Lab 3. Introduction to Git version control

Introduction

In this lab session you will become familiar with the basic Git workflow. You will learn to create and manage a source code repository, hosted in GitHub, both through the web page interface and a desktop client. In addition, if you have some spare time you will also have the opportunity to practice with the command line.

Objectives

By the end of the session, the student should:

- Understand the importance of using a version control system in software development.
- Be comfortable with the GitHub flow.

1. Git using GitHub's web interface

Start by signing up for a free account in GitHub (`https://github.com`). Once you have logged in, complete the Hello World tutorial (`https://guides.github.com/activities/hello-world`) that will guide you through your first steps with a distributed version control system.

2. Git with a GUI desktop client

Now that you know how to perform the most common operations, it is time to move to your computer. Download and install the official GitHub desktop client for Windows from `https://windows.github.com` (there is also a Mac version available at `https://mac.github.com`) and sign in with your username and password. Afterwards, click on the plus sign in the upper left hand corner and select Clone to download the `hello-world` repository to the directory of your choice (Figure 1).

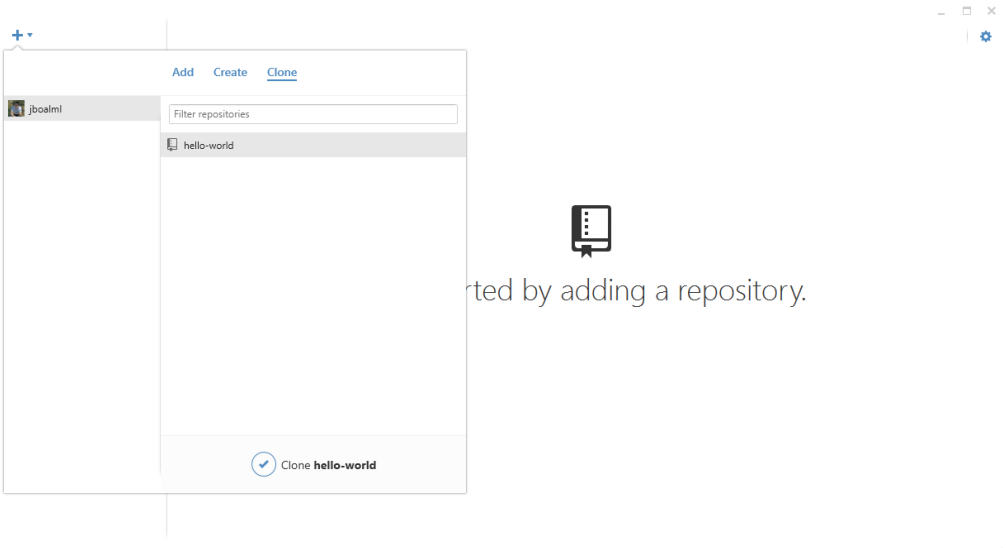

Figure 1. Cloning a repository in GitHub for Windows.

At this point, you should have a local copy of your project that you can freely edit, so go ahead and do it! You can modify the content of `README.md`, create new folders inside the project's folder, or add files. For example, you could create a folder named MATLAB and include the code from Lab 2. Remember to create a branch before editing! You do not want to break production code!

When you are done, go back to the desktop client and you should see something similar to Figure 2. The *Uncomitted changes* section shows the modifications you made. Select the files you want to commit (this is equivalent to adding them to the staging area), type a comment, and commit the changes.

