GitHub

Matt Hewitt

2024-08-22

This is a starter guide for using GitHub, understanding its parts and features, and its integration with Program R to facilitate seamless group working and meeting of reproducibility standards.

For starters, GitHub is the website on which you create a user account and manage all repository settings, and Git is the program that manages the local repositories on your separate machines. There has previously been some confusion as to what the difference between these terms were.

As summered by some AI algorithm:

“GitHub is a web-based platform that allows developers to create, store, and collaborate on software projects. It uses Git, an open-source version control software, to let multiple people make changes to code simultaneously. GitHub provides access control, bug tracking, and other features to help developers manage their projects.”

As you can imagine, this is a really powerful tool in a world or remote workers and complex project structures by allowing everyone to stay up to date with the latest version of documents or analysis code. It is also useful for working on the same project on multiple machines, even for just one person, because it allows you to keep all your files up to date across machines.

Another power feature of Git, kinda the reason why it was made, is that it tracks and controls versions of files through time. If a bug gets introduced into a file, or you are just curious what a past version of a file looked like, you are able to essentially go back in time and continue working off a past version. It tracks what changes are made, who they are made by, and when they are saved. A very useful feature when many people are working on the same project.

# What you will need to get set up

## GitHub

To get started, you will need to set up a user account on GitHub’s website (<https://github.com/signup>). It will require that you verify your email address, so be sure you use an email you have access to.

Once you have an account set up, navigate to the Git download website (<https://git-scm.com/downloads>) and select your operating system. I do not use a macOS, so if you are using a mac book, it looks like you will need to first install a third party program that will then install Git. If you are using a Windows OS, you will want to download the ‘64-bit Git for Windows Setup’ program.

Once the download is done, work through the instillation prompts. There are ~20 different options and you can just accept all the defaults.

This gets us all set up with Git and GitHub.

## R & RStudio

If you have made it to this point in R Bootcamp I think it’s safe to assume that you already have R and R Studio installed on your computer.

However, for those of you who are joining us just for today, you will need to download and install these programs. At this website you can download both R and R Studio <https://posit.co/download/rstudio-desktop/>.

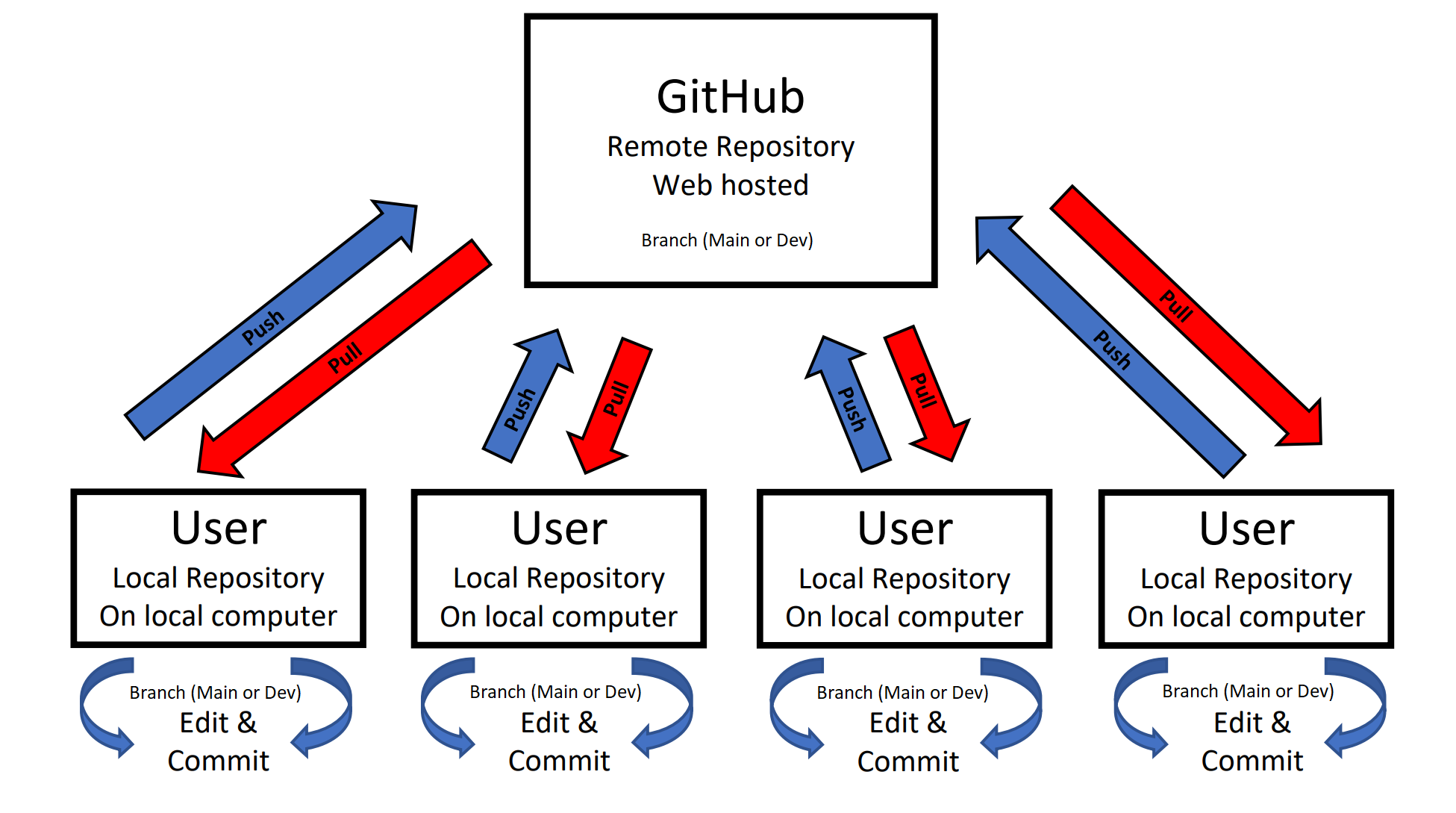
You will need to install Program R first, then download R Studio second. From what I remember, you can just accept all the defaults.

