

# Creating a GitHub account

## GitHub

GitHub is the single largest host for Git repositories and a large percentage of all Git repositories are hosted on GitHub. Many open-source projects use it for Git hosting, issue tracking, code review, and other things. So while it's not a direct part of the Git open source project, there's a good chance that you'll want or need to interact with GitHub at some point while using Git professionally.

This section is about using GitHub effectively. We'll cover signing up for and managing an account, creating and using Git repositories, common workflows to contribute to projects and to accept contributions to yours, GitHub's programmatic interface and lots of little tips to make your life easier in general.

## Creating a GitHub repo

A repo is usually used to organise a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. I recommend including a README, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repo. It also offers other common options such as a license file.

The standard account in GitHub provides unlimited free repos that are public access. Public access means that everyone with a GitHub account can view your repo. This may not be suitable for students who wish to keep their repo work private from unknown GitHub users. Alternatively we can apply to GitHub for a student account. This type of account provides free unlimited public and private repos for a student user.

Let's set up our student GitHub account. Browse to <https://education.github.com/> and click **Request a discount**.

Enter your personal details. Your username can contain alphanumeric characters and hyphens – no spaces or "." Remember to make it readable as it will be used in links and navigation of all your projects. The username must be unique in GitHub – I use **jamesconnollyit**.

Enter your email address. It will be used later to verify your student account upgrade.

Enter a password – it should not be the same as your LYIT password, since this makes your email account vulnerable if GitHub is hacked in future.

## Creating a GitHub account

**Join GitHub**  
The best way to design, build, and ship software.

Step 1: Set up a personal account | Step 2: Choose your plan | Step 3: Tailor your experience

**Create your personal account**

Username  
jamesconnollyyt  
This will be your username — you can enter your organization's username next.

Email Address  
james.connolly@lyt.ie  
You will occasionally receive account related emails. We promise not to share your email with anyone.

Password  
••••••••••  
Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

**Create an account**

**You'll love GitHub**

- Unlimited collaborators
- Unlimited public repositories
- Great communication
- Frictionless development
- Open source community

Click **Create an account**.

On the next screen, choose the default plan of **Unlimited public repositories for free**.

Click **Continue**.

**Welcome to GitHub**  
You've taken your first step into a larger world, @jamesconnollyyt.

Completed: Set up a personal account | Step 2: Choose your plan | Step 3: Tailor your experience

**Choose your personal plan**

☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month. [\(view in EUR\)](#)

Don't worry, you can cancel or upgrade at any time.

☐ **Help me set up an organization next**  
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.  
[Learn more about organizations.](#)

**Both plans include:**

- Collaborative code review
- Issue tracking
- Open source community
- Unlimited public repositories
- Join any organization

**Continue**

At the Request a discount screen, choose the option that best describes you. This choice does not affect your entitlement to a free account.

Click **Next**.

## Creating a GitHub account

### Request a discount

Discounted and free plans are available for educational use

**Step 1**  
Tell us what you need

**Step 2**  
Tell us about you

**Which best describes you?**

☐ Student

☒ Teacher

☐ Researcher

☐ Administrator/staff

☐ Other

**What are you looking to get a discount for?** [?](#)

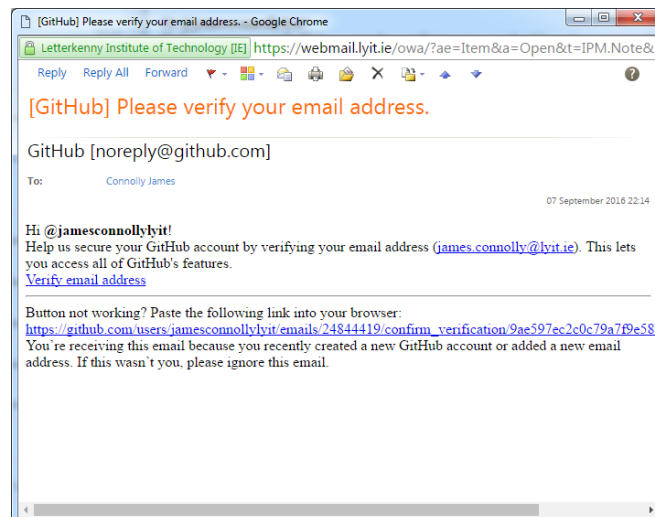
☒ Individual account

☐ Organization account

Next

Go to the email account you used when you started the registration process. This is the contact email address used by GitHub to verify your identity.

