git and GitHub

What is git?

Git is a version control system. There are other version control systems, but git is the most widely used.

Version control software keeps track of modifications to the code. It is critical in enabling development teams to easily work in the same code base.

Git is installed locally on your machine.

What is GitHub?

GitHub is a cloud based service that hosts git repositories. GitHub is not the only option for hosting git repositories, but it is widely used.

Using git

You can use git through a command line interface or through a git client.

git Cheat Sheet
https://education.github.com/git-cheat-sheet-education.pdf

A glossary of the basic git terms
https://www.pluralsight.com/resources/blog/cloud/git-terms-explained

Branch

A version of the repository that diverges from the main working project. Branches can be a new version of a repository, experimental changes, or personal forks of a repository for users to alter and test changes.

Git Checkout

The git checkout command is used to switch branches in a repository. git checkout testing-el would take you to the *testing-el* branch whereas git checkout master would drop you back into master. Be careful with your staged files and commits when switching between branches.

Clone

A clone is a copy of a repository or the action of copying a repository. When cloning a repository into another branch, the new branch becomes a remote-tracking branch that can talk upstream to its origin branch (via pushes, pulls, and fetches).

Fork

Creates a copy of a repository.

Merge

Taking the changes from one branch and adding them into another (traditionally master) branch. These commits are usually first requested via pull request before being merged by a project maintainer.

Pull/Pull Request

If someone has changed code on a separate branch of a project and wants it to be reviewed to add to the master branch, that someone can put in a pull request. Pull requests ask the repo maintainers to review the commits made, and then, if acceptable, merge the changes upstream. A pull happens when adding the changes to the master branch.

Push

Updates a remote branch with the commits made to the current branch. You are literally "pushing" your changes onto the remote.

Remote

A copy of the original branch. When you clone a branch, that new branch is a remote, or *clone*. Remotes can talk to the origin branch, as well as other remotes for the repository, to make communication between working branches easier.

Repository ("Repo")

In many ways, you can think of a Git repository as a directory that stores all the files, folders, and content needed for your project. What it actually is, is the object database of the project, storing everything from the files themselves, to the versions of those files, commits, deletions, et cetera. Repositories are not limited by user, and can be shared and copied (see: fork).

What is an IDE?

IDE stands for Integrated Development Environment. This is simply a software application that helps programmers develop software code efficiently. It makes code development easier by combining a variety of tools into a single application. Visual Studio Code is one example, but there are many others.

You can find a tutorial on using Visual Studio Code at the bottom of Lesson 6 here: https://classes.codethedream.org/course/intro-to-programming-v4/io?week=6&lesson=Introduction+to+Algorithms

Command Line Interface

A command line interface is simply a way of interacting with a computer by inputting lines of text. Most of what you currently do on a computer is through a GUI (or graphical user interface). However, there are things that can't always be accomplished through a GUI.

Command Line Basics can be found here:
https://www.theodinproject.com/lessons/foundations-command-line-basics

This describes using a mac or linux environment. On a windows machine, you can use git bash which is installed when you install git on your machine.

Command Line Cheat Sheet
https://www.git-tower.com/blog/command-line-cheat-sheet/

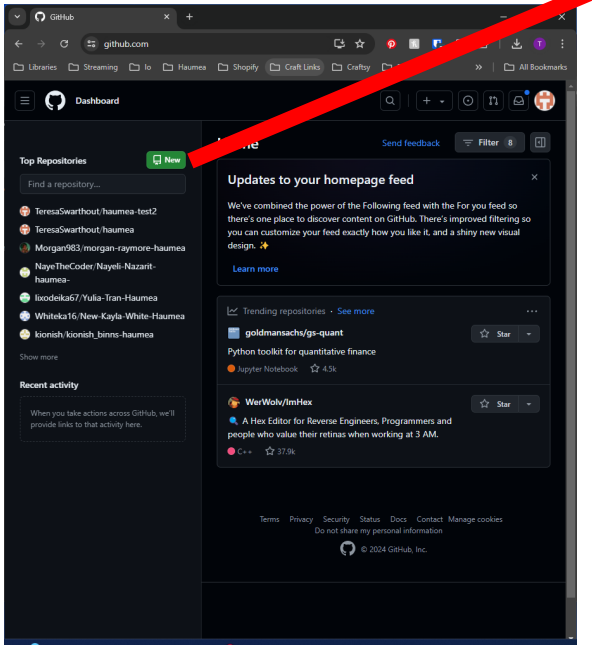
Getting Started

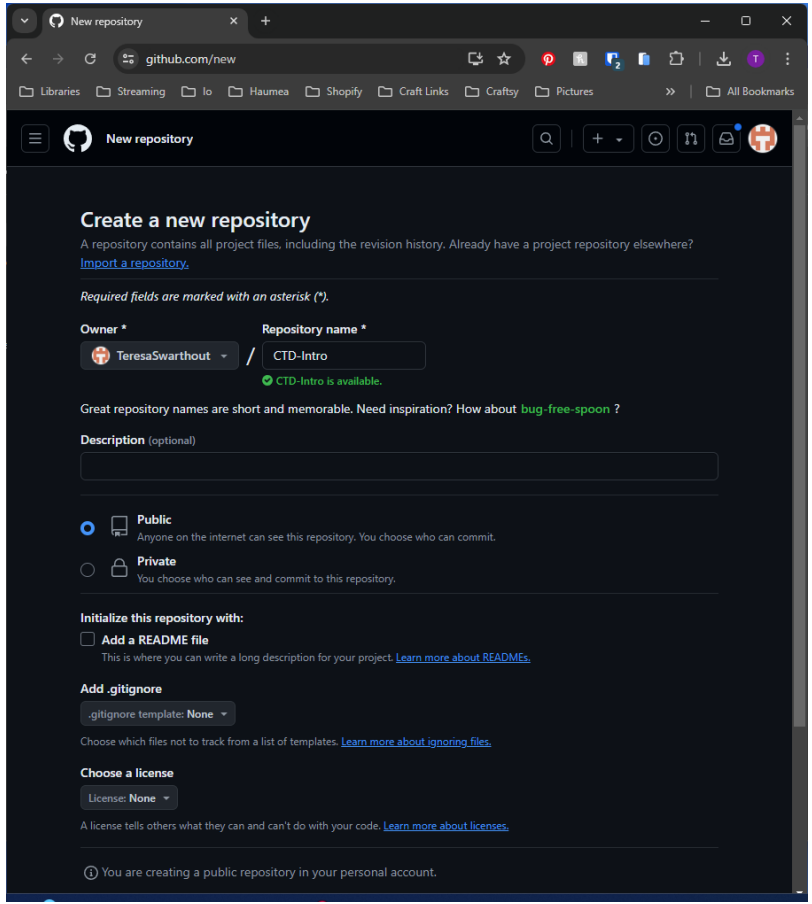
You will only need to do these things once:

