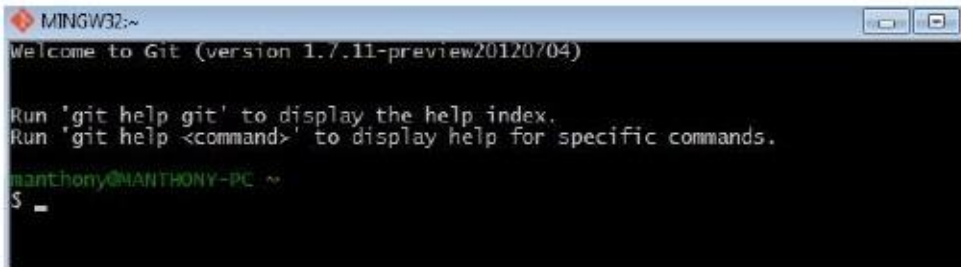


# Software Testing and Inspection

## Lab 1.1 - Using Git

1. Install and run Git for windows (should be already installed)



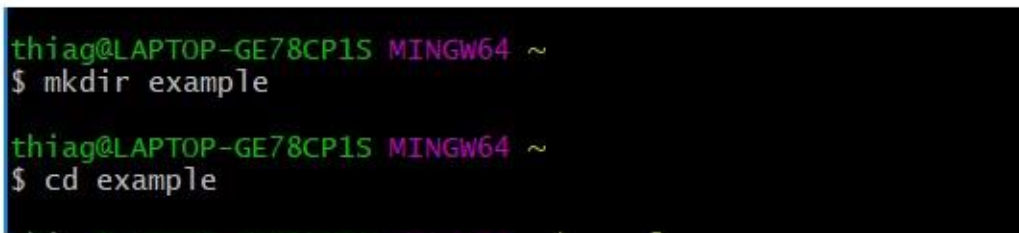
```
MINGW32:~
Welcome to Git (version 1.7.11-preview20120704)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

manthony@NANTHONY-PC ~
$ _
```

2. Create a local git repository

When creating a new project on your local machine using git, you'll first create a new repository (or often, 'repo', for short). To use git we'll be using "Git Bash". To begin, open up a terminal and move to where you want to place the project on your local machine using the cd (change directory) command. For example, if you have a 'projects' folder on your desktop, you'd do something like:

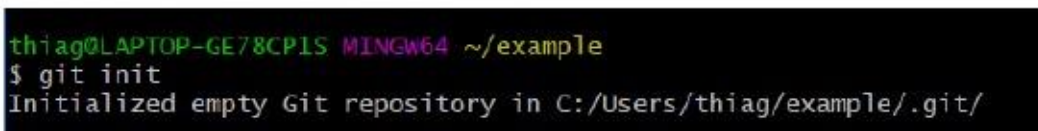


```
thiag@LAPTOP-GE78CP1S MINGW64 ~
$ mkdir example

thiag@LAPTOP-GE78CP1S MINGW64 ~
$ cd example
```

3 – Initialize git repository

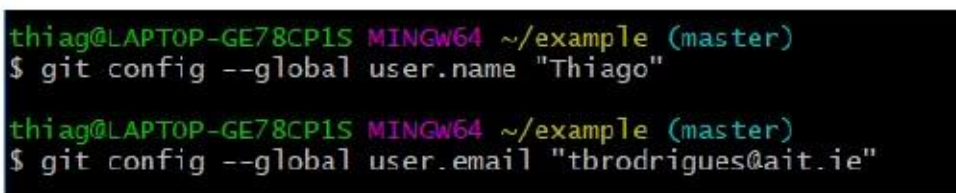
To initialize a git repository in the root of the folder, run the git init command:



```
thiag@LAPTOP-GE78CP1S MINGW64 ~/example
$ git init
Initialized empty Git repository in C:/Users/thiag/example/.git/
```

