**Introduction to Git-flow**

Git-flow is a workflow that helps with continuous software development and implementing DevOps practices. It was first published and made popular by Vincent Driessen at nvie. The Git-flow Workflow defines a strict branching model designed around the project release.

Diagram

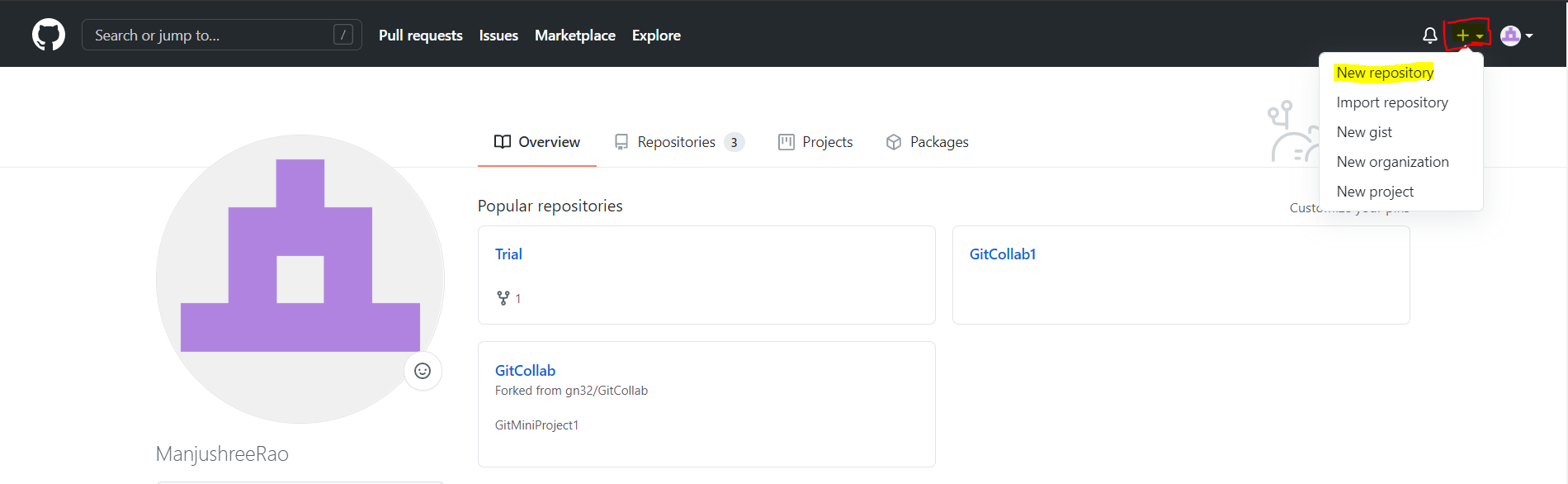
Description automatically generated

**GIT Terminology & Commands**

**1. Repository**

A repository is a storage space. It contains all the project related files and their file's revision history. Repository can also be used to discuss, manage and track project's work. Repositories can be owned individually or can be shared ownership of repositories with other people in an organization.