Once you clicked **Next**, you should have received an email from GitHub (example shown below).



Click on **Verify email address** inside this email. This redirects you back to the GitHub login window. Log in with your new GitHub credentials - note it may be easier to use your email address than the username you set up.

Click **Sign in**.

Congratulations – you’ve just set up your GitHub account

## Creating a GitHub account



Go back to the GitHub registration page and refresh it

Step 1 Tell us what you need	Step 2 Tell us about you
<p><b>Name</b></p> <input type="text" value="jamesconnollylyit"/>	
<p><b>Verify academic status</b></p> <p>Select your school-issued email address:</p> <input type="text" value="james.connolly@lyit.ie"/> <p>If your school-issued email address isn't listed, please <a href="#">add and verify it</a>, then refresh this page.</p>	
<p><b>School name</b></p> <input type="text" value="LYIT"/>	
<p><b>How do you plan to use GitHub?</b></p> <div> <p>For storing repositories and sharing repositories with students</p> </div>	
<p>Please note, your request cannot be edited once it has been submitted, so please verify your details for accuracy before sending them to us.</p>	
<p><a href="#">Submit Request</a></p>	

Your verified email address will now be shown under School-issued email address.

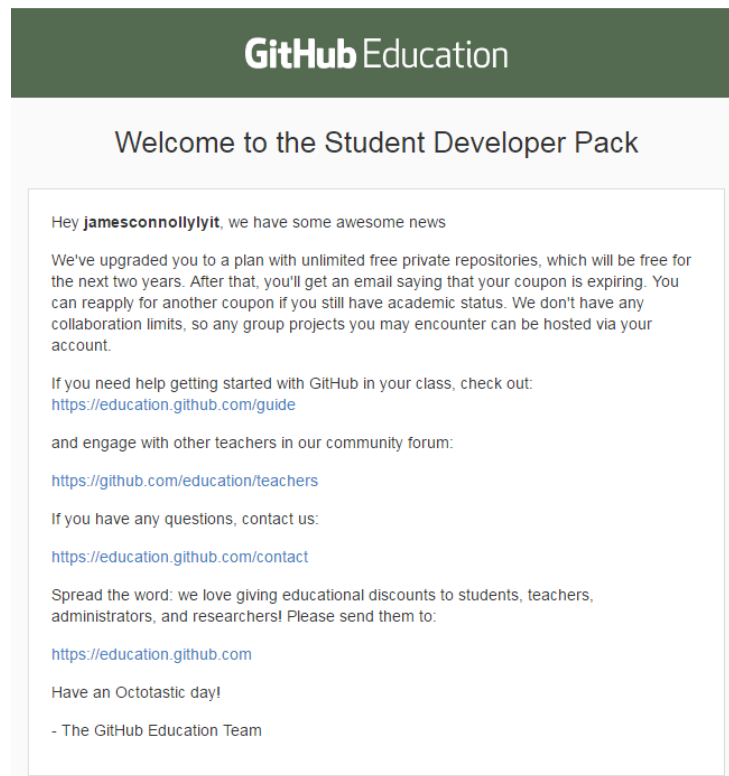
Key in the School name and how to plan to use GitHub. (I don't think this is read by a human).

Click **Submit Request**

The submission page advises you will get feedback within a few weeks.

Approximately 5-10 minutes later you should receive an email to verify approval of your account to student status. Check the email address you used for verification of your account for approval from GitHub. Be sure to have an **octotastic** day (read the email).

## Creating a GitHub account



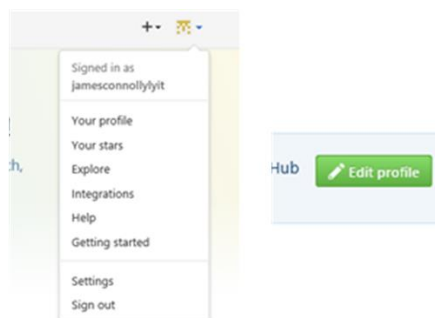
### GitHub – first look around

Log into your GitHub account through the account login page at <https://github.com>.

Once authenticated, you are shown the main GitHub dashboard for your user account.

