



INTRODUCTION TO GIT & GITHUB

by Bobby Ilier



Table of Contents

About the book	6
About the author	7
Sponsors	8
Ebook PDF Generation Tool	10
Ebook ePub Generation Tool	11
Book Cover	12
License	13
 Introduction to Git	 14
 Version Control	 16
 Installing Git	 18
 Basic Shell Commands	 21
 Git Configuration	 25
 Introduction to GitHub	 29
GitHub Stars	33
 Initializing a Git project	 35
 Git Status	 37
 Git Add	 39

Git Commit	41
Signing Commits	43
Git Diff	48
Git Log	51
Gitignore	54
SSH Keys	64
Git Push	68
Creating and Linking a Remote Repository	69
Pushing Commits	70
Checking the Remote Repository	72
Git Pull	73
Git Branches	77
Git Merge	85
Reverting changes	92
Resetting Changes (⚠ Resetting Is Dangerous ⚠)	93
Git Clone	97
Forking in Git	99

Output:

```
main
* newFeature
```

We can tell that we have 2 branches: the `main` one and the `newFeature` one that we just created. The star before the `newFeature` branch name indicates that we are currently on the `newFeature` branch.

If you were to use the `git checkout` command to switch to the `main` branch:

```
git checkout main
```

And then run `git branch` again. You will see the following output indicating that you are now on the `main` branch:

```
* main
  newFeature
```

Making changes to the new branch

Now let's go ahead and make a change on the new feature branch. First switch to the branch with the `git checkout` command:

```
git checkout newFeature
```

Note: we only need to add the `-b` argument when creating new branches

Check that you've actually switched to the correct branch:

```
git branch
```

Output:

```
main  
* newFeature
```

Now let's create a new file with some demo content. You can do that with the following command:

```
echo "<h1>My First Feature Branch</h1>" > feature1.html
```

The above will echo out the `<h1>My First Feature Branch</h1>` string and store it in a new file called `feature1.html`.

After that, stage the file and commit the change:

```
git add feature1.html  
git commit -m "Add feature1.html"
```

The new `feature1.html` file will only be present on the `newFeature` branch. If you were to switch to the `main` branch and run the `ls` command or check the `git log`, you will be able to see that the file is not there.

You can check that by using the `git log` command:

```
git log
```

With that, we've used quite a bit of the commands that we've covered in the previous chapters!

Compare branches

You can also compare two branches with the following commands.

- Shows the commits on **branchA** that are not on **branchB**:

```
git log BranchA..BranchB
```

- Shows the difference of what is in **branchA** but not in **branchB**:

```
git diff BranchB...BranchA
```

Renaming a branch

In case that you've created a branch with the wrong name or if you think that the name could be improved as it is not descriptive enough, you can rename a branch by running the following command:

```
git branch -m wrong-branch-name correct-branch-name
```

If you want to rename your **current branch**, you could just run the following:

```
git branch -m my-branch-name
```

After that, if you run **git branch** again you will be able to see the correct branch name.

Deleting a branch

If you wanted to completely delete a specific branch you could run the following command:

```
git branch -d name_of_the_branch
```

This would only delete the branch from your local repository, in case that you've already pushed the branch to GitHub, you can use the following command to delete the remote branch:

```
git push origin --delete name_of_the_branch
```

If you wanted to synchronize your local branches with the remote branches you could run the following command:

```
git fetch
```

Conclusion

With that, our **newFeature** branch is now ahead of the main branch with 1 commit. So in order to get that new changes over to the main branch, we need to merge the **newFeature** branch into our **main** branch.