# Creating version controlled repositories

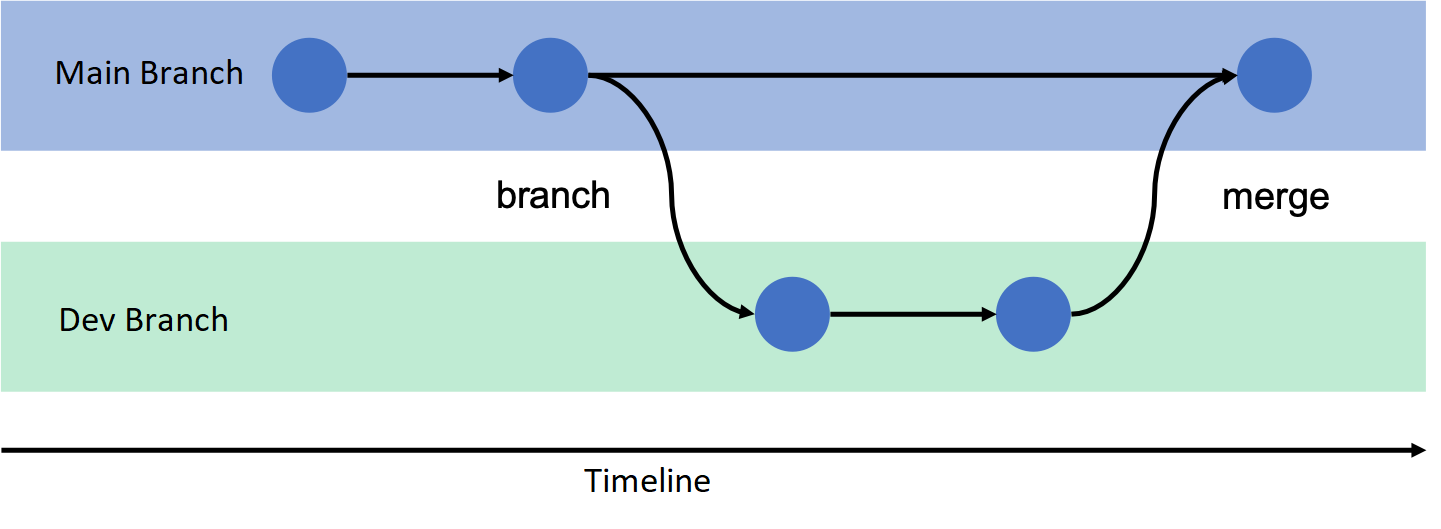
## Structure

Lets explore the structure of Git repositories.

In the image below you can see that there are two levels of organization. The **GitHub** level is the central repository that serves at the common source of all shared information. It is the top level where all users and developers send their updates and where they get updates from others.  
The next level down is the **User** level. The 4 User boxes represent the local repositories on peoples individual computers. You could think of these as four computers that a single person work’s on for their job, or it could represent four different users all working on the same project.

  
Fig 1

Another cool aspect of Git is that you can split a repository into multiple branches. In the image below I have written out a ‘Main’ and a ‘Dev’ branch. The concept of a branch is splitting a repository into multiple identical versions of itself, and working on the two versions separately. The idea behind having a Main and Dev branch (which stands for development) is that all users can use the files or programs on the main branch while developers continue to tweak features or other functions on the Dev branch. When developers are building or designing new features it often leaves a project in a state that does not work. Therefore, they work in the Dev branch until the feature is fully operational then ‘merge’ the head of the Dev branch to the head of the Main branch to then make the new feature available to users. You can think of the two branches as working in parallel and periodically merging and splitting.

  
Fig 2

## Operating within the system

The procedure for operating within this system has some good rules of thumb. If you are the only person working within a

## Integrating R and Git