To begin with, when you are forking in git, you are using a command specific to GitHub. This is not something you do on your local computer, in git repositories, it is something you do specifically on GitHub. To fork means to "Make a copy of someone's code so that I can add my own modifications". You are making a copy of their repository, and putting it under your account. This is now a repository to which you can push changes, over which you have full control.

Cloning, on the other hand, is taking a copy of someone else's code, and copying it down to your local computer. Unless you have been added as a collaborator, you will not be able to push changes up to the remote repository, but you will be able to continue to pull changes down to your computer.

Primarily, the difference between forking and cloning is about permissions. Forking makes a copy of someone's repository in your GitHub account. ANY repository on GitHub can be cloned to your local computer, whether it's yours or someone else's repository. However, the only repositories to which you can push changes to are ones which you either a) own, or b) to which you have been given permissions.

Points to raise and understand at this stage

What does it mean to fork a repo

To make a copy of someones code so I can make my own modifications to it.

s forking a git thing or a GitHub thing?

Git hub thing

what is the difference between git and GitHub?

Git hub is a hosting service which you can upload a copy of your git repositories.

What does it mean to clone a repository? How is it different to forking?

Cloning makes a copy of a repo on to your computer but you cant make changes to it

* Is cloning a git thing or a GitHub thing?

Cloning is a git thing because you can pull it down but you cant make changes and push it back up,

* Why does Developer A have to give Developer B access to their repo? Especially since the repo is already public.

So they can make changes to it then push it back up to the git hub with these changes.

* What is the difference between a public and private repo on GitHub?

Public is just for everybody. Something private (new idea) should be kept private.

**Question 1:** During this linear process, where Dev B changed the document after Dev A had already committed and pushed their change to the remote (GitHub), what extra step(s) could Developer B have taken to completely avoid this merge conflict?

Pulled before committing changes

**Question 2:** Given your answer in Q1, does this mean that merge conflicts can be completely (always) avoided in the real world? Why or why not?

In the real world no, because you never know when developer a will make changes.

A commit says – to keep track of all modifications – youre not saving the whole file, youre only keeping track of the changes!

Commands

Git in it 🡪 creates a new folder .git

Example from day 2 where it asked you to clone max.rb. When you clone someone elses code, were not starting at point 0, were starting at some point on someone elses time axis. Where that person saved it.

Ls –la 🡪 the l tells it to give it as a vertical list, and a part of the argument says all.

When asking git status (shows what exactly has changed in the file), and the modified: filename.rb is red means it hasn’t been put on the stage yet.

Git commit 🡪 will take a msg that says what you did between the clone and the commit.

Always do git commit –m “Comment” so that you don’t have to do it seperately.

Lets say someone says theres a problem with that commit 🡪 then make some changes and commit again and add a note saying about what you fixed.

Now even though you are committing, it is still on your local machine. So send it to github with 🡪 git push

Git diff – prints out the changes that have been made with + … or - …. This is before commiting it because when we commit it we start a new modified file.