

# 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).

+•			
	Add Create <u>Clone</u>		
jboalml	Filter repositories		
	📮 hello-world		
		ted by adding a repository.	
	Clone hello-world		





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.

+*	new-files 🔻		🖟 Publish 🔅
Filter repositories	Uncommitted changes Hide 🔹	Files to commit	💉 Collapse all
lei hello-world	Summary	▼ ✓ MATLAB\Lab02.m	NEW
	Description	00 -0,0 +1,3 00 1 + XX Test script for GitHub 2 + X This is just a sample file 3 + a = 1 + 2;	
	Commit to new-files 1 file to be committed		
	History		
	Merge pull request #2 from jboalml/readm 56 minutes ago by Jaime Boal		
	Finish README 1 hour ago by Jaime Boal		
	Initial commit 1 hour ago by Jaime Boal		

Figure 2. Uncommitted 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.

			×
+-	new-files 🔻		🚯 Publish 🛛 🔅
Filter repositories	Unsynced changes	Add Lab 02 files	
la hello-world	Add Lab 02 files 6 minutes ago by jboalml	jboalml ◆ 41b686c MATLAB source code from Lab 02.	S Revert ≠ <sup>e</sup> Collapse all
	History	✓ MATLAB\Lab02.m	3
	Merge pull request #2 from jboalml/readm 1 hour ago by Jaime Boal	@@ -0,0 +1,3 @@ 1 + %% Test script for GitHub 2 + % This is just a sample file	
	Finish README 1 hour ago by Jaime Boal	3 + a = 1 + 2; 1 4 No newline at end of file	
	Initial commit 1 hour ago by Jaime Boal		

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).



#### ADVANCED COMPUTING TOOLS LABORATORY

	Unsynced changes Update Lab02.m 2 minutes ago by Jooatml History	Update Lab02.m jooalml	S Revert → <sup>4</sup> Collapse all		
Sync conflicts Please resolve all conflicted files. commit, then try syncing again.					

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!

					~
+•	new-files 🔻				🗘 Sync 💶 🖬 🔅
Filter repositories	Uncommitted changes Hide 🔦	F	iles to	o commit	💉 Collapse all
le hello-world	Merge remote-tracking branch 'origin/new-files' into new	-	MAT	TLAB\Lab02.m	
	Conflicts: MATLAB/Lab02.m			00 -1,3 +1,8 00 %% Test script for GitHub	
	Commit to new-files 0 files to be committed	2	3	<pre>&gt; * * *******************************</pre>	
		4	4	<pre>\ No newline at end of file</pre>	
	Unsynced changes		7	<pre>/ + % This is just a sample file that I modified in GitHub / + b = 2.5 + 3.2;</pre>	
	18 minutes ago by jboalml		9	<pre>/ + &gt;&gt;&gt;&gt;&gt;&gt; origin/new-tiles</pre>	
	History				
	Add Lab 02 files 55 minutes ago by jboalml				
	Merge pull request #2 from jboalml/readm 2 hours ago by Jaime Boal				
	Finish README 2 hours ago by Jaime Boal				
	Initial commit 2 hours ago by Jaime Boal				

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.