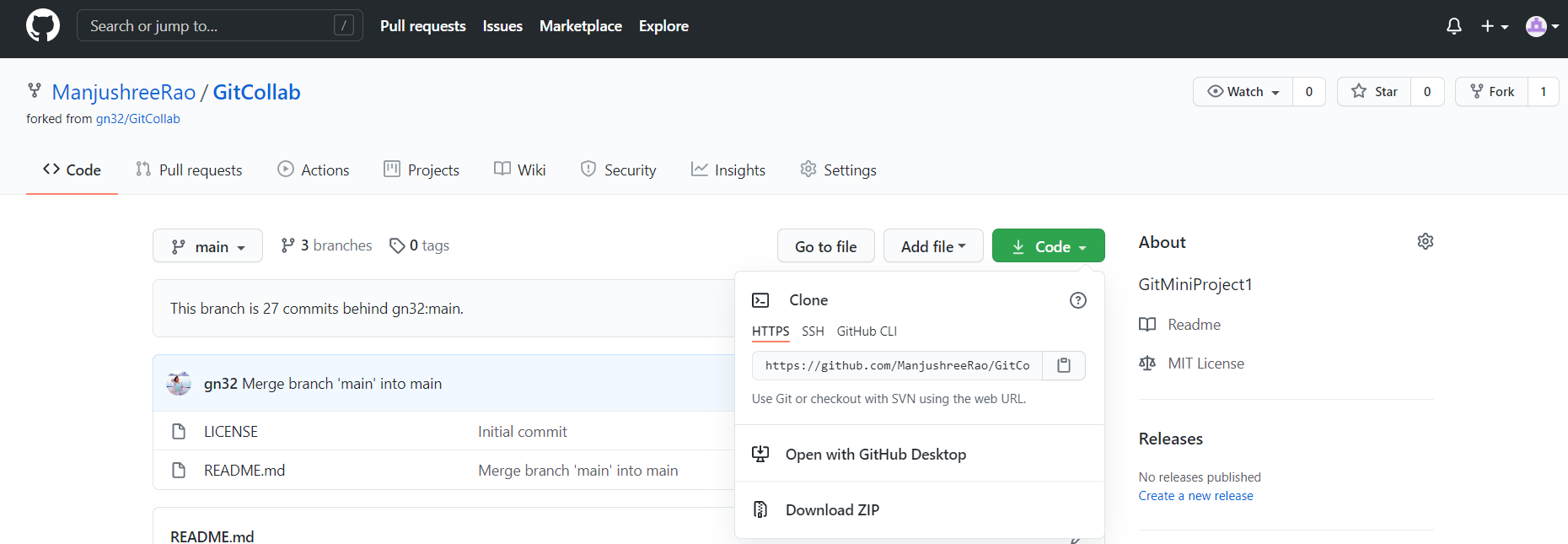
Command: In the upper-right corner of any page, use the drop-down menu, and select New repository.



**2. Clone**

Cloning a repository means obtaining a copy of all the File and folder of the project in that repository at that point of time, from GitHub to our local system. This includes the version related files too.

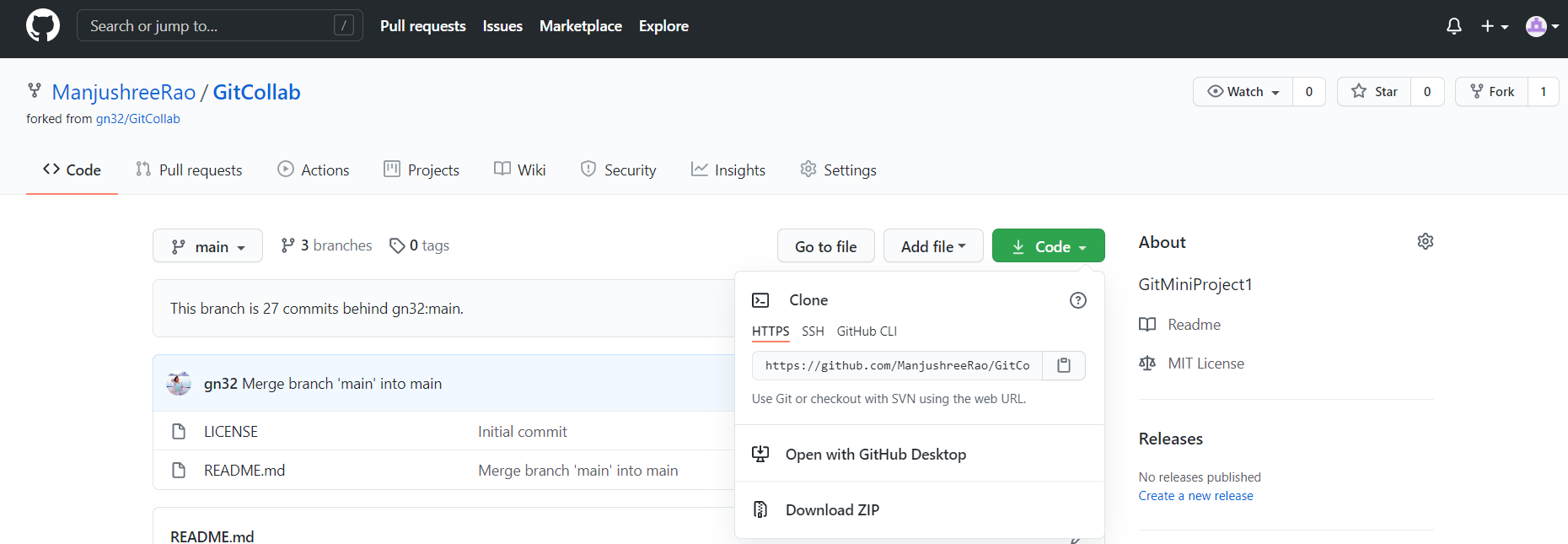
Command: From your repository page on GitHub, click the green button labeled Clone or download, and in the “Clone with HTTPs” section, copy the URL for your repository. Next, on your local machine, open your bash shell and change your current working directory to the location where you would like to clone your repository.