There are 3 main sections to the dashboard – overview information, repo information and public activity.

Let's edit your profile. Click on the square icon (your avatar) to the top right of your GitHub window and click on **Your Profile**. Alternatively click on the **Edit profile** button.



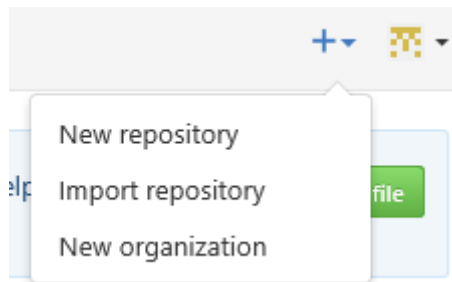
This section hold information on you, your activity and the projects you work on. In this section we can view profile settings – add / modify password, contact email address, notifications, followers and organisations.

Click the back button to return to your main GitHub page.

## Creating a GitHub account

### Create a new GitHub repo

In the upper right corner, next to your avatar, click + and then select New repo.



Call your new repo **Hello-world**.

Write a short description.

Leave the access settings for the repo at Public and select **Initialize this repository with a README**.

You'll notice we also have an option to add a .gitignore file. Let's leave it for now.

Click **Create repository**.

#### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: jamesconnollyit / Repository name: Hello-world

Great repository names are short and memorable. Need inspiration? How about **fictional-spork**.

Description (optional):

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

**Create repository**

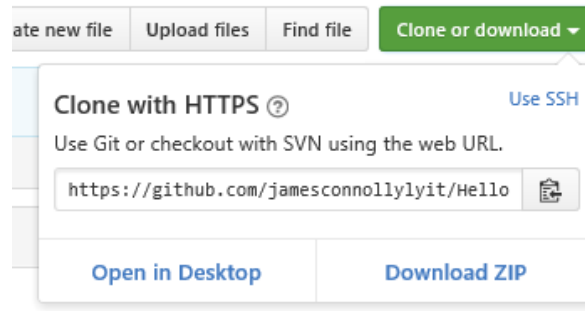
Now that your project is hosted on GitHub, you can give the URL to anyone you want to share your project with. Every project on GitHub is accessible over HTTP as `https://github.com/<user>/<project_name>`, and over SSH as `git@github.com:<user>/<project_name>`.

Git can fetch from and push to both of these URLs, but they are access-controlled based on the credentials of the user connecting to them.

## Creating a GitHub account

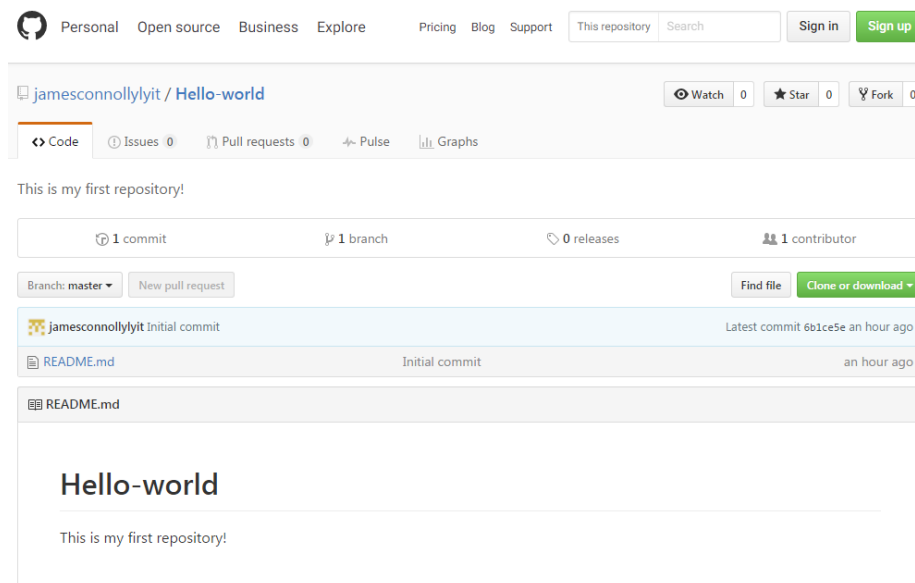
It is often preferable to share the HTTP based URL for a public project, since the user does not have to have a GitHub account to access it for cloning to a local repo. Users will have to have an account and an uploaded SSH key to access your project if you give them the SSH URL. The HTTP one is also exactly the same URL they would paste into a browser to view the project there.