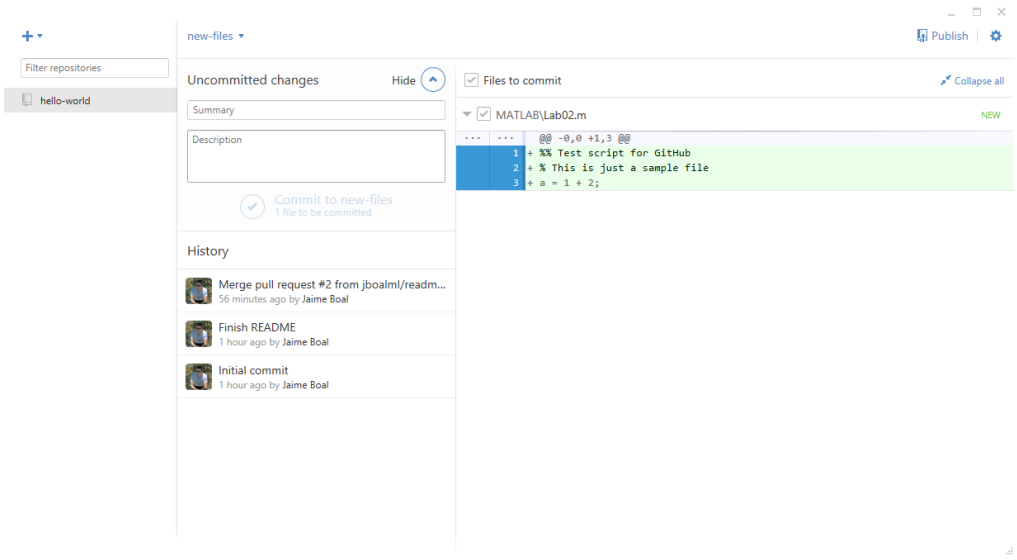

Figure 2. Uncommited changes.

However, the files have not been synced to the GitHub server yet. To do so, click Publish in the upper right hand corner (Figure 3). If the branch already existed in the server, the Sync button would appear instead. You should now be able to see the updates in your browser.

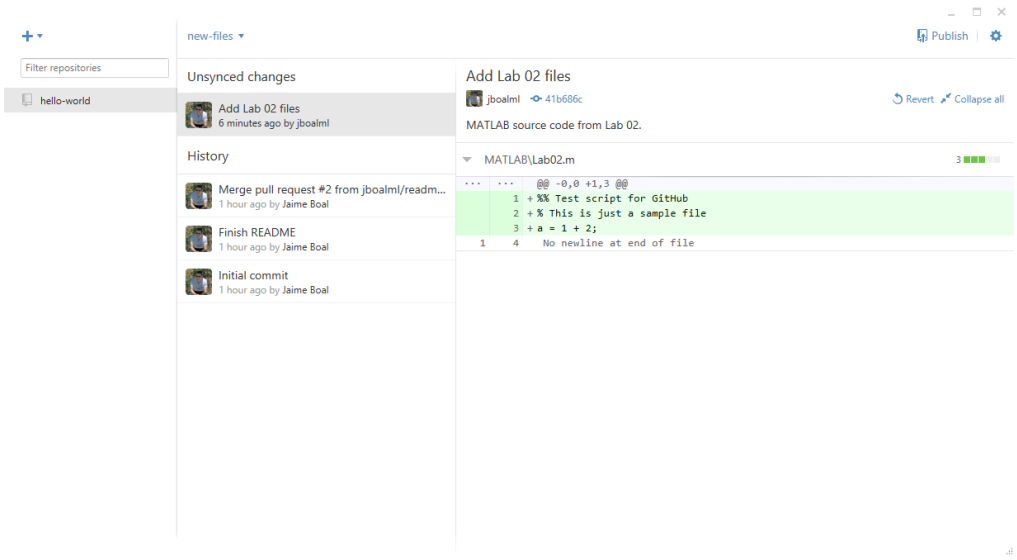

Figure 3. Unsynced commit.

Finally, try to modify the same line of code in different ways both in the web interface and in your local copy without syncing. When you try to sync, the desktop client should display a message stating that it encoutered conflicts that need to be manually resolved before proceeding (Figure 4).