4 – User information

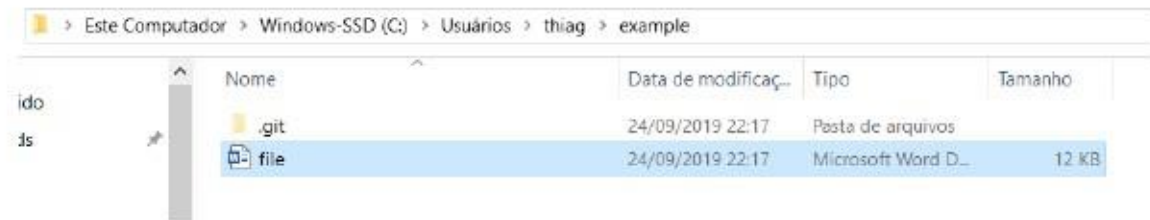
Next, let git know who you are by running.



```
thiag@LAPTOP-GE78CP1S MINGW64 ~/example (master)
$ git config --global user.name "Thiago"

thiag@LAPTOP-GE78CP1S MINGW64 ~/example (master)
$ git config --global user.email "tbrodrigues@ait.ie"
```

## 5 – Add a new file to the repo



Once you've added or modified files in a folder containing a git repo, git will notice that changes have been made inside the repo but git won't officially keep track of the file (that is, put it in a commit - we'll talk more about commits next) unless you explicitly tell it to.

After creating the new file, you can use the git status command to see which files git knows exist.

```
thiag@LAPTOP-GE78CP15 MINGW64 ~/example (master)
$ git status
On branch master

No commits yet

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

    file.docx

nothing added to commit but untracked files present (use "git add" to track)
```

What this basically says is, "Hey, we noticed you created a new file called "file.docx", but unless you use the 'git add' command we aren't going to do anything with it."

An interlude: The staging environment, the commit, and you.

One of the most confusing parts when you're first learning git is the concept of the staging environment and how it relates to a commit. A commit is a record of what files you have changed since the last time you made a commit.

Essentially, you make changes to your repo (for example, adding a file or modifying one) and then tell git to put those files into a commit. Commits make up the essence of your project and allow you to go back to the state of a project at any point. So, how do you tell git which files to put into a commit? This is where the staging environment or index come in. As seen in Step 5, when you make changes to your repo, git notices that a file has changed but won't do anything with it (like adding it in a commit).

To add a file to a commit, you first need to add it to the staging environment. To do this, you can use the git add command (see Step 3 below). Once you've used the git add command to add all the files you want to the staging environment, you can then tell git to package them into a commit using the git commit command. Note: The staging environment, also called 'staging', is the new preferred term for this, but you can also see it referred to as the 'index'.

## 6 – Add a file to the staging environment

Add a file to the staging environment using the git add command.

```
thiag@LAPTOP-GE78CP1S MINGW64 ~/example (master)
$ git add file.docx

thiag@LAPTOP-GE78CP1S MINGW64 ~/example (master)
$ git commit -m "New file added"
[master (root-commit) ab35d91] New file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file.docx
```

If you rerun the git status command, you'll see that git has added the file to the staging environment (notice the "Changes to be committed" line). To reiterate, the file has not yet been added to a commit, but it's about to be.

## 7 – Create a commit

It's time to create your first commit! Run the command

git commit -m "Your message about the commit"

```
thiag@LAPTOP-GE78CP1S MINGW64 ~/example (master)
$ git commit -m "New file added"
[master (root-commit) ab35d91] New file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file.docx
```

The message at the end of the commit should be something related to what the commit contains - maybe it's a new feature, maybe it's a bug fix, maybe it's just fixing a typo. Don't put a message like "asdfsdf" or "foobar". That makes the other people who see your commit sad. Very, very, sad.

Touch .gitignore

This command will create our new file

```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

In the file we will add \*.log so it will ignore any files with this extension

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
```

It does not see the log file because it is listed as a file to ignore

## Lab 1.2 – Working with a remote repository

1 – Go to <https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>

2 – Create a Bitbucket account

3 – Create a git repository

<https://confluence.atlassian.com/bitbucket/create-a-git-repository-759857290.html>

4 – Make a new directory and clone your git repository locally.

5 – Create a file and locally push it to Bitbucket

`git add`

`git status`

`git commit -m 'Initial commit message'`

`git push -u origin master`

6 – Change your file online with Bitbucket

7 – Pull changes to the local repository.

`git pull --all`