A Beginner's Guide to Implementing New Features in Git

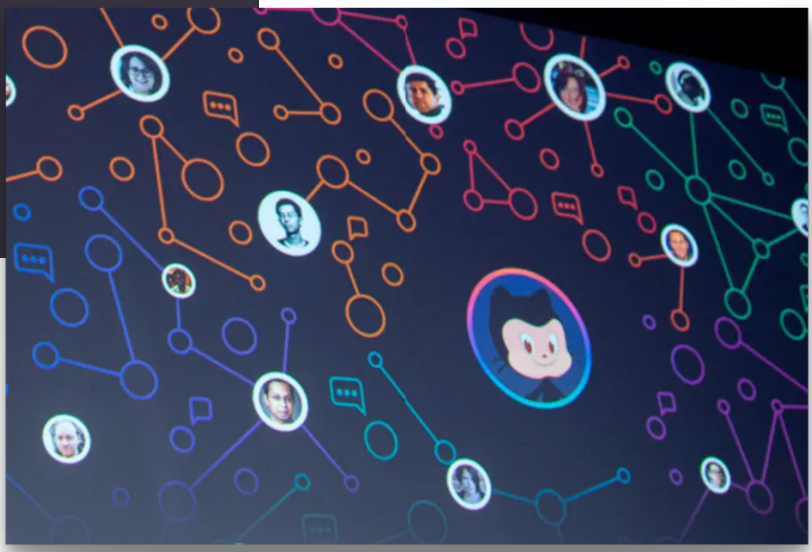
Without Touching the Main Branch

Learning made fun.

Noe Tovar-MBA

A Beginner's Guide to Implementing New Features in Git Without Touching the Main Branch

PDF is available for download
scan below



Getting Started

Elevate your development game:

Hey there, fellow coding enthusiasts! Today, I'm thrilled to guide you through a fundamental aspect of Git that can elevate your development game: implementing new features or changes without causing chaos in the main branch. You see, Git is like a superhero tool for version control, and mastering its abilities can make your coding life a whole lot easier.

Let's dive right in by understanding the "why" behind keeping the main branch clean and untouched. Imagine the main branch as the heart of your project, holding the stable and approved code. Making changes directly in this branch can disrupt ongoing work or even introduce bugs, causing headaches for you and your team. Instead, we'll use branches to isolate our new changes, test them, and then smoothly integrate them into the main branch when they're ready.

Step 1.

With a passion for learning

So, the first step on our journey is creating a new branch where we'll work on our fantastic new feature or fix. In Git lingo, a branch is like a separate path where you can experiment without affecting the main flow of the project. To create a new branch, we'll use the git checkout command with the -b option followed by the branch name. Here's an example:

```bash
git checkout -b my-new-feature
```

This command creates a new branch named my-new-feature and switches us to that branch simultaneously. Now, we're all set to work our coding magic without worrying about messing up the main branch.

Next up, let's make those awesome changes or add that incredible feature in this newly created branch. Modify your files, write new code, or do whatever it takes to bring your innovation to life. Git will track these changes within the specific branch, keeping everything neatly organized.

Testing is Crutial!

Once our changes are ready and tested within our branch (testing is crucial, trust me), it's time to prepare them for integration into the main branch. But before that, we need to commit our changes. A commit in Git is like a snapshot of your code at a particular moment. To commit changes, we'll use a combination of commands:

```bash
git add . # This stages all changes for commit
git commit -m "Descriptive message about the changes made"
```

The **git add** . command stages all the changes you've made, preparing them for the commit.
The **git commit -m "Descriptive message"** command then captures those changes along with a descriptive message explaining what was done in this commit.
Now, we're on the brink of integrating our fantastic changes into the main branch. To do this, we'll switch back to the main branch using **git checkout main** and then merge our feature branch into it using the **git merge** command:

```bash
git checkout main # Switching back to the main branch
git merge my-new-feature # Merging our changes from 'my-new-feature' in
```

The git merge command combines the changes from our feature branch (my-new-feature) into the main branch. Git cleverly incorporates these changes while keeping everything smooth and organized.

Finally, our mission is complete!

We've implemented our new feature without causing any disruptions to the main branch. And guess what? You're now ready to embark on more Git adventures, branching out, experimenting fearlessly, and contributing to your projects with confidence.
Remember, Git empowers you to innovate and collaborate seamlessly. So, go ahead, create, experiment, and let Git be your trusty sidekick in the realm of coding!
Until next time, happy coding!

Places where beginners can learn more basics of GitHub:

1. **GitHub Learning Lab**: GitHub offers its own Learning Lab that provides interactive courses on various topics, including an introduction to Git and GitHub basics. It offers hands-on experiences to help beginners understand the platform.

2. **Git Handbook and Documentation**: GitHub's own documentation, as well as resources like the "Pro Git" book by Scott Chacon and Ben Straub, provide comprehensive guides to understanding Git and GitHub basics.

These resources can serve as excellent references where beginners can gain a solid understanding of Git and GitHub basics, enabling them to start using these tools confidently.