Let's have a quick look at how our project will look like to the outside world. Click on **Clone or download** and choose the web URL in the textbox. Copy this text using **CTRL + C** keys.



Open another tab in your browser and paste the copied URL into the URL bar. Press **Return** to initiate the URL request.

Once opened, the hello-world repo should open and should look similar to this screen shot shown below.

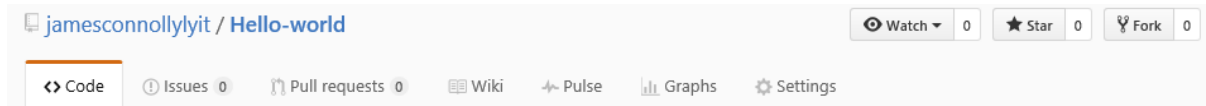


### Adding Collaborators

If you're working with other people who you want to give commit access to, you need to add them as "collaborators". Each collaborator needs to have an account with GitHub and you need to give them push access to your repo. This means they have both read and write access to the project and Git repo.

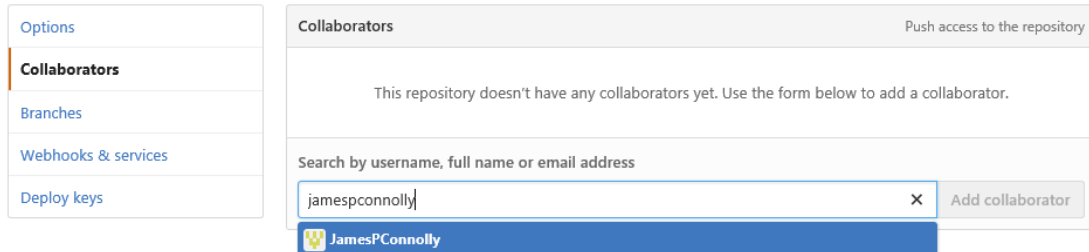
Click on the **settings** button towards the top of the GitHub window.

## Creating a GitHub account



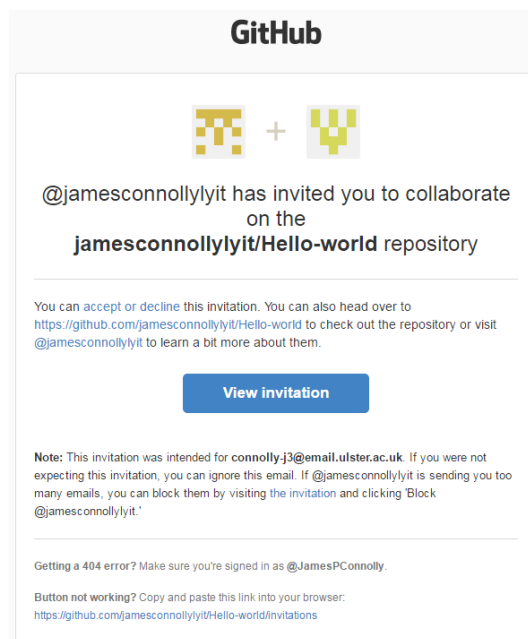
Select Collaborators from the side menu. Type in the username of the user you would like to add as a collaborator to your project.

Click **Add collaborator**



The username you added will appear as a potential collaborator – subject to acceptance by the user you added.

The user you added as a collaborator will receive an email. The user can accept or decline the offer, and they can view the project before accepting the collaboration offer.



## Branches

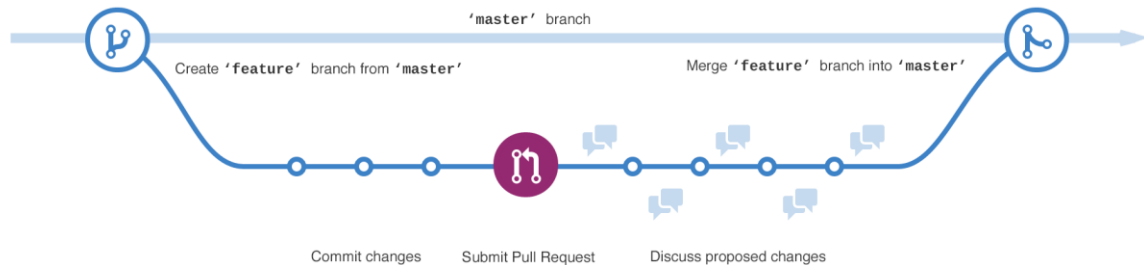
Earlier we discussed branching as a way to work on different versions of a repo at one time. By default your repo has one branch named **master** which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.