- Install git on your local machine and configure it using the steps given here: https://www.theodinproject.com/lessons/foundations-setting-up-git

- Create a GitHub account

CREATE a new repository.

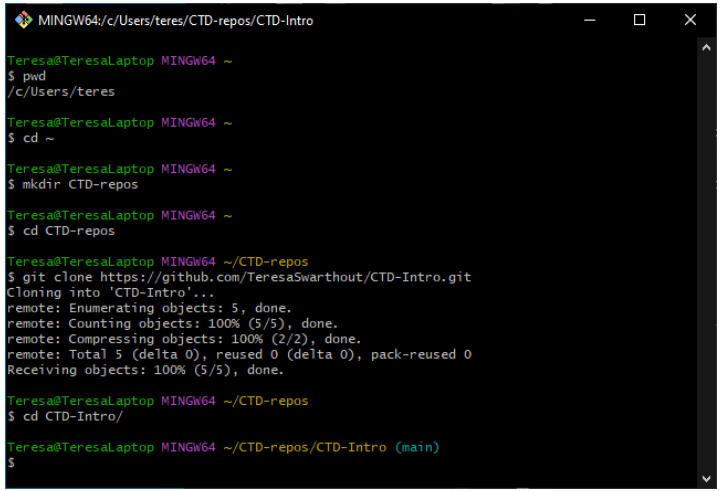


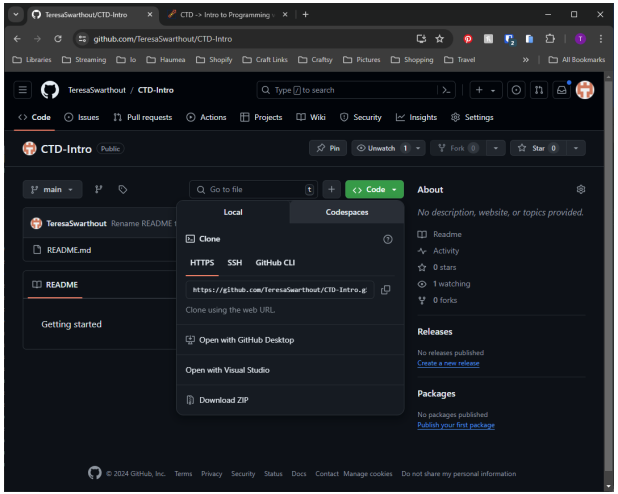


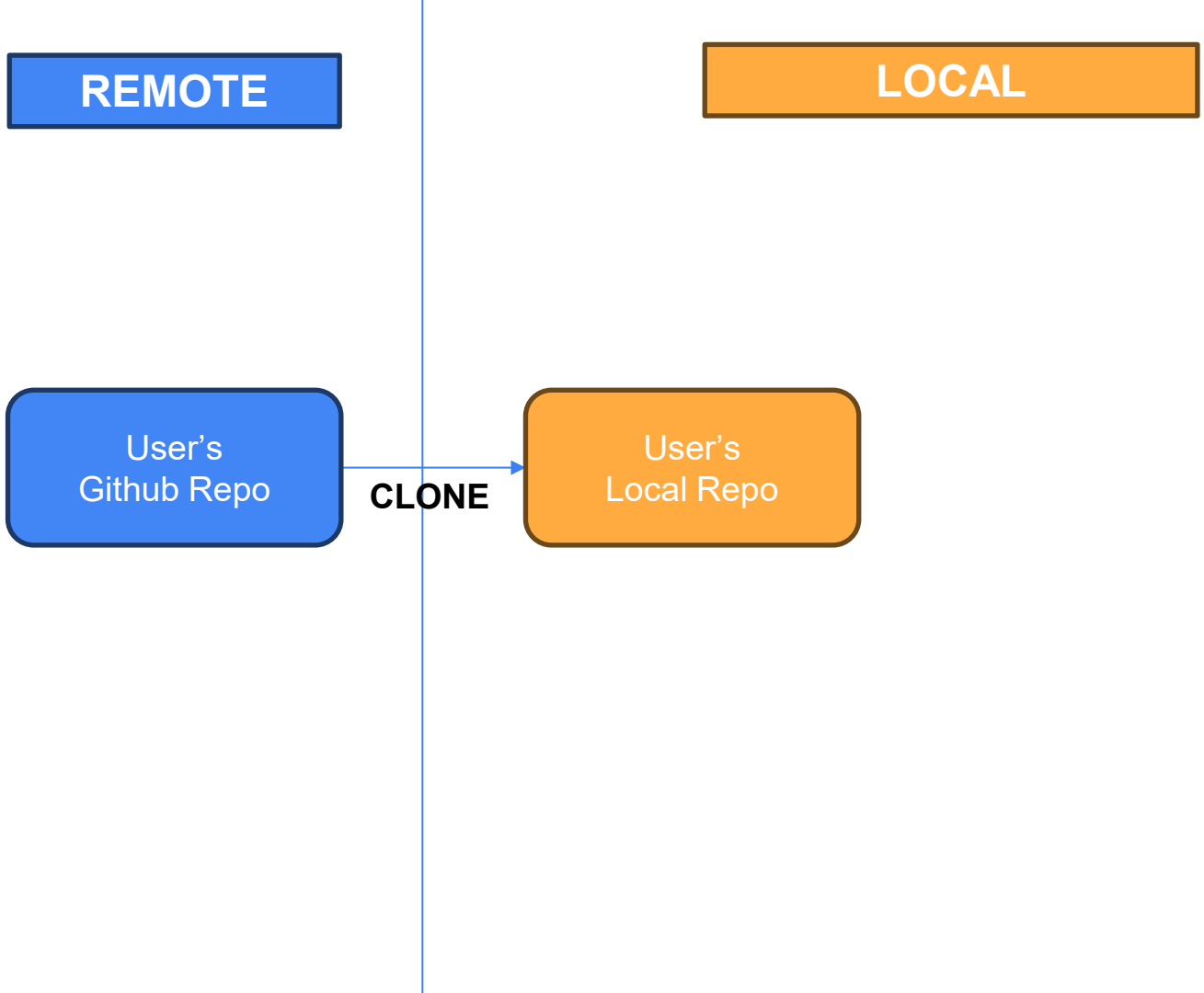
CLONE the copy of the repository to your local machine**.**
Switch to the directory you want to use then issue the command:

You can use the mkdir command to create directory (folder)
for your repositories.

cd ~
mkdir CTD-repos
cd CTD-repos
git clone https://github.com/TeresaSwarthout/CTD-Intro.git

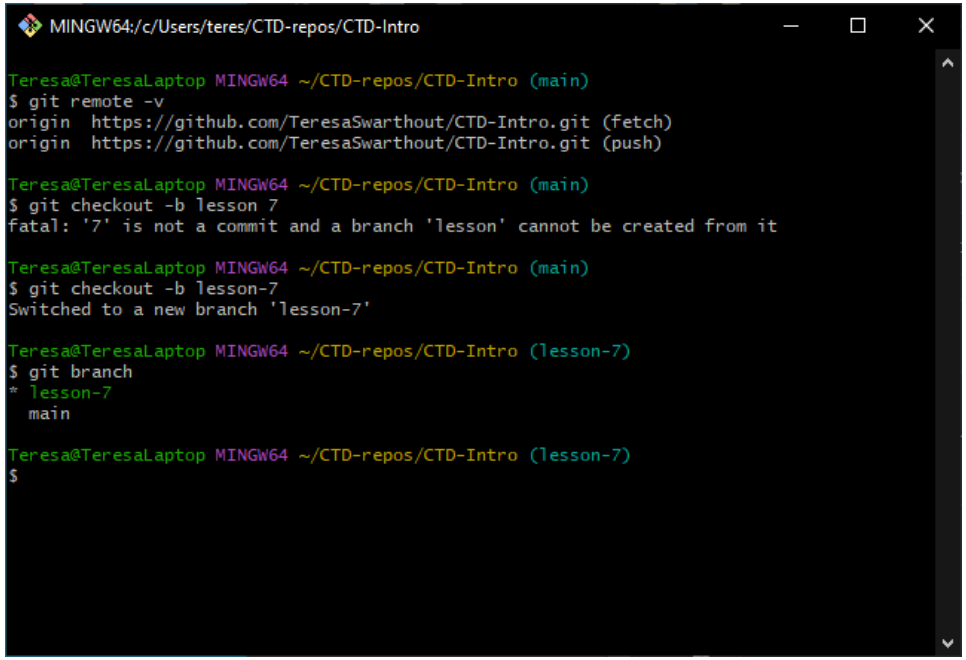


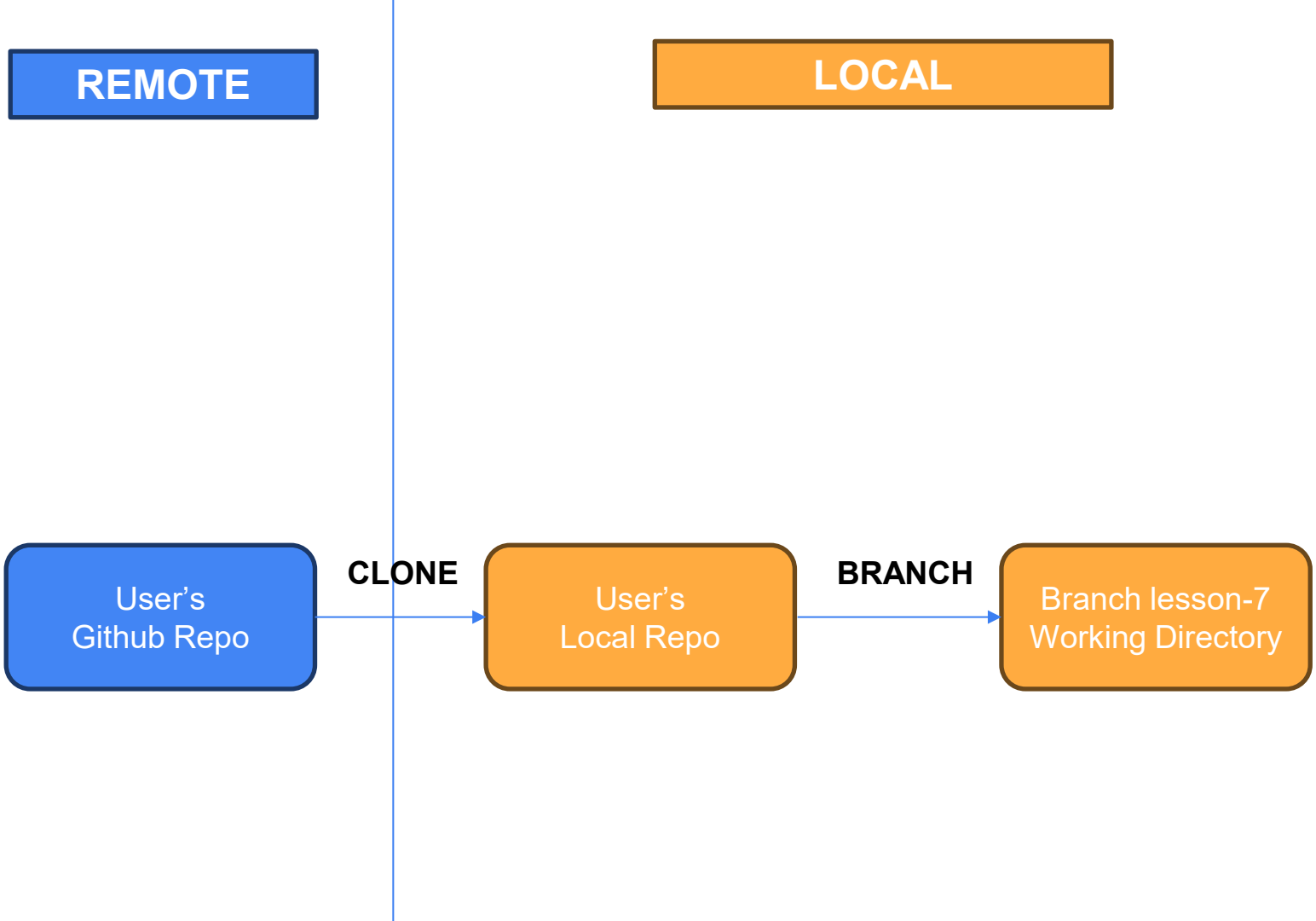



REMOTE
LOCAL
User's
Github Repo
CLONE
User's
Local Repo

Now we have a local repo set up and we are ready to work. However, we don't want to work in the main repo. We want to set up a branch for the new and exciting code we are working on, but might make many mistakes on as we go along and don't want to mess up all of the great code in our main repo along the way.

- Switch to the directory we created during the clone
  cd CTD-Intro
- Verify we're in the right place
  git remote –v
- And now create a branch for this lesson
  git checkout –b lesson-7
- Check you are on that branch
  git branch




REMOTE
LOCAL
User's
Github Repo
CLONE
User's
Local Repo
BRANCH
Branch lesson-7
Working Directory

Once we have made all the changes and feel good about the code in our branch, we want to commit it to our local repo.