**3. Fork**

Creating a fork is producing a personal copy of someone else's project, which means when you fork the copy of the repository you have taken remaining in you GitHub account only. Forks act as a sort of bridge between the original repository and your personal copy.

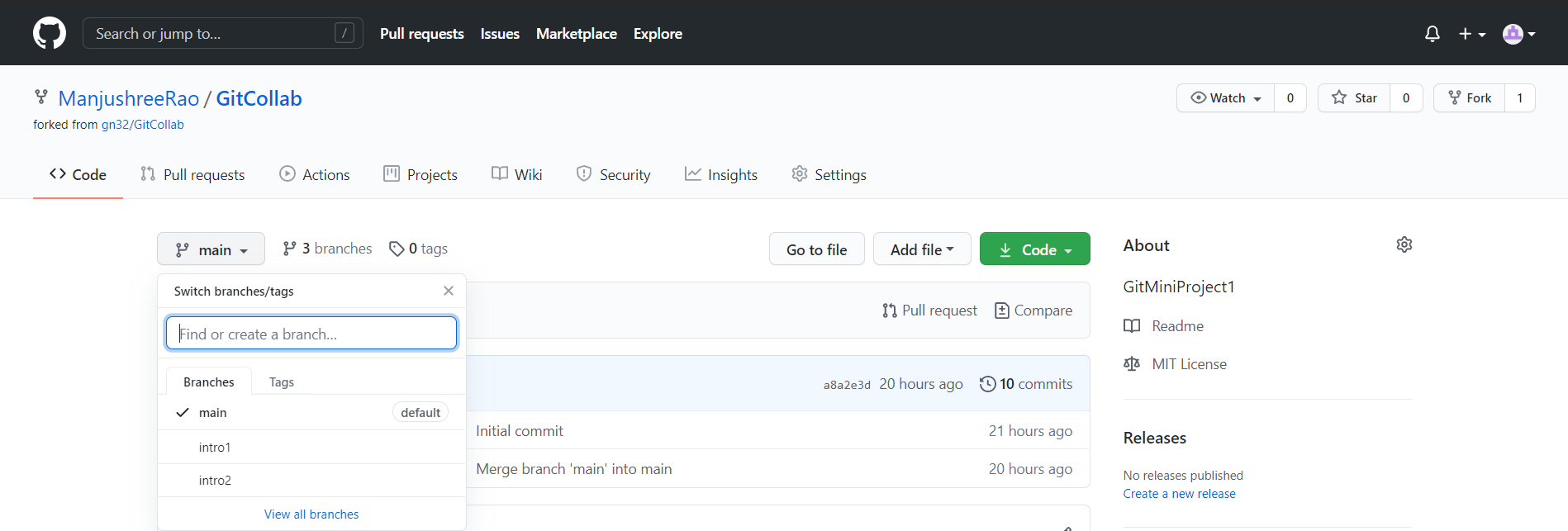
Command: You can fork any repo by clicking the fork button in the upper right-hand corner of a repo page. Click on the Fork button to fork any repo on github.com.