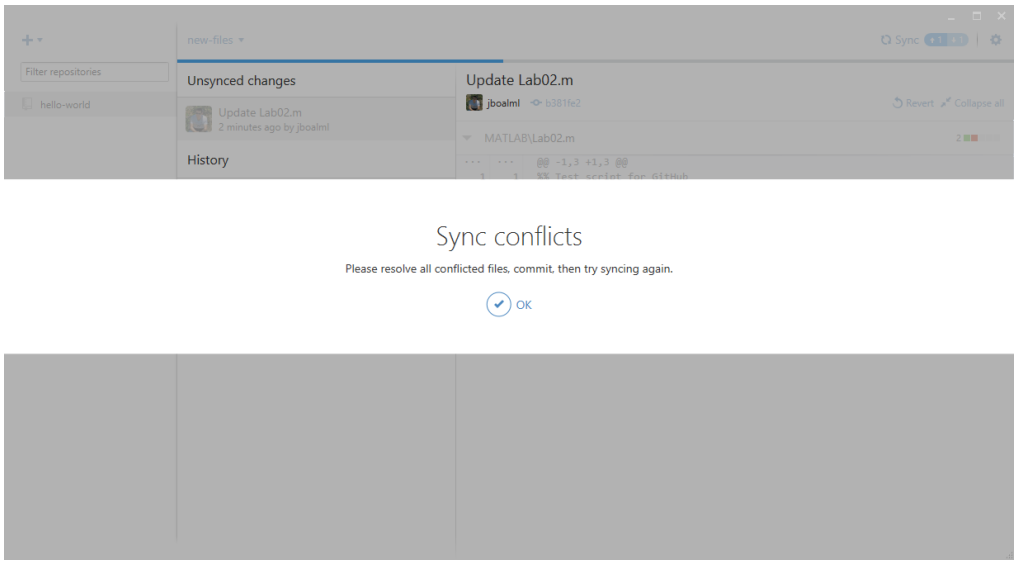

Figure 4. Conflict error message.

Update the conflicting parts marked between "<<<<<<< HEAD" and ">>>>>>>" (Figure 5), commit the changes, and sync. Congratulations, you have successfully solved your first merge conflict! You already know all the fundamentals to continue experimenting with version control systems!

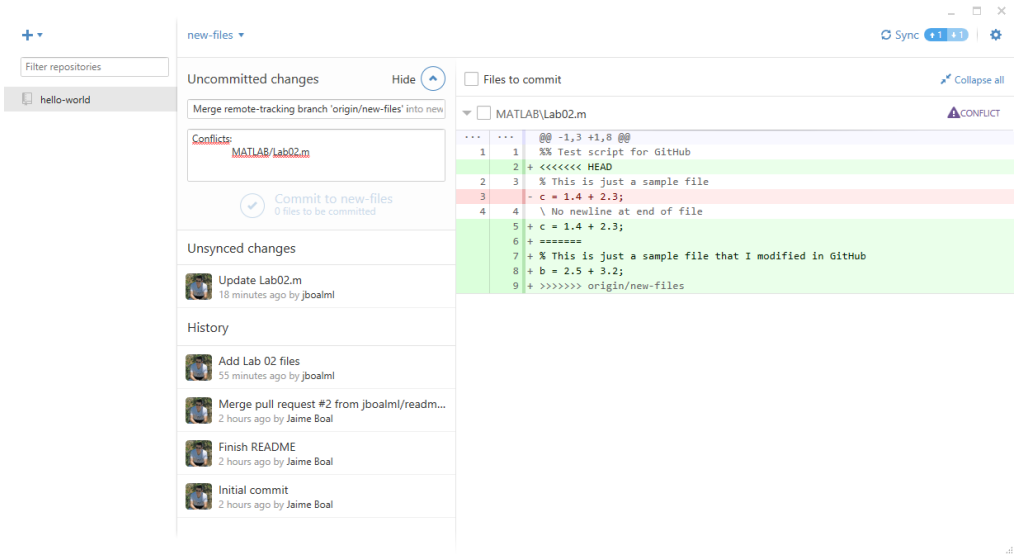

Figure 5. Merge conflict example.

3. Git through the command line (Optional)

Git was originally conceived to be used with a CLI (Command Line Interface). For this reason, many everyday commands are run much faster through the terminal than with a GUI, and some complex operations can still only be performed using the command line. If you have some time left, there is a brief interactive tutorial at `https://try.github.io` where you can practice with the basic commands. A more extensive Git cheat sheet can be downloaded at `https://training.github.com/kit`.