- Check the status of your local repo and branch

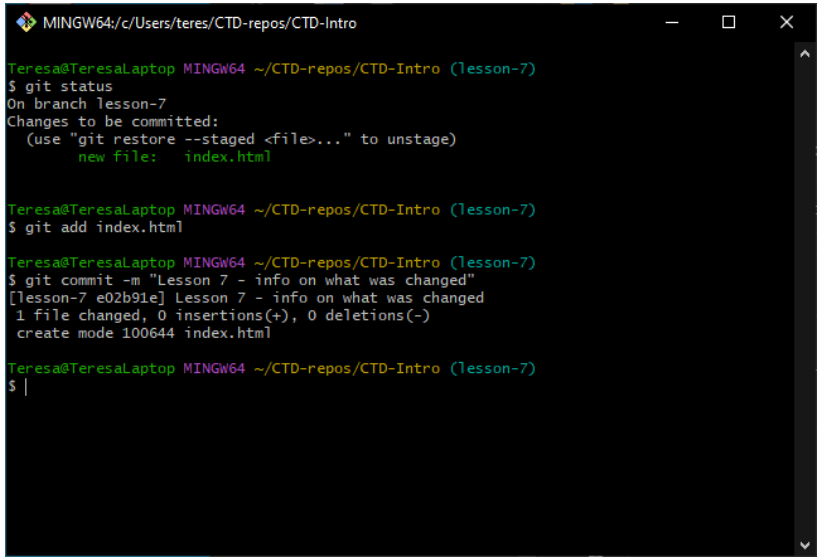
  git status

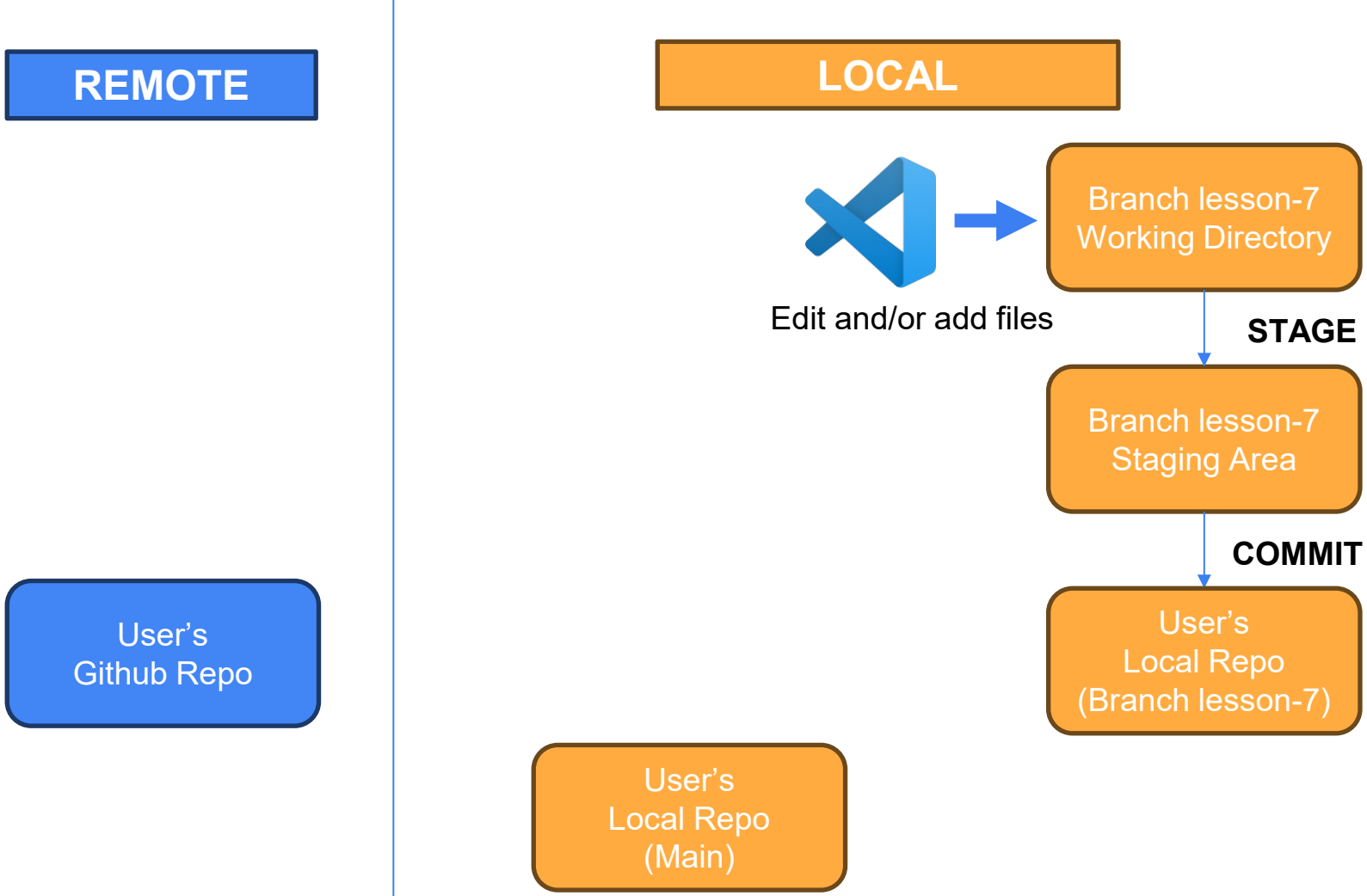
- Stage the file(s) that you edited. Staging just means that you are marking a file that you have modified as ready to go into your next commit.

  git add index.html

- Now, commit the files. Be sure and add a comment.

  git commit -m "Lesson 7 - info on what was changed"




REMOTE
LOCAL
Branch lesson-7
Working Directory
Edit and/or add files
STAGE
Branch lesson-7
Staging Area
COMMIT
User's
Local Repo
(Branch lesson-7)
User's
Github Repo
User's
Local Repo
(Main)