## Creating a GitHub account

When you create a branch off the master branch, you're making a copy, or snapshot of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

This diagram shows:



- The master branch
- A new branch called feature (because we're doing 'feature work' on this branch)
- The journey that feature takes before it's merged into master.

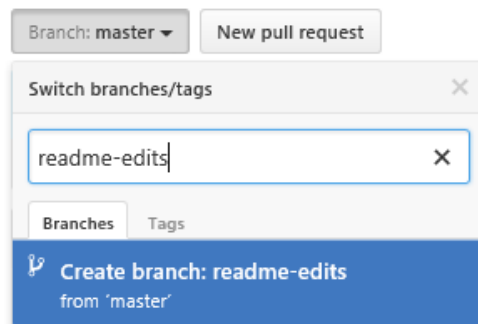
## Creating a new branch

To create a new branch, go to your new repo **hello-world**.

Click the drop down at the top of the file list that says branch: master.

Type a branch name, **readme-edits**, into the new branch text box.

Select the blue **Create branch** box or hit "Enter" on your keyboard.




Now we have two branches, **master** and **readme-edits**. They look exactly the same, but not for long. Next we'll add our changes to the new branch.

On GitHub, saved changes are called **commits**. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you've done and why. We used these messages earlier when we used the Git command line console.

## Make and commit changes

Click the **README.md** file.