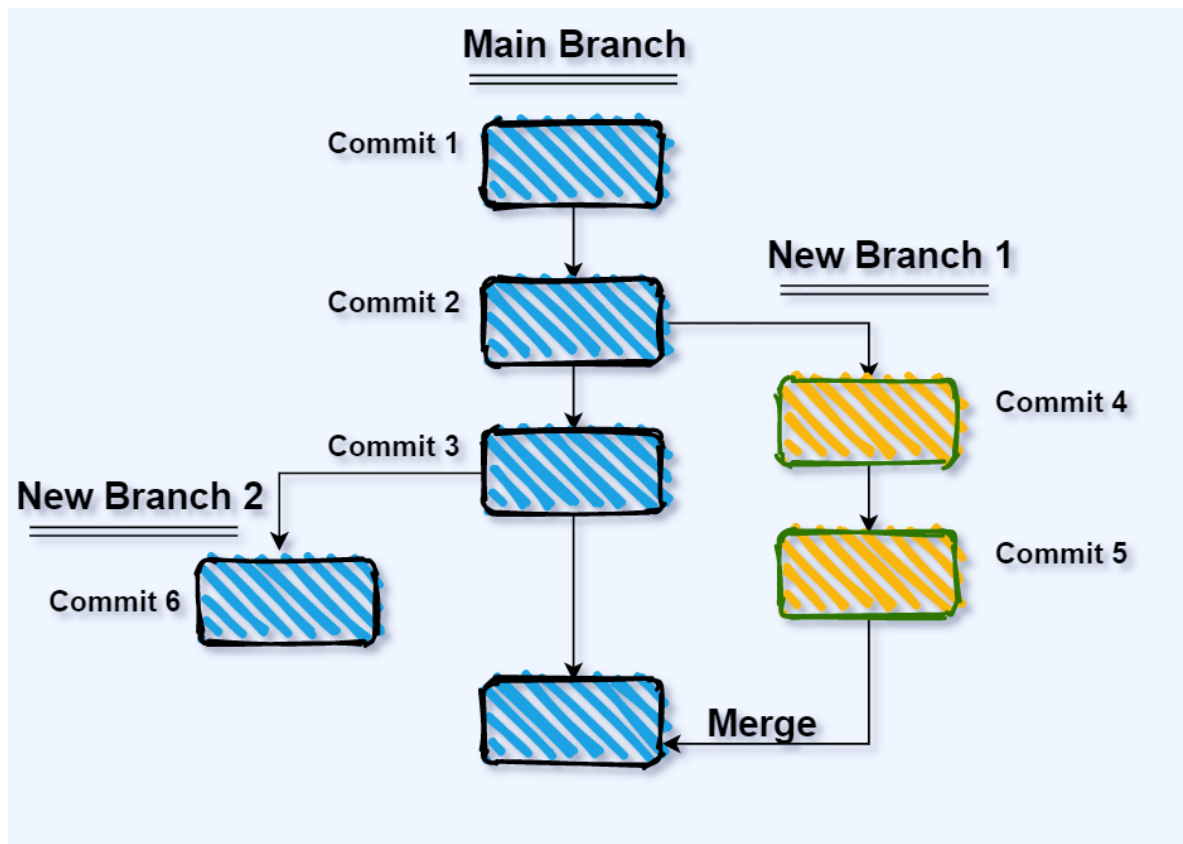
In the next chapter, you will learn how to merge your changes from one branch to another!

One thing that you might want to keep in mind is that in the past when creating a new GitHub repository the default branch name was called **master**. However, new repositories created on GitHub use **main** instead of **master** as the default branch name. This is part of GitHub's effort to

remove unnecessary references to slavery and replace them with more inclusive terms.

Git Merge

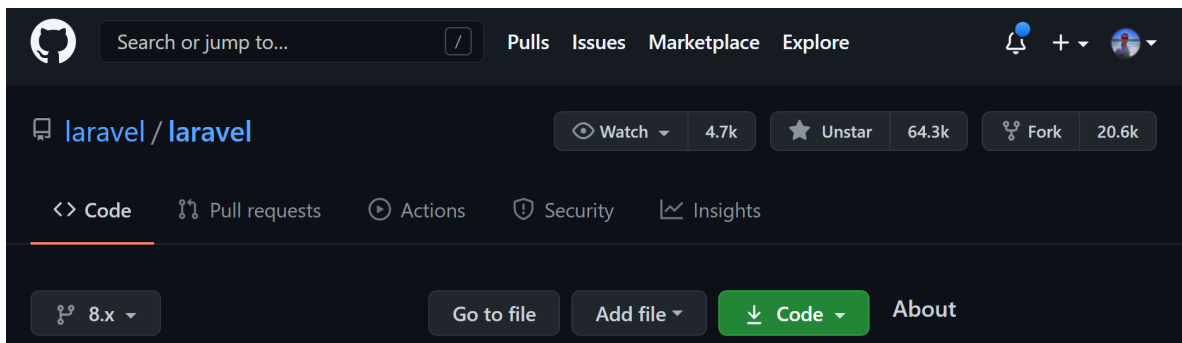
Once the developers are ready with their changes, they can merge their feature branches into the main branch and make those features live on the website.