Now, we want to **PUSH** the changes from our local branch to our remote repo

git push

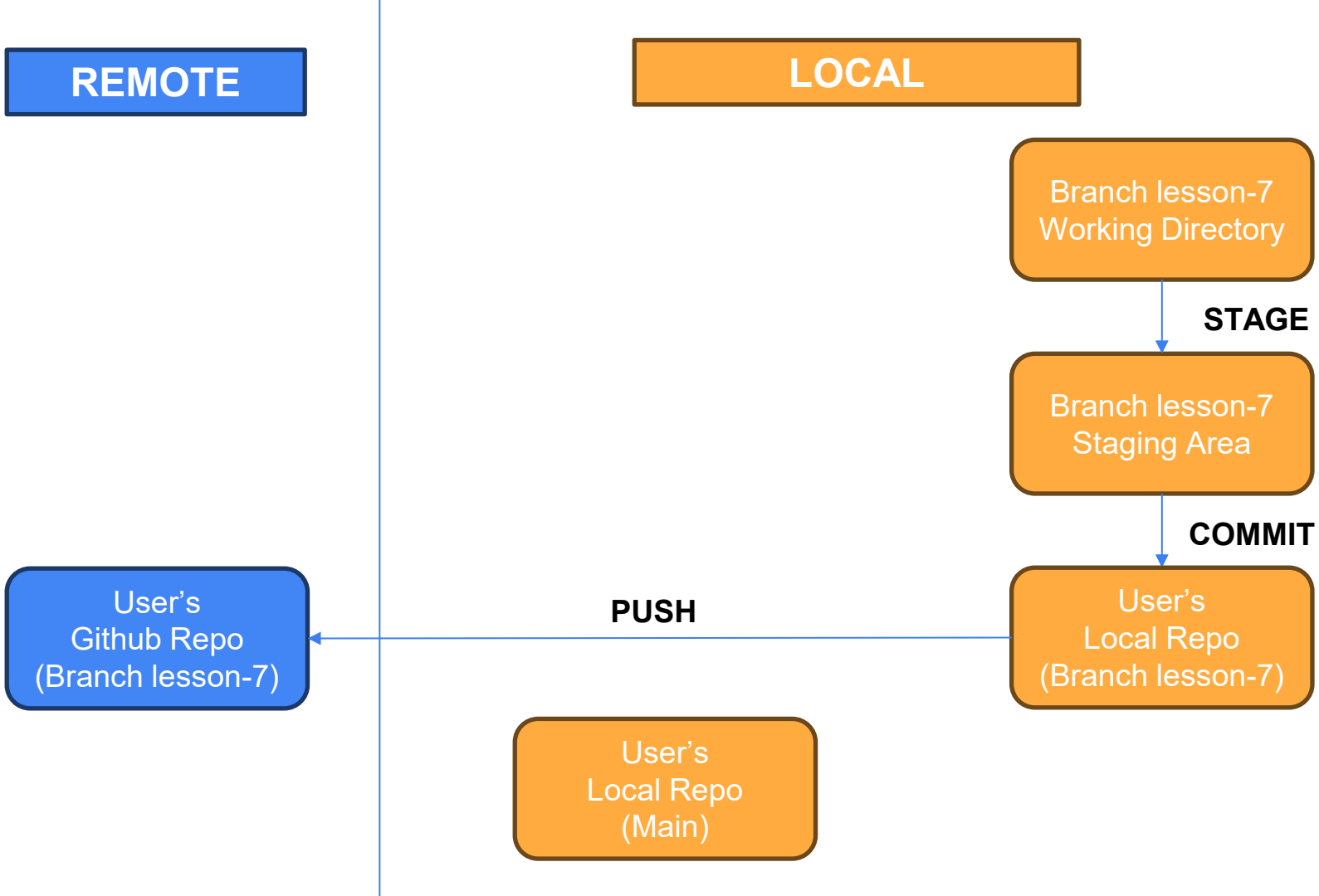
Now, you will probably see a message like this:

```
$ git push
fatal: The current branch lesson-2-1 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin lesson-2-1

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

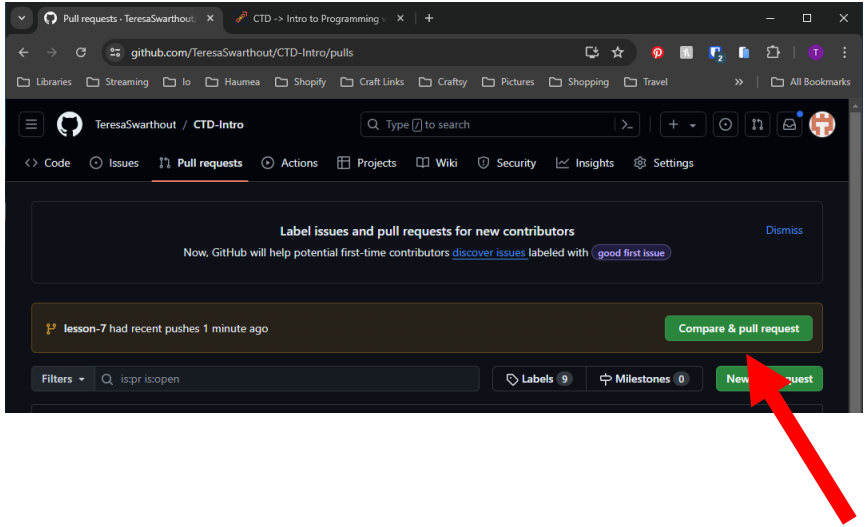
This is because you created the branch on your local repository and the remote repository has no knowledge of this branch. You just need to tell the remote repo about the branch.