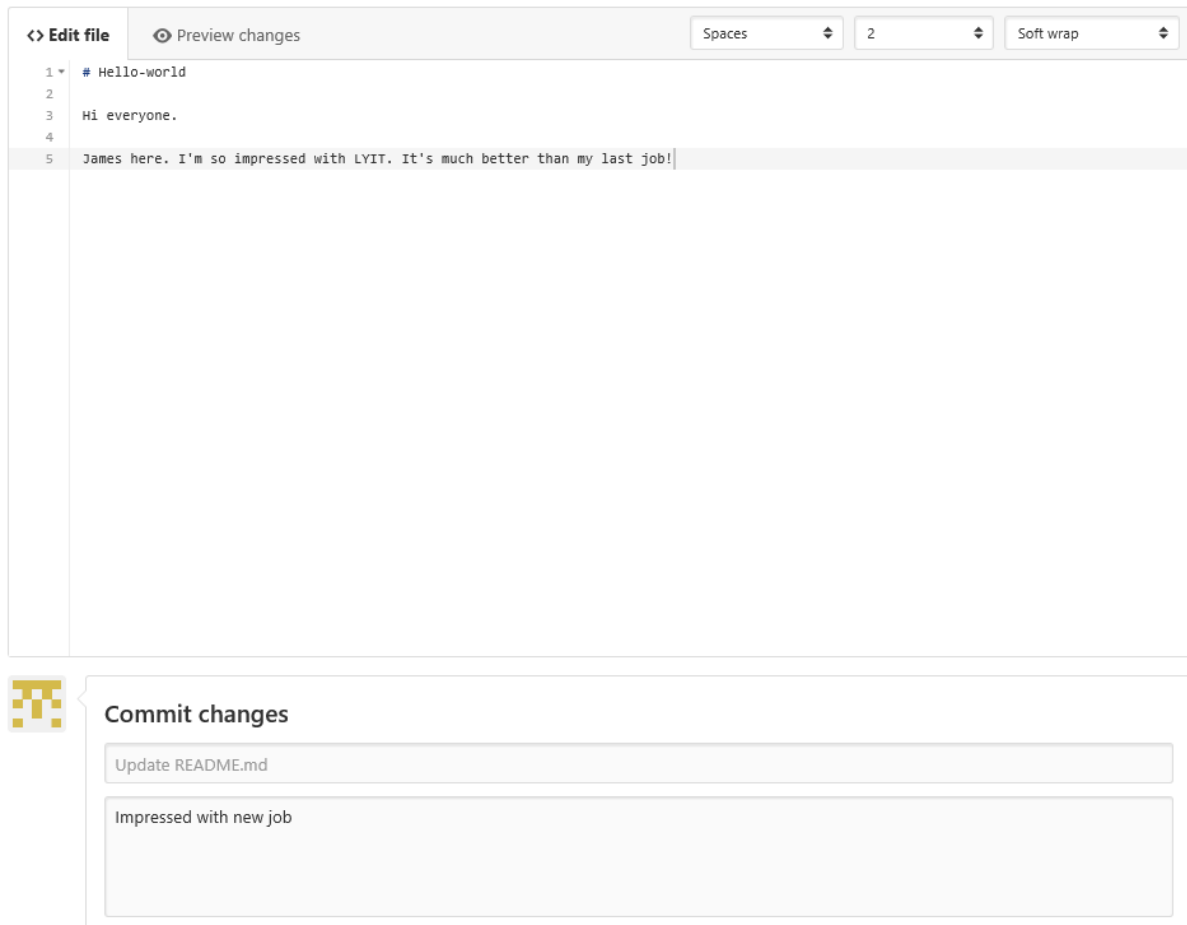
## Creating a GitHub account

Click the  pencil icon in the upper right corner of the file view to edit.

In the editor, write a bit about yourself.

Write a commit message that describes your changes.

Click **Commit changes** button.



These changes will be made to just the README file on your readme-edits branch, so now this branch contains content that's different from master.

### Open a Pull Request

Now that you have changes in a branch off master, you can open a pull request.

Pull Requests are the heart of collaboration on GitHub. When you open a pull request, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

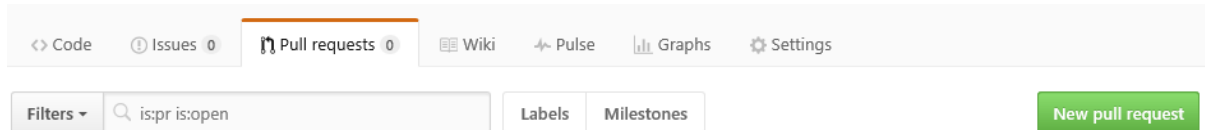
## Creating a GitHub account

By using GitHub's @mention system in your pull request message, you can ask for feedback from specific people or teams.

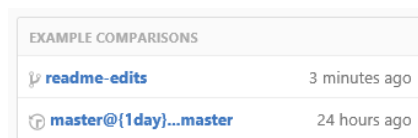
You can even open pull requests in your own repo and merge them yourself. It's a great way to learn the GitHub Flow before working on larger projects.

Open a Pull Request for changes to the README.

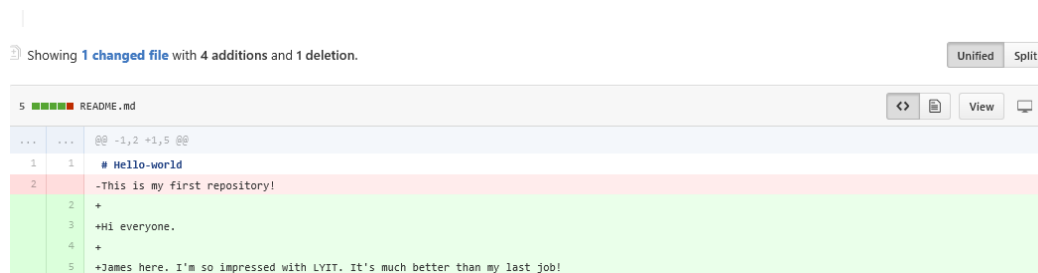
Click the **Pull Request** tab, then from the Pull Request page, click the green **New pull request** button.



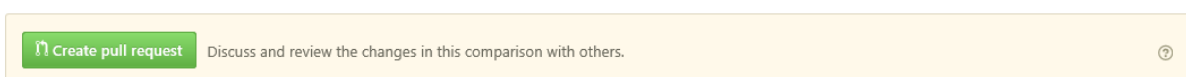
Select the branch **readme-edits**, to compare with master (the original).



Look over your changes on the Compare page, make sure they're what you want to submit.



When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.



Give your pull request a title and write a brief description of your changes. You can leave the title as it is if you prefer.

## Creating a GitHub account

Update to the readme file

Write Preview

Impressed with new job

Attach files by dragging & dropping or [selecting them](#).