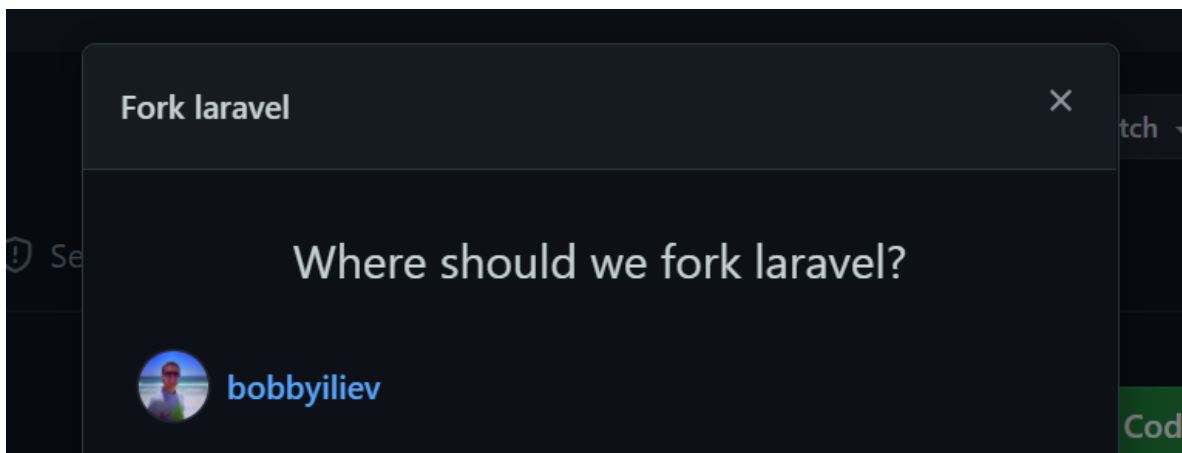
If you followed the steps from the previous chapter, then your **newFeature** branch is now ahead of the main branch with 1 commit. So in order to get that new changes over to the main branch, we need to merge the **newFeature** branch into our **main** branch.

Merging a branch

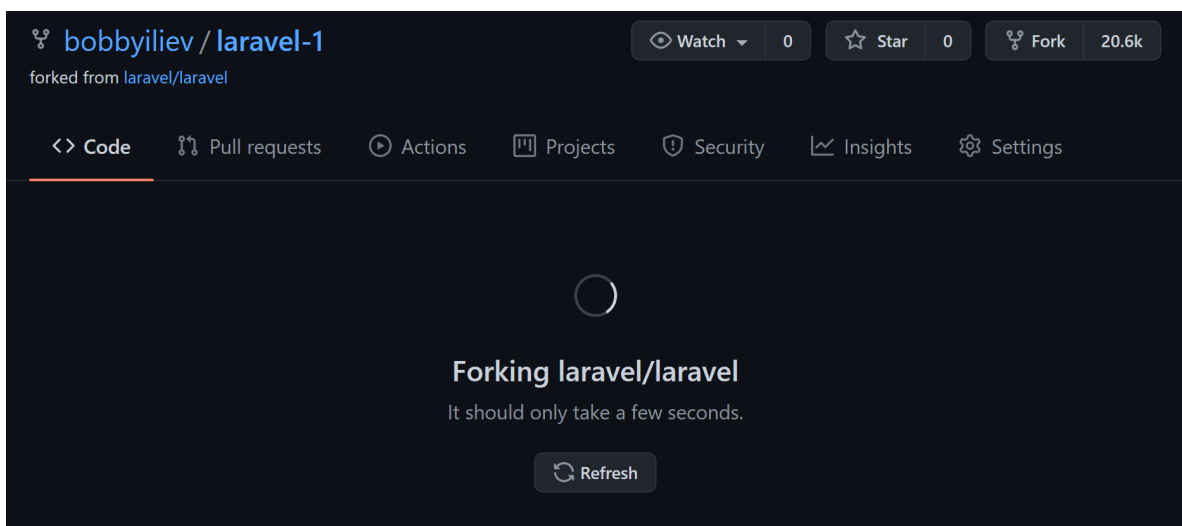
You can do that by following these steps:



Then choose the account that you want to fork the repository to:



Then it might take a few seconds for the repository to be forked under your account:



With that, you would have an exact copy of the repository in question under your account.

The benefit here is that you can now clone the forked repository under your account, make the changes to that repository as normal, and then once you are ready, you can submit a pull request to the original repository contributing your changes.

As we've now mentioned submitting pull requests a few times already, let's go ahead and learn more about pull requests in the next chapter!

Git Workflow

Now that you know the basic commands, let's put it all together and go through a basic Git workflow.

Usually, the workflow looks something like this:

- First, you clone an existing project with the `git clone` command, or if you are starting a new project, you initialize it with the `git init` command.
- After that, before starting with your code changes, it's best to create a new Git branch where you would work on. You can do that with the `git checkout -b YOUR_BRANCH_NAME` command.
- Once you have your branch ready, you would start making the changes to your code.
- Then, once you are ready with the changes, you need to stage them with the `git add` command.
- Then, to commit/save the changes to your local Git repository, you need to run the `git commit` command and provide a descriptive commit message.
- To push your local changes to your remote GitHub project, you would use the `git push origin YOUR_BRANCH_NAME` command

- Finally, once you've pushed your changes, you would need to submit a pull request (PR) from your branch to the main branch of the repository.
- It is considered good practice to add a couple of people as reviewers and ask them to review the changes.
- Finally, once the changes have been approved, the PR would get merged into the main branch taking all of your changes from your branch into the main branch.

The overall process will look like this:



My advice is to create a new repository and go over this process a few times until you feel completely comfortable with all of the commands.

This is a sample from "Introduction to Git and GitHub" by Bobby Iliev.

For more information, [Click here](#).