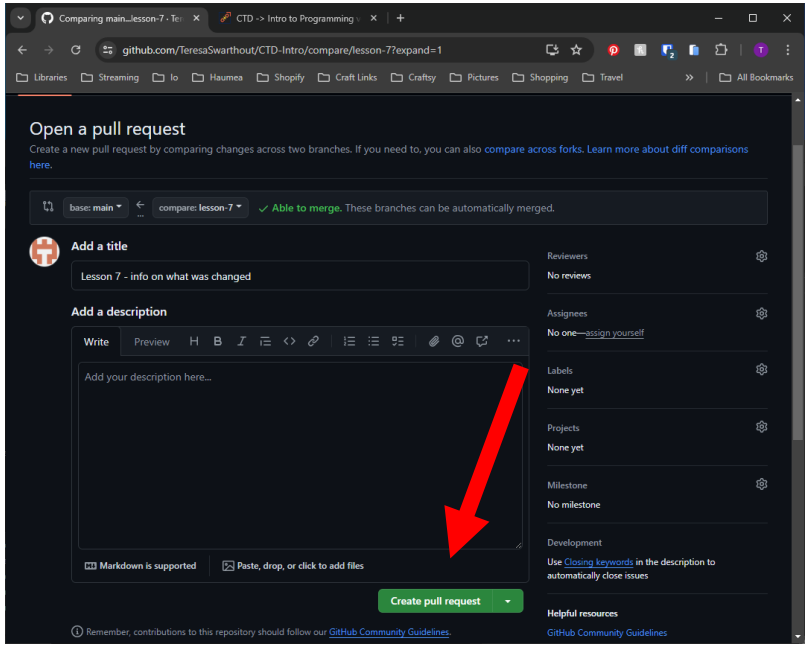
git push --set-upstream origin lesson-7

```
Teresa@TeresaLaptop MINGW64 ~/CTD-repos/CTD-Intro (lesson-7)
$ git push --set-upstream origin lesson-7
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'lesson-7' on GitHub by visiting:
remote:      https://github.com/TeresaSwarthout/CTD-Intro/pull/new/lesson-7
remote:
To https://github.com/TeresaSwarthout/CTD-Intro.git
 * [new branch]      lesson-7 -> lesson-7
branch 'lesson-7' set up to track 'origin/lesson-7'.