**4. Branch**

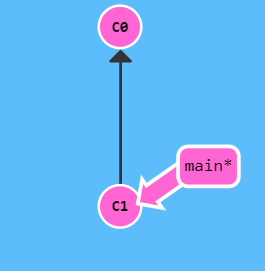
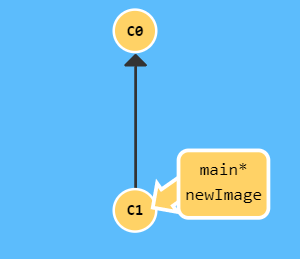
A branch represents an independent line of development. The git branch command lets you create, list, rename, and delete branches. It does not let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

Method 1: Creating a branch On GitHub, navigate to the main page of the repository. Click the branch selector menu. Type a unique name for your new branch, then select Create branch.



Method 2: You can do this with

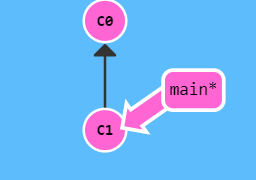
***$ git branch <branch name>***

 **🡪** *Note: newImage is the new branch created*

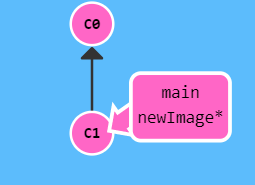
Method 3: To create the branch from your current branch and it will switch you to that branch in a single command.

You can do this with

***$ git checkout -b <branch-name>***



🡫

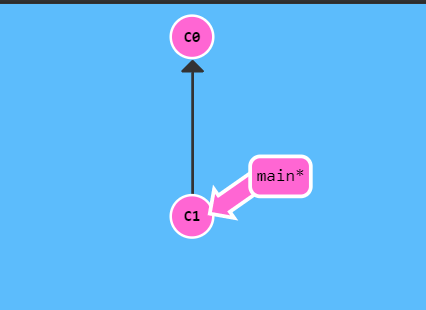


**5. Commit**

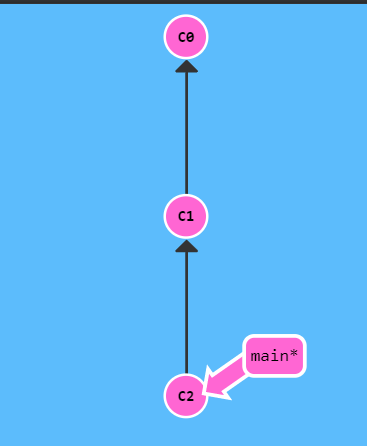
The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as safe versions of a project. Git will never change them unless you explicitly ask it to. Commits include lots of metadata in addition to the contents and message, like the author, timestamp, and more.

You can do this with

***$ git commit***



🡫

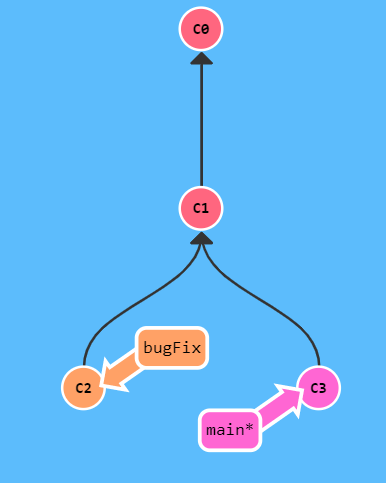
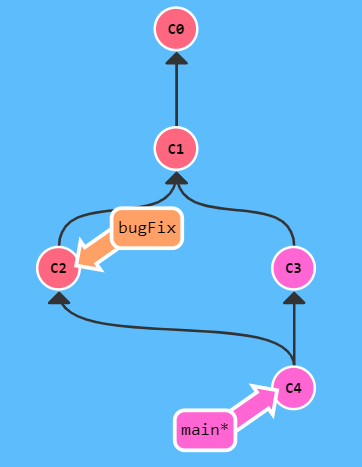


**6. Merge**

Merge is a way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch. ... The current branch will be updated to reflect the merge, but the target branch will be completely unaffected.

