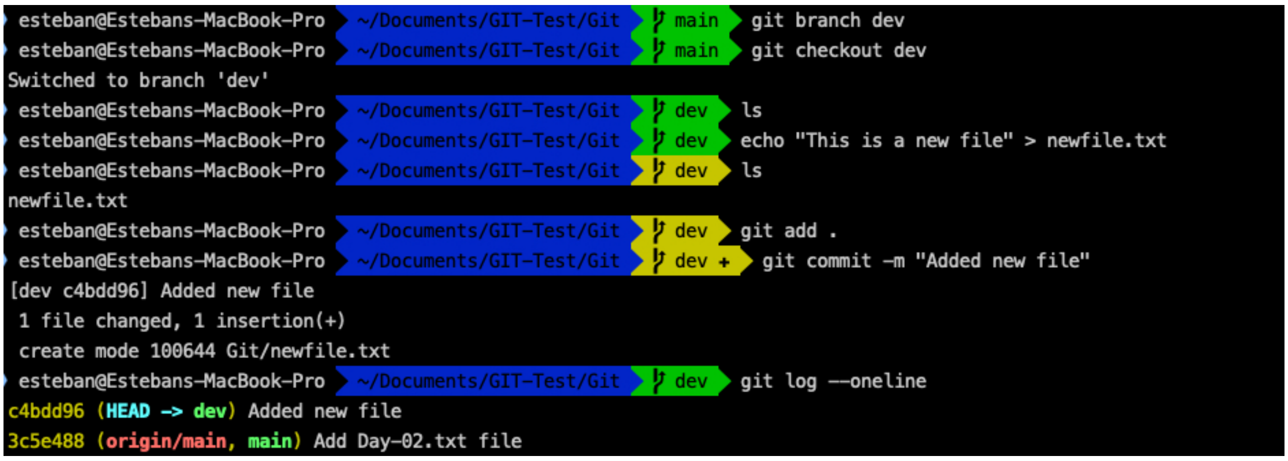
Task 2: Branching, Merging, and Rebasing

In this task, we will demonstrate the concept of branches, merging, and rebasing. Follow the steps below:

1. Create two or more branches with different names using the command `git branch`. In this case, I will use `main` and `dev`:

```
git branch dev
```

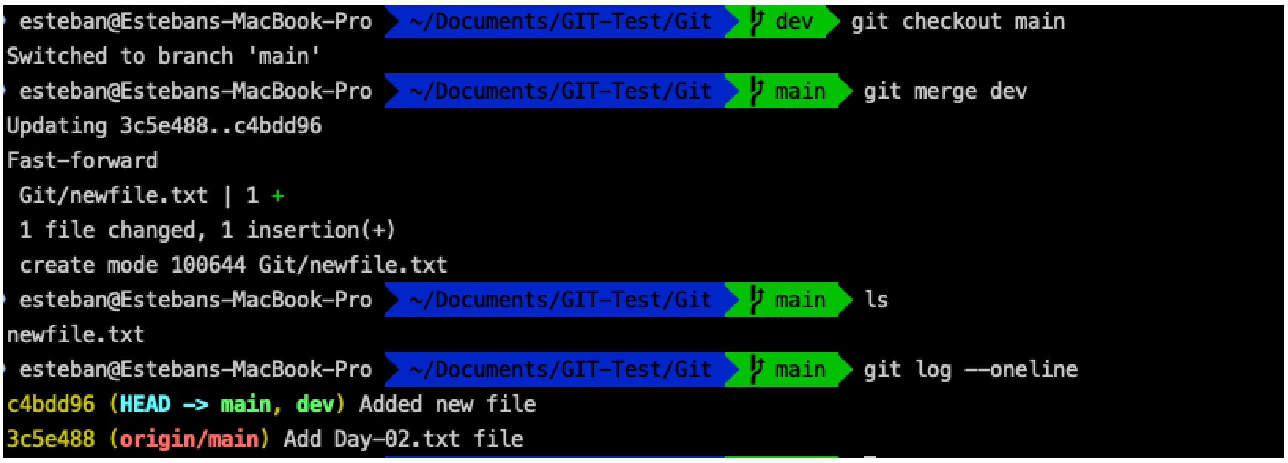
Make some changes to the `dev` branch and commit them. Take a screenshot of the commit history and branch visualization to demonstrate the concept of branches.



```
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git branch dev
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git checkout dev
Switched to branch 'dev'
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  ls
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  echo "This is a new file" > newfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  ls
newfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git add .
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev +  git commit -m "Added new file"
[dev c4bdd96] Added new file
 1 file changed, 1 insertion(+)
 create mode 100644 Git/newfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git log --oneline
c4bdd96 (HEAD -> dev) Added new file
3c5e488 (origin/main, main) Add Day-02.txt file
```