Teresa@TeresaLaptop MINGW64 ~/CTD-repos/CTD-Intro (lesson-7)
$ |
```

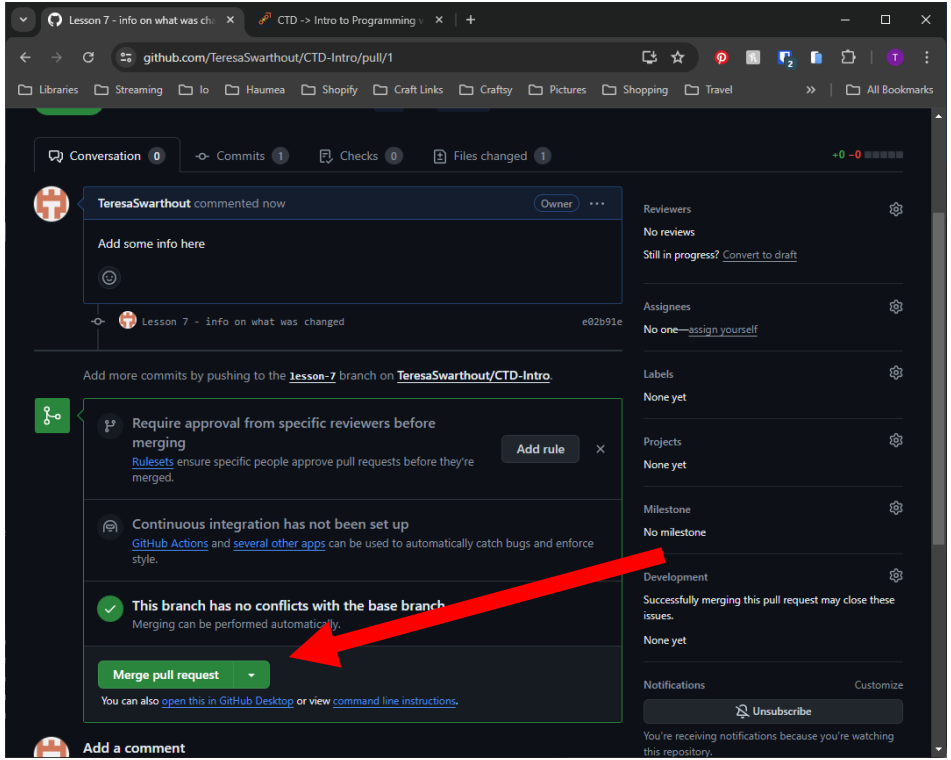

REMOTE
LOCAL
Branch lesson-7
Working Directory
STAGE
Branch lesson-7
Staging Area
COMMIT
User's
Local Repo
(Branch lesson-7)
PUSH
User's
Github Repo
(Branch lesson-7)
User's
Local Repo
(Main)

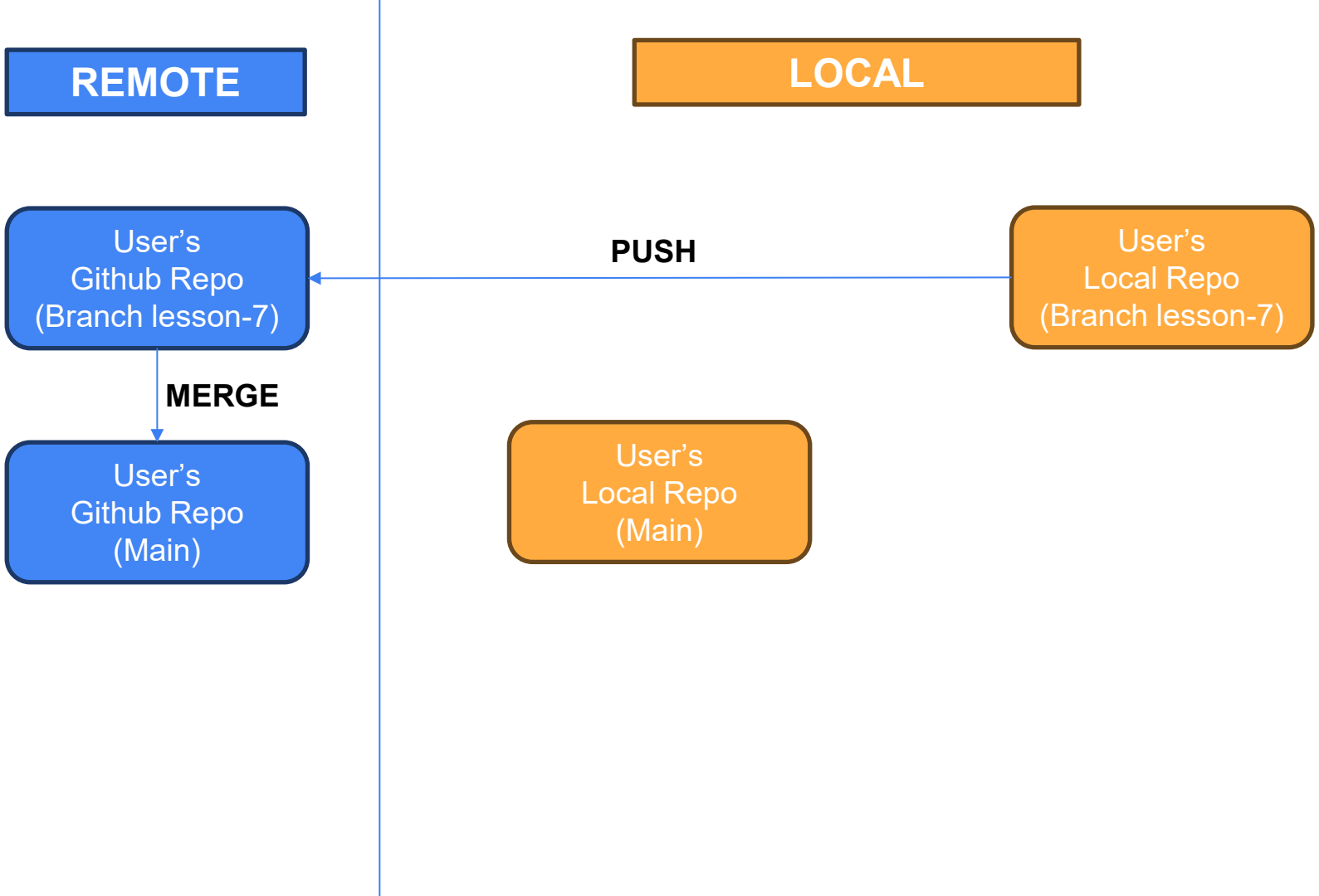
We've done a lot of work so far, but our file is only in the branch of our remote repo. Next, we want to create a **PULL REQUEST.** A pull request lets you tell others about changes you've pushed to a branch. In most cases, the files that are contained in your pull request would be reviewed by one or more reviewers before they would be allowed to be merged into the main repository.





Once your code has been reviewed (or if you are ready to move onto the next lesson), you want to **MERGE** your code into your main repo). At this point, your pull request will most likely be successful and the merge will happen automatically, but in a more complicated code bases, there may be conflicts that you have to resolve manually.




