[1 Step 1: Create a local git repository 1](#_Toc499884679)

[2 Step 2: Add a new file to the repo 1](#_Toc499884680)

[2.1 An interlude: The staging environment, the commit, and you 2](#_Toc499884681)

[3 Step 3: Add a file to the staging environment 2](#_Toc499884682)

[4 Step 4: Create a commit 3](#_Toc499884683)

[5 Step 5: Create a new branch 3](#_Toc499884684)

[5.1 Try This! 4](#_Toc499884685)

# Step 1: 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, if it is not available on your machine download it from https://git-scm.com/download/win

To begin, open Git bash and move to where you want to place the project on your local machine using the cd (change directory) command. In our case move to the G: drive.

Create a new folder/repo using the mkdir command and call it sweng: **mkdir sweng2018**

Move to this repo using the cd command: **cd sweng2018**

Next we need to initialise this repo for Git run the **git init** command:  **git init sweng2018**

# Step 2: Add a new file to the repo

Go ahead and add a new file to the project, using any text editor you like or running a [touch](http://linux.die.net/man/1/touch) command (this command creates an empty file): **touch helloworld.txt**

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](http://git-scm.com/docs/git-status) command to see which files git knows exist: **git status**

What this basically says is, "Hey, we noticed you created a new file called helloworld.txt, 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](http://git-scm.com/docs/git-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](https://git-scm.com/book/en/v2/Getting-Started-Git-Basics) come in. As seen in Step 2, 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**](http://git-scm.com/docs/git-add)**<filename>** 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**](http://git-scm.com/docs/git-commit) command.

# Step 3: Add a file to the staging environment

Add a file to the staging environment using the **git add** command: **git add helloworld.txt**

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.

# Step 4: Create a commit

It's time to create your first commit!

Run the command **git commit -m "Your message about the commit"** 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 "asdfadsf" or "foobar".

# Step 5: Create a new branch

Now that you've made a new commit, let's try something a little more advanced.

Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where [git branches](https://git-scm.com/book/en/v1/Git-Branching-What-a-Branch-Is)come in.

Branches allow you to move back and forth between 'states' of a project. For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project.

Once you're done with the page, you can [merge](http://git-scm.com/docs/git-merge) your changes from your branch into the master branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files.

Let's say you are on the master branch and want to create a new branch to develop your web page. Here's what you'll do:

Run[**git checkout -b <my branch name>**](http://git-scm.com/docs/git-checkout). This command will automatically create a new branch and then 'check you out' on it, meaning git will move you to that branch, off of the master branch.

After running the above command, you can use the [**git branch**](http://git-scm.com/docs/git-branch) command to confirm that your branch was created

The branch name with the asterisk next to it indicates which branch you're pointed to at that given time.

Now, if you switch back to the master branch (use the command **git checkout master**) and make some more commits, your new branch won't see any of those changes until you [merge](http://git-scm.com/docs/git-merge) those changes onto your new branch or vice versa.

## Try This!

Open helloworld.txt (from Windows Explorer (g:/sweng2018)) and add a few lines to it. Save the file:

Back in Git bash (make sure you are still in the sweng2018 repo):

cat helloworld .txt //command to view the file

git add helloworld.txt

git commit –m “testing merge”

git status

git branch testmerge //this is the command to create a new branch called testmerge

git branch //this command lets you see the branches that have been created. To delete a branch you use the command git branch –d <branchname>

git checkout testmerge //this command moves you to the working branch testmerge

ls //you should still see helloworld.txt

Open helloworld.txt again (from Windows Explorer (g:/sweng2018)) and add a few more lines to it. Save the file.

cat helloworld .txt //command to view the file

git add helloworld.txt

git commit –m “testing merge”

git status

git checkout master // this command moves you back to the master branch

ls //you should still see helloworld.txt

cat helloworld .txt //command to view the file. Can you see the new changes you made? No is the answer. You need to merge the branches for this to happen.

Git merge testmerge //this is the command for merging

Cat helloworld.txt //can you see the changes?