You can do this with

***$ git merge <branch\_name>***

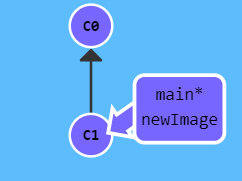
 🡪 

**7. Checkout**

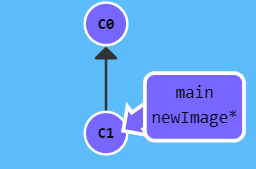
Checkout is the act of switching between different versions of a target entity. The git checkout command operates upon three distinct entities: files, commits, and branches.

You can do this with

***$ git checkout <branch\_name>***



🡫

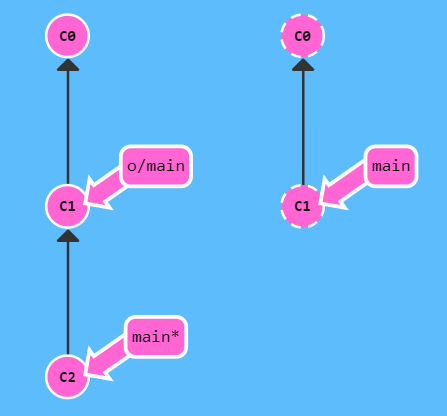


**8. Push**

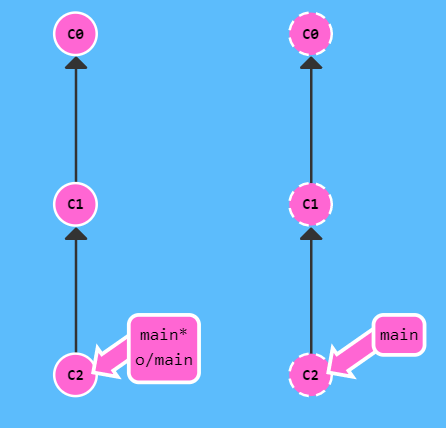
The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch, but whereas fetching imports commits to local branches, pushing exports commits to remote branches.

You can do this with

***$ git push***



**🡫**

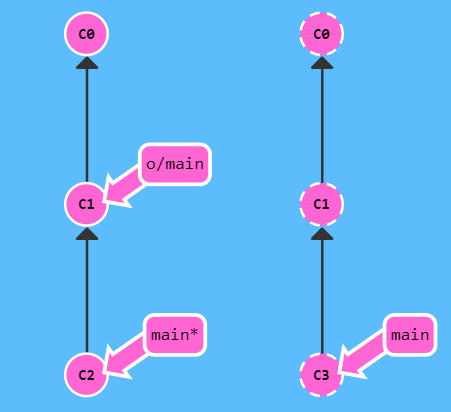


**9. Pull**

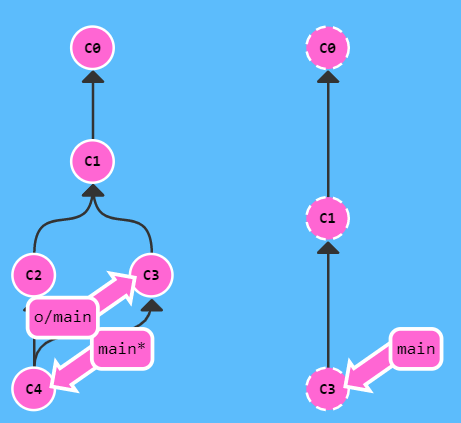
The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. ... Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be be-created, and HEAD updated to point at the new commit.

You can do this with

***$ git pull***



🡫



**10. Remote Add / Remove / Show**

The git remote command lets you create, view, and delete connections to other repositories. Remote connections are more like bookmarks rather than direct links into other repositories. Instead of providing real-time access to another repository, they serve as convenient names that can be used to reference a not-so-convenient URL.

You can do this with

***$ git remote add origin <branch\_url>. git***

**11. Status**

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which have not, and which files are not being tracked by Git. Status output does not show you any information regarding the committed project history.

You can do this with

***$ git status***

**12. Master branch**

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you are given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.