REMOTE
LOCAL
User's
Github Repo
(Branch lesson-7)
PUSH
User's
Local Repo
(Branch lesson-7)
MERGE
User's
Github Repo
(Main)
User's
Local Repo
(Main)

But wait, our local main repo doesn't have the changes yet!!

- Switch to main

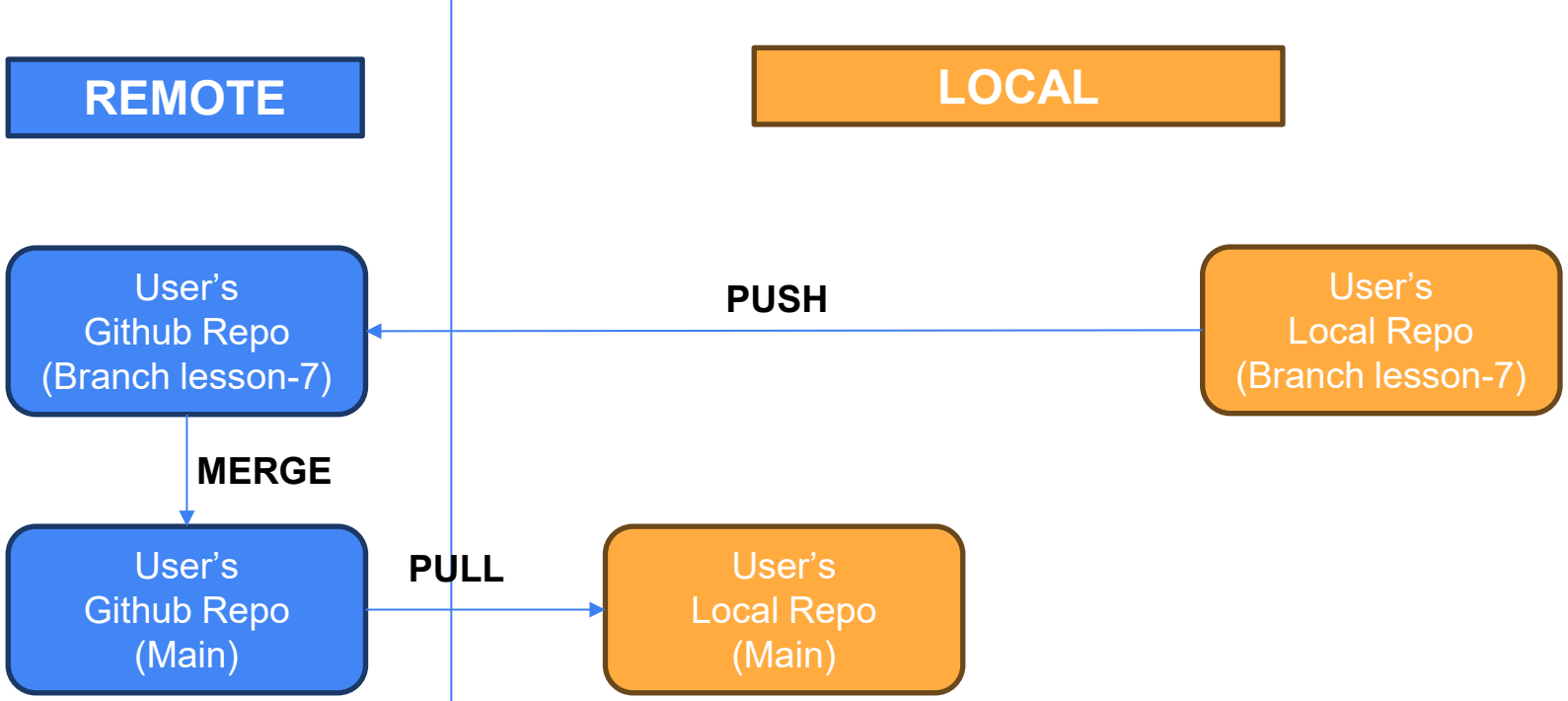
  git checkout main

- And pull the updates from the remote repo

  git pull

MINGW64:/c/Users/teres/CTD-repos/CTD-Intro

```
Teresa@TeresaLaptop MINGW64 ~/CTD-repos/CTD-Intro (lesson-7)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Teresa@TeresaLaptop MINGW64 ~/CTD-repos/CTD-Intro (main)
$ git pull
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 913 bytes | 6.00 KiB/s, done.
From https://github.com/TeresaSwarthout/CTD-Intro
   2222334..9cf6d71  main       -> origin/main
Updating 2222334..9cf6d71
Fast-forward
 index.html | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html

Teresa@TeresaLaptop MINGW64 ~/CTD-repos/CTD-Intro (main)
$
```



YAY!!
We're done!