

# MATLAB & Git Versioning: The Very Basics

*basic guide for using git (command line) in the development of MATLAB code (windows)*

The information for this small guide was taken from the following websites:

[http://benheavner.com/systemsbio/index.php?title=MATLAB\\_git](http://benheavner.com/systemsbio/index.php?title=MATLAB_git)

[http://dasl.mem.drexel.edu/wiki/index.php/Git\\_Tutorial\\_-\\_Part\\_1](http://dasl.mem.drexel.edu/wiki/index.php/Git_Tutorial_-_Part_1)

<http://rogerdudler.github.io/git-guide/>

<http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1#awesm=~oGRovPJT6emrJc>

<https://codio.com/s/docs/git/git-primer/>

This goal of this short guide is to get a simple, working environment for using Git and MATLAB. The command line is used, so no gui.

Pulling and pushing requests on GitHub will be done using the Gui bash command line.

The best way to learn Git is to create a simple project with a few files in it and then experiment like crazy. You can play around with all the Git commands and sync with GitHub repos to your heart's content without worrying about doing any coding.

Topics advanced features like branching are not considered.

# 1 REQUIREMENTS — GETTING STARTED

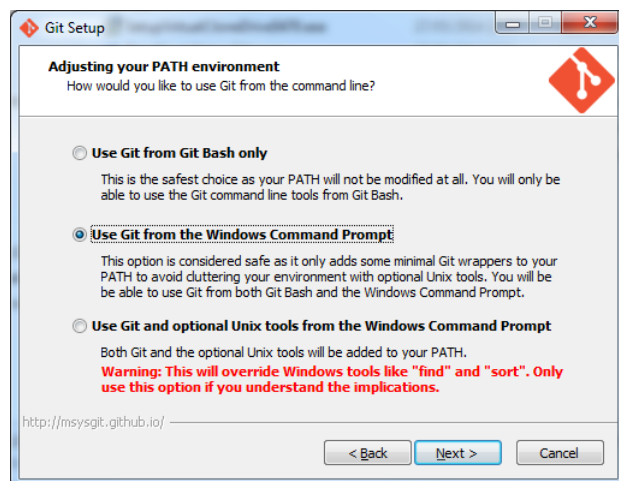
---

When Git versioning is being used, it must be installed and a repository must be initialized. Git can be used from within MATLAB by using a wrapper that can be found on MATLAB Central.

## 1.1 GIT SOFTWARE

The Git software can be downloaded from <http://git-scm.com/download/win>

During the installation it is important to enable use from the Windows command line



In this way the Windows Path is adapted and Git can be used from within other applications like MATLAB.

## 1.2 INSTALLING THE GIT WRAPPER

Get the git wrapper from MATLAB Central.

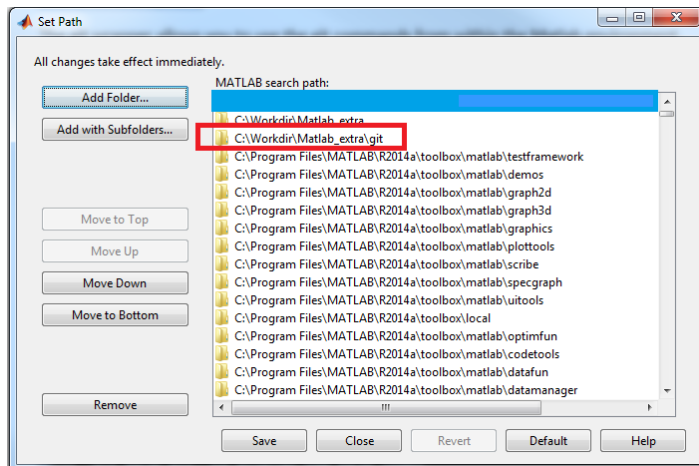
<http://www.mathworks.com/MATLABcentral/fileexchange/29154-a-thin-MATLAB-wrapper-for-the-git-source-control-system>

Or from

<https://github.com/manur/MATLAB-git>

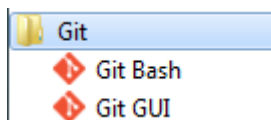
The git wrapper allows you to use the git commands from within the MATLAB environment.

Note: I use a folder on my C-drive containing *special* MATLAB utilities, this folder is added to the MATLAB Path, such that the utilities, in this case the 'git' command is always accessible from within MATLAB.