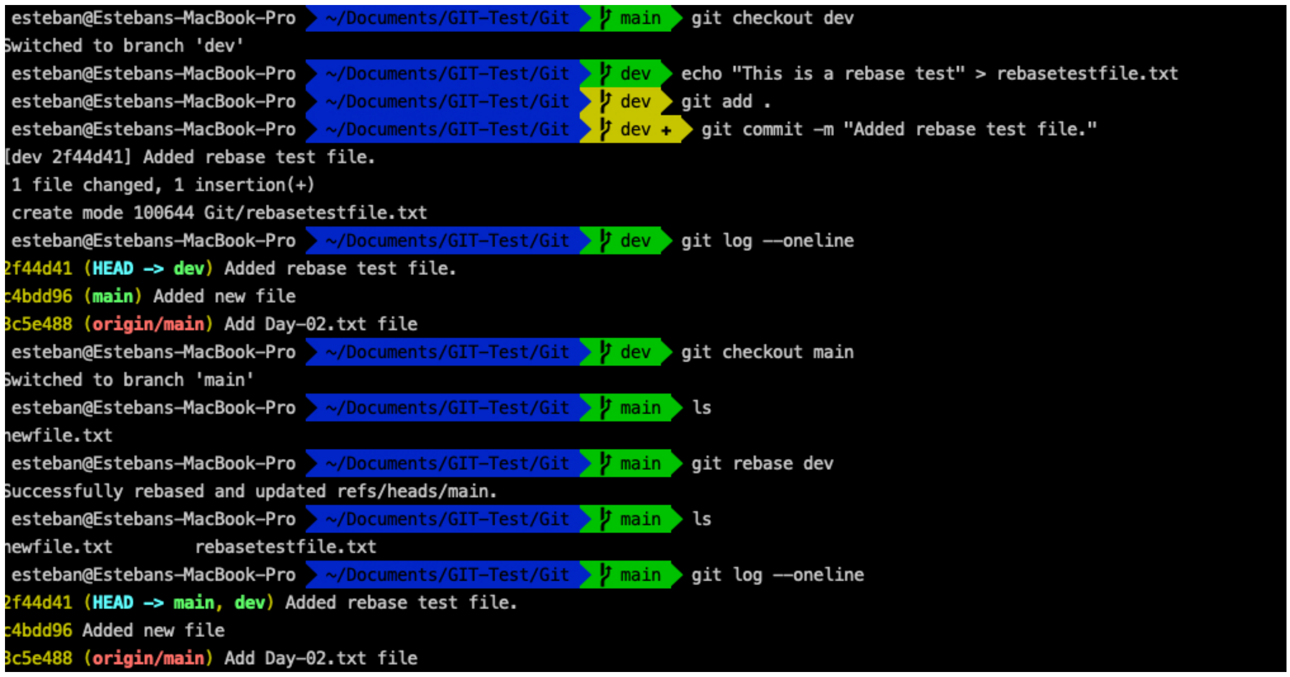
2. Merge the `dev` branch into the `main` branch using the command:

```
git checkout main
git merge dev
```



```
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git checkout main
Switched to branch 'main'
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git merge dev
Updating 3c5e488..c4bdd96
Fast-forward
 Git/newfile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Git/newfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  ls
newfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git log --oneline
c4bdd96 (HEAD -> main, dev) Added new file
3c5e488 (origin/main) Add Day-02.txt file
```

3. As a practice, perform a `git rebase` operation to see the difference it makes. Describe the differences you observe.



```
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git checkout dev
Switched to branch 'dev'
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  echo "This is a rebase test" > rebasetestfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git add .
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev +  git commit -m "Added rebase test file."
[dev 2f44d41] Added rebase test file.
 1 file changed, 1 insertion(+)
 create mode 100644 Git/rebasetestfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git log --oneline
2f44d41 (HEAD -> dev) Added rebase test file.
4bdd96 (main) Added new file
3c5e488 (origin/main) Add Day-02.txt file
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  dev  git checkout main
Switched to branch 'main'
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  ls
ewfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git rebase dev
Successfully rebased and updated refs/heads/main.
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  ls
ewfile.txt         rebasetestfile.txt
 esteban@Estebans-MacBook-Pro  ~/Documents/GIT-Test/Git  main  git log --oneline
2f44d41 (HEAD -> main, dev) Added rebase test file.
4bdd96 Added new file
3c5e488 (origin/main) Add Day-02.txt file
```

Merge preserves the branch structure and creates a new merge commit, while **rebase** rewrites the commit history and provides a linear sequence of commits. The choice between merge and rebase depends on the specific use case, project requirements, and collaboration workflow.

Congratulations on completing Day 10 of the #90DaysOfDevOps challenge. Today, we explored advanced Git techniques, including branching, merging, and reverting. These techniques play a crucial role in efficient collaboration and version control in software development projects.

Stay tuned for Day 11, where we will continue exploring Git and GitHub for DevOps Engineers in the second part of this topic.