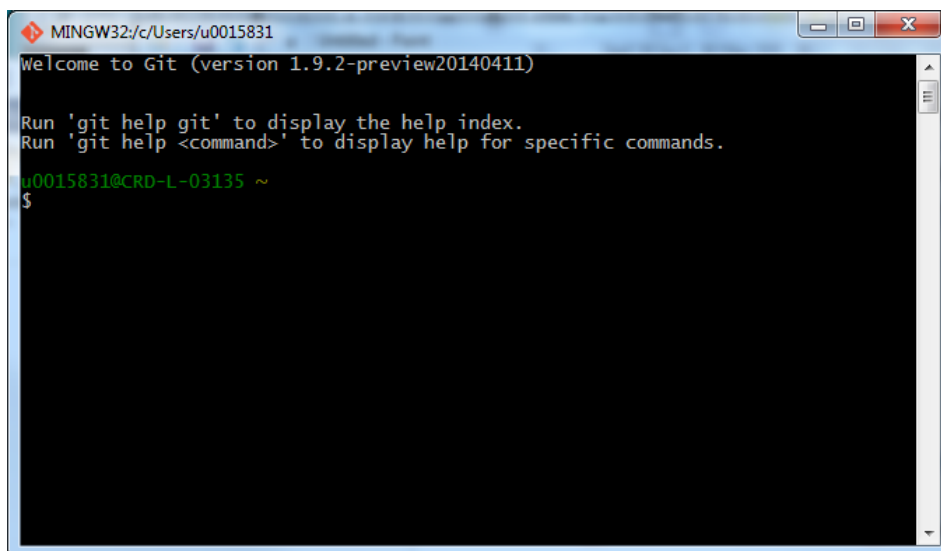


### 1.3 CREATE A LOCAL REPOSITORY

Once git is installed on your computer, it will be available under the Programs Menu



Start the Git Bash command line



In this command window the basic Git commands can be entered.

When starting with versioning, the project folder containing the MATLAB code need to be prepared for Git.

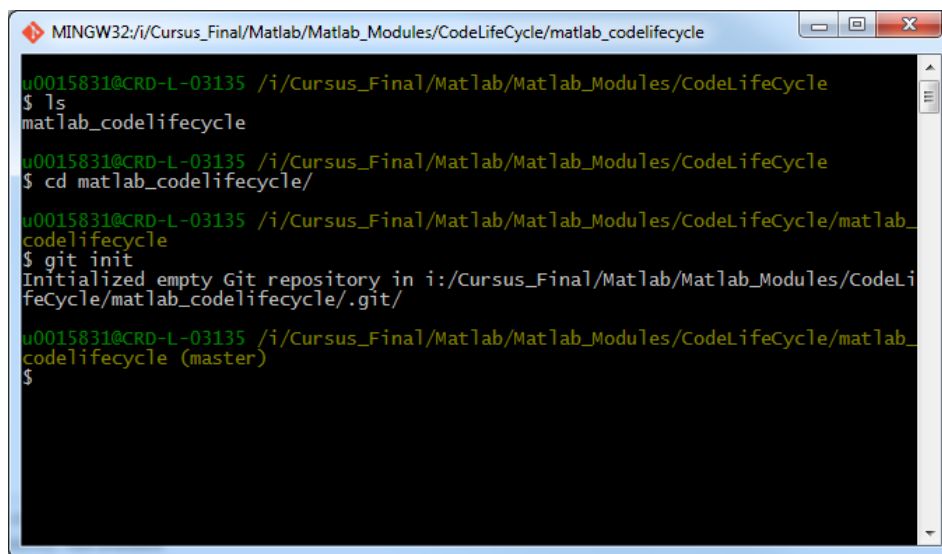
When using Git, you will need to have a repository. You can start by creating a project repository on your own local machine (laptop, desktop) first.

Assume your project directory will be called 'MATLAB\_codelifecycle', and this folder will contain all the code related to the project

Initialize Git using the (Git Bash) command window

- Go to the directory (use `cd`) and initialise Git on the directory with the command:

```
git init
```

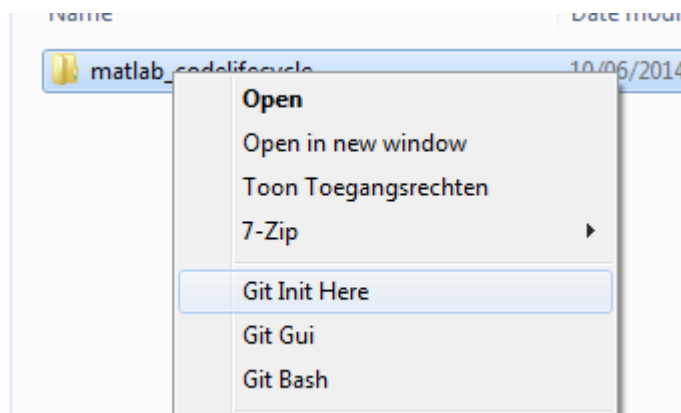


```

MINGW32:/i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle/matlab_codelifecycle
u0015831@CRD-L-03135 /i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle
$ ls
matlab_codelifecycle
u0015831@CRD-L-03135 /i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle
$ cd matlab_codelifecycle/
u0015831@CRD-L-03135 /i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle/matlab_
codelifecycle
$ git init
Initialized empty Git repository in i:/Cursus_Final/Matlab/Matlab_Modules/CodeLi
feCycle/matlab_codelifecycle/.git/
u0015831@CRD-L-03135 /i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle/matlab_
codelifecycle (master)
$

```

Tip: Another Option: Initialize Git using Windows Explorer  
when correctly installed, right-clicking the folder in Windows Explorer will offer the possibility to initialize the directory (Git Init Here)



At this point the hidden folder `.git` will be created in this directory and the Git repository is ready to be used: Git commands are accepted.

## 1.4 VERY FIRST TIME USE

### 1.4.1 Initial configuration

If this is the very first time that git is used, some minimal additional configuring work needs to be done. These steps can be found in any Git tutorial:

```
git config --global user.name "Your Name Here"
```

specifying your user name is useful when committing changes, the user name will also be noted.

```
git config --global user.email "your\_email@youremail.com"
```

email account is useful when using Github, use the email account that was used for signing up in Github.

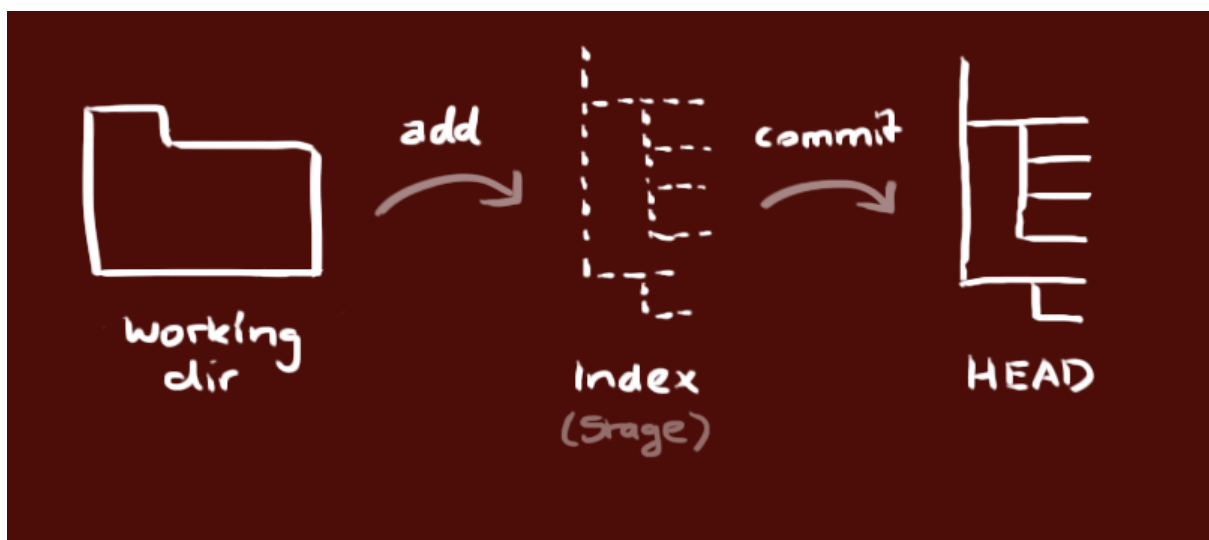
When using Github, you have to connect remotely to this repository, this can best be done securely with ssh keys. The steps that you need to perform are given at:

<https://help.github.com/articles/generating-ssh-keys>

## 2 WORKFLOW

---

your local repository consists of three "trees" maintained by git. the first one is your `Working Directory` which holds the actual files. The second one is the `Index` which acts as a staging area and finally the `HEAD` which points to the last commit you've made.

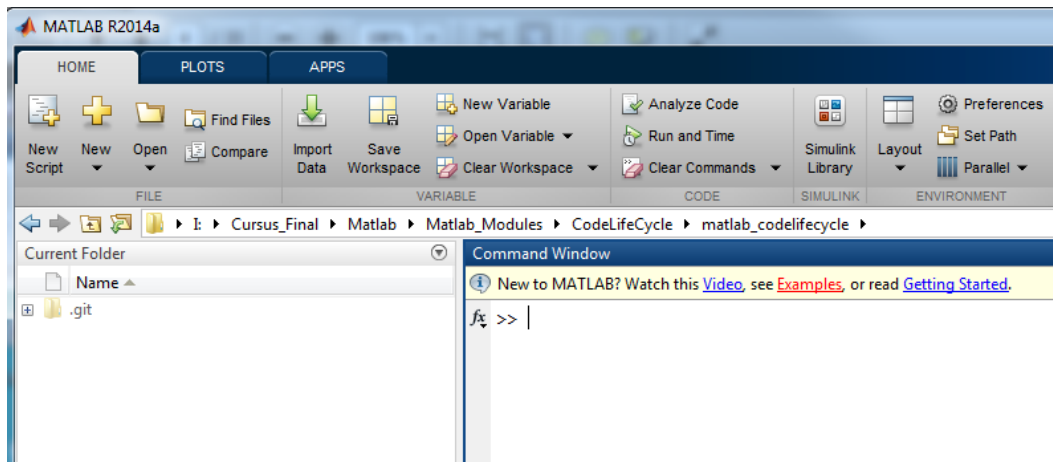


(copied from <http://rogerdudler.github.io/git-guide/>)

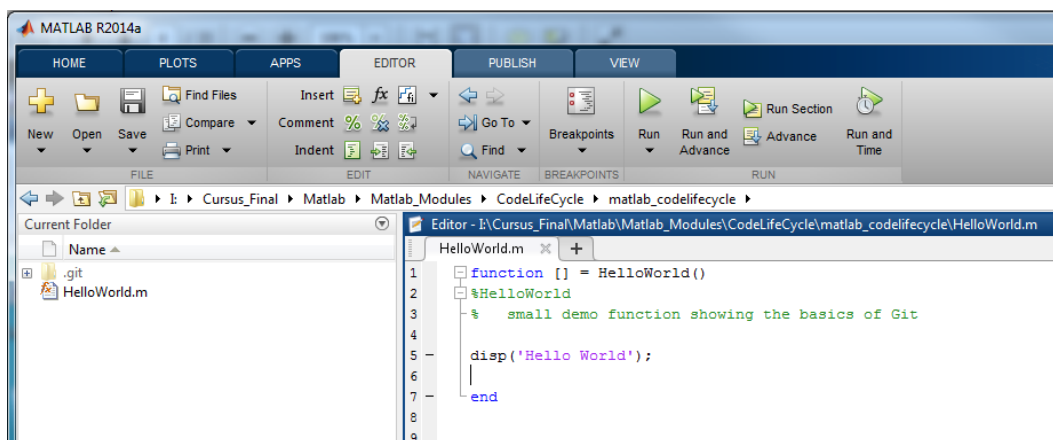
### 2.1 ADD A FILE TO THE REPOSITORY

Once the repository is initialized, a first file for this repository can be created.

Make sure you are in your git repository folder, then open up a text editor, or make the repository folder your current folder in MATLAB



Write a simple MATLAB function.

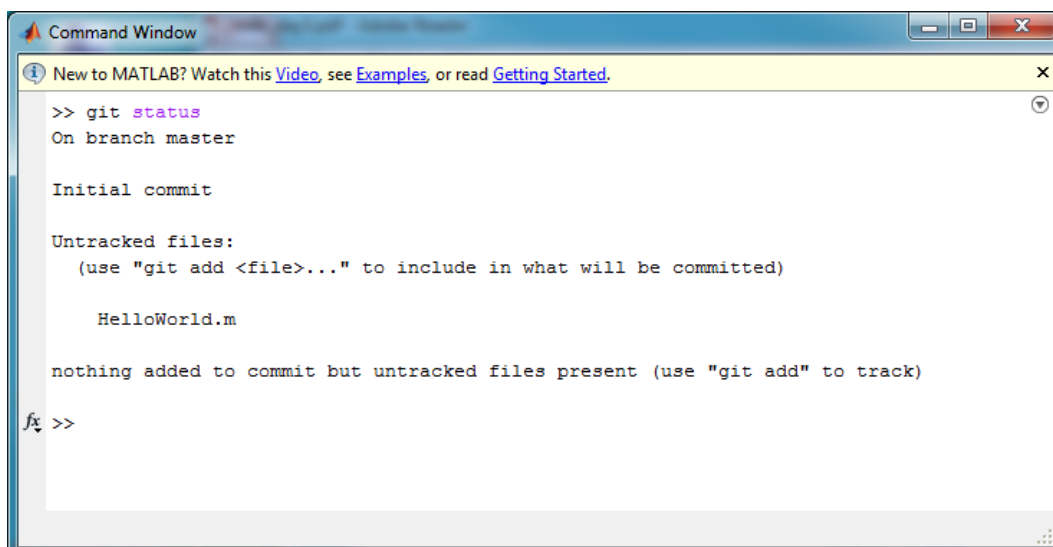


For this example, write a simple MATLAB function (HelloWorld.m)

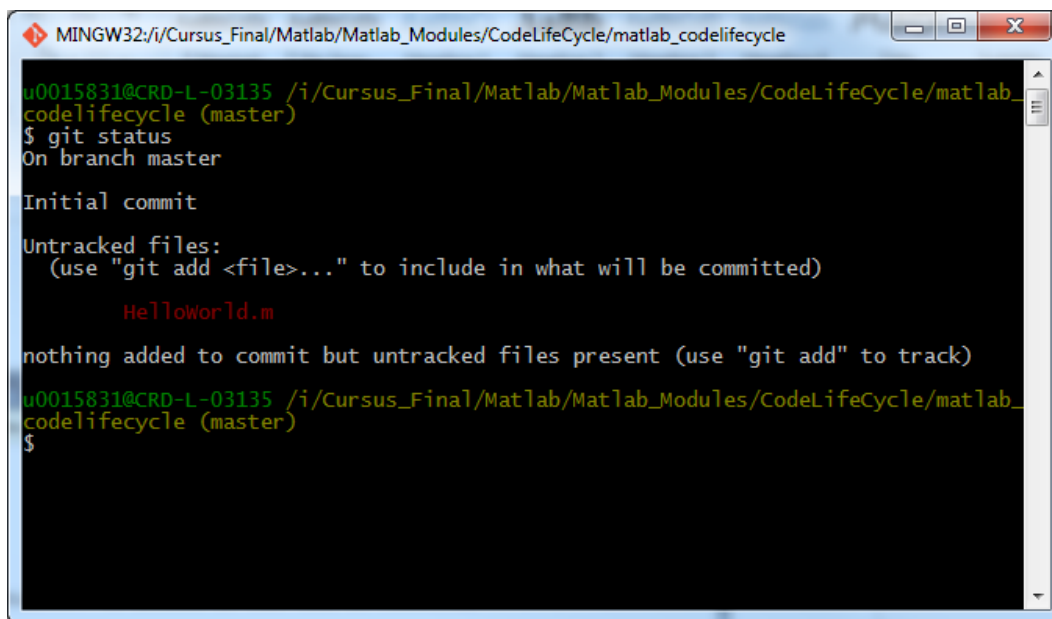
When the git wrapper is installed in MATLAB, you can use the Git commands inside the MATLAB command window.

Report the status:

```
git status
```



Or in the command window

A screenshot of a terminal window titled 'MINGW32:/i/Cursus\_Final/Matlab/Matlab\_Modules/CodeLifeCycle/matlab\_codelifecycle'. The terminal shows the output of the 'git status' command. It indicates that the user is on the 'master' branch and that there is an 'Initial commit'. A file named 'HelloWorld.m' is listed as an 'Untracked file'. The terminal also provides instructions on how to use 'git add' to track the file. The prompt '\$' is visible at the bottom.

```
MINGW32:/i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle/matlab_codelifecycle
u0015831@CRD-L-03135 /i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle/matlab_codelifecycle (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

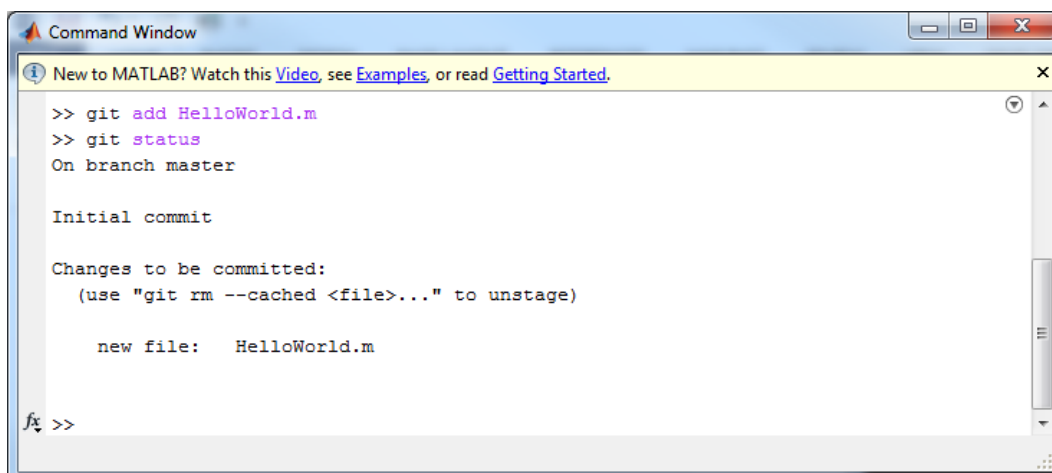
        HelloWorld.m

nothing added to commit but untracked files present (use "git add" to track)
u0015831@CRD-L-03135 /i/Cursus_Final/Matlab/Matlab_Modules/CodeLifeCycle/matlab_codelifecycle (master)
$
```

Git found 1 *untracked* file in the current folder. In order to get this file in the versioning system, it must be staged, using the add command.

```
git add HelloWorld.m
```

Check the status of the repository again to see what changes you've made:

A screenshot of a MATLAB Command Window titled 'Command Window'. It shows the output of the 'git add' and 'git status' commands. The file 'HelloWorld.m' is now listed as a 'Changes to be committed' file. The terminal also provides instructions on how to use 'git rm --cached' to unstage the file. The prompt '>>' is visible at the bottom.

```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
>> git add HelloWorld.m
>> git status
On branch master

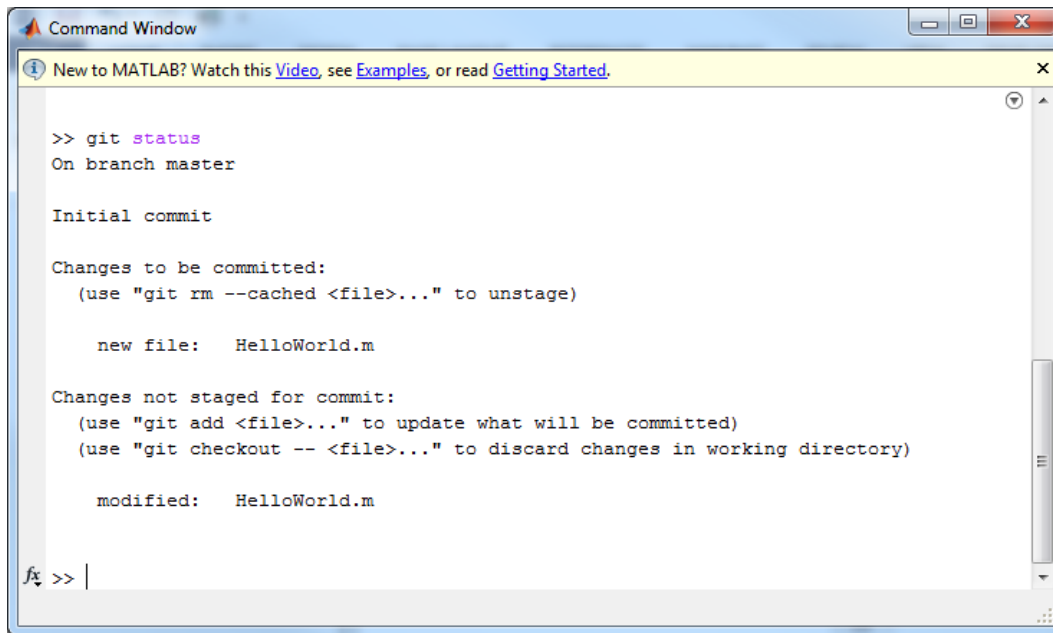
Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   HelloWorld.m

fx >>
```

If the file is not yet committed, and you start already to make changes, Git will report it.

A screenshot of the MATLAB Command Window. At the top, there is a yellow banner with the text "New to MATLAB? Watch this [Video](#), see [Examples](#), or read [Getting Started](#)." Below the banner, the command window shows the output of the `git status` command. The output indicates that the user is on the `master` branch and has made an initial commit. It lists changes to be committed, specifically a new file named `HelloWorld.m`. It also shows changes not staged for commit, including the same file `HelloWorld.m` which has been modified. The prompt `>> |` is visible at the bottom.

```
>> git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   HelloWorld.m

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   HelloWorld.m

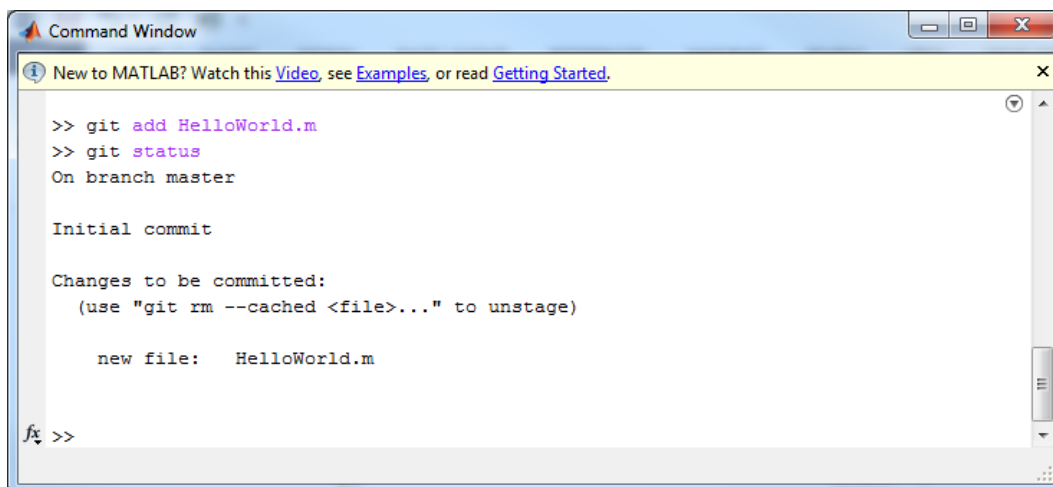
fx >> |
```

It will mark that the file is modified and not staged yet.

Adding the file to the staging, with the command

```
git add HelloWorld.m
```

Will change the status.

A screenshot of the MATLAB Command Window showing the output of `git add HelloWorld.m` followed by `git status`. The output is similar to the previous screenshot, but now only the `HelloWorld.m` file is listed under "Changes to be committed", and there are no changes listed under "Changes not staged for commit". The prompt `>>` is visible at the bottom.

```
>> git add HelloWorld.m
>> git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   HelloWorld.m

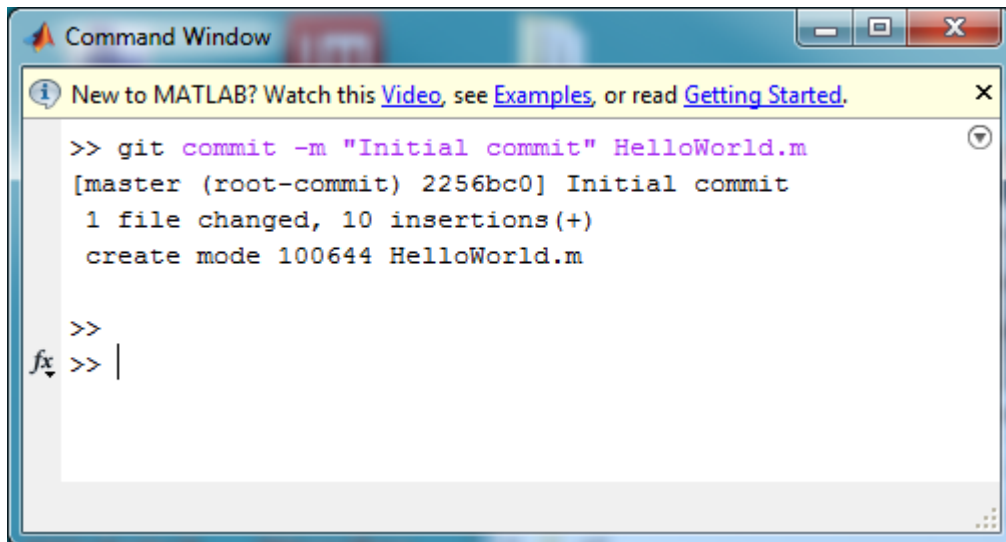
fx >>
```

## 2.2 CREATE YOUR FIRST COMMIT

Without committing, the file is not yet stored in the repository. This is done with the command:

```
git commit -m "first commit" HelloWorld.m
```





```

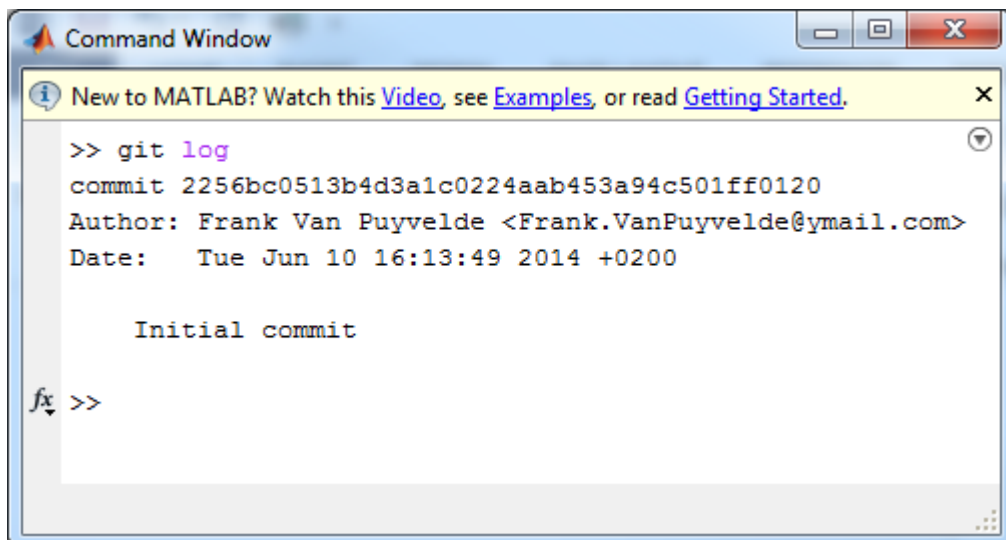
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
>> git commit -m "Initial commit" HelloWorld.m
[master (root-commit) 2256bc0] Initial commit
 1 file changed, 10 insertions(+)
 create mode 100644 HelloWorld.m

>>
fx >> |

```

You can view the commit history using:

```
git log
```



```

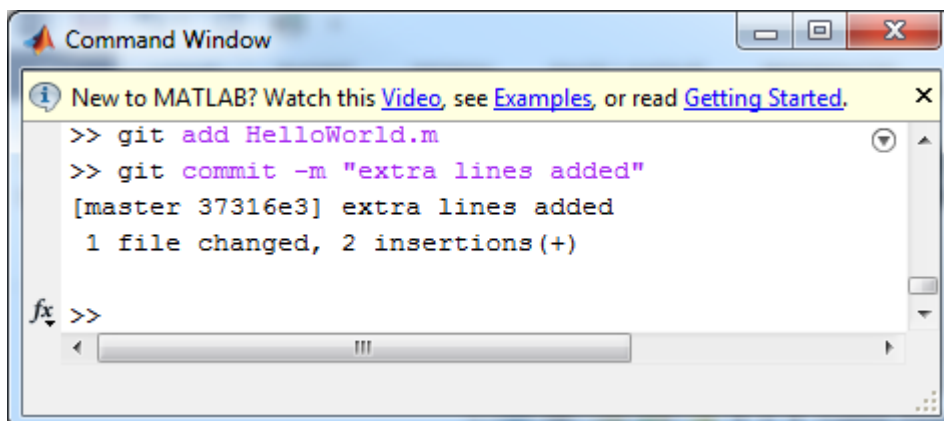
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
>> git log
commit 2256bc0513b4d3a1c0224aab453a94c501ff0120
Author: Frank Van Puyvelde <Frank.VanPuyvelde@ymail.com>
Date:   Tue Jun 10 16:13:49 2014 +0200

    Initial commit

fx >>

```

Change the initial code and start tracking changes to the project



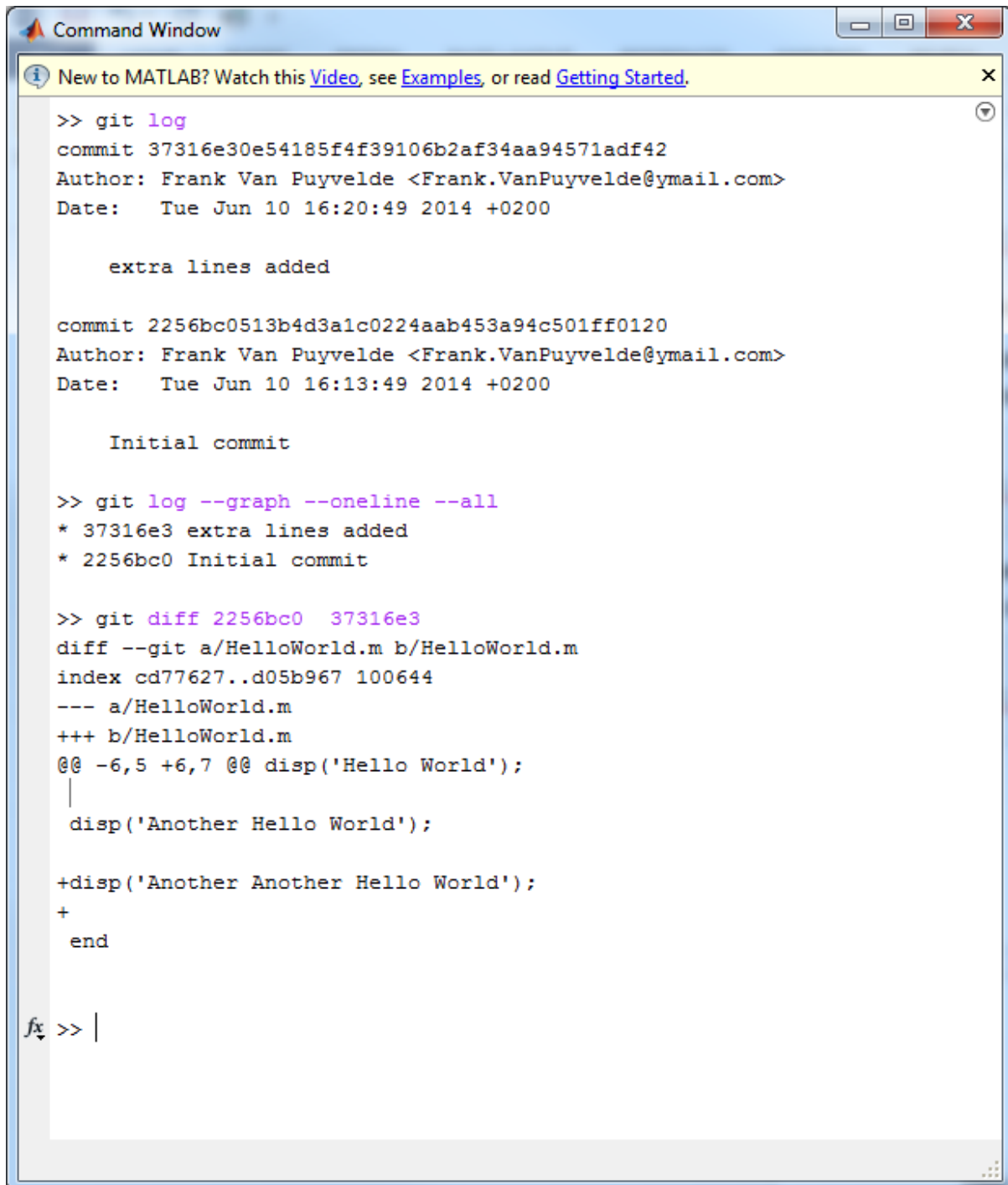
```

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
>> git add HelloWorld.m
>> git commit -m "extra lines added"
[master 37316e3] extra lines added
 1 file changed, 2 insertions(+)

fx >>

```

Check the logging, and with the diff option you can track the differences in between the files.



```

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

>> git log
commit 37316e30e54185f4f39106b2af34aa94571adf42
Author: Frank Van Puyvelde <Frank.VanPuyvelde@ymail.com>
Date: Tue Jun 10 16:20:49 2014 +0200

    extra lines added

commit 2256bc0513b4d3a1c0224aab453a94c501ff0120
Author: Frank Van Puyvelde <Frank.VanPuyvelde@ymail.com>
Date: Tue Jun 10 16:13:49 2014 +0200

    Initial commit

>> git log --graph --oneline --all
* 37316e3 extra lines added
* 2256bc0 Initial commit

>> git diff 2256bc0 37316e3
diff --git a/HelloWorld.m b/HelloWorld.m
index cd77627..d05b967 100644
--- a/HelloWorld.m
+++ b/HelloWorld.m
@@ -6,5 +6,7 @@ disp('Hello World');
|
|   disp('Another Hello World');
|
+disp('Another Another Hello World');
+
+end

fx >> |
  
```

## 2.3 INSPECTING A PREVIOUS VERSION

With checkout, you can inspect a previous version

```

Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
>>
>> git log --oneline
e413b5c added extra subfunction
37316e3 extra lines added
2256bc0 Initial commit

>> git checkout 2256bc0|
Note: checking out '2256bc0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b new_branch_name

HEAD is now at 2256bc0... Initial commit

>>
fx >>

```

After inspecting the code, you can return to the correct state with

```
git checkout master
```

## 2.4 SOME MORE COMMANDS

```
git log
```

This log will show a history of the changes.

For a fancier output, extra options can be specified:

```
git log --graph --oneline --all
```

check your settings, with

```
git config --list
```

## 2.5 PUSH THE COMMIT TO THE SERVER

Note: pushing and pulling doesn't seem to work fine from within the MATLAB environment. This is done in the Git Bash command window.

Pushing a repository, requires a remote repository. The basic command to add a remote repository is:

```
git remote add <shortname> <url>
```

Shortnames are short, easy-to-remember names that make it easier to work with your remote repos in the future

Note: there is a difference in accessing github using ssh-keys or using logon-id and password.

When pushing a local repository to a remote repository, you need to tell git on the local machine to which repo on Github should be linked to.

Option 1: using ssh-keys

```
Git remote add myProj1 git@github.com:username/repo\_name.git
```

Option 2: userid / password

```
git remote add myProj1 https://github.com/username/repo\_name.git
```

Check the status of the remote repository

```
git remote -v
```

Now that a commit was made, the local repository can be synchronized with the remote server to make your changes available. Moving commits from your a local repository to a remote repository is done with `git push`

```
git push myProj1 master
```

pushes the local repository to Github:

## 2.6 USING THE GIT PULL COMMAND

If you only ever work from a single workstation, then your workflow will be very simple. Changes to your local files get added, committed, then pushed to the server. However, if more than one person is working on a project, your local copy won't necessarily be up-to-date. To make sure you have the latest version, you pull changes from the server to your local repository.

### 2.6.1 Resolving Conflicts

Taken from <https://codio.com/s/docs/git/git-primer/>

When you pull in from the remote, you may get a conflict warning. This will happen if someone else has modified code in a way that Git cannot automatically resolve, usually because you have been editing the same bit of code.

When this happens, you will need to resolve the conflict. If you open up the file, you will see something like this

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">
  please contact us at support@github.com
</div>
>>>>>> iss53:index.html
```

You simply need to remove the code block that you want to dispose of. The top block is your code and the bottom comes from the code being merged. If you want to keep your code, you will want to end up with

```
<div id="footer">contact : email.support@github.com</div>
```

if you want the merged code to remain, it will be

```
<div id="footer">
  please contact us at support@github.com
</div>
```

To minimize conflicts, you should *Commit little and often* and *Pull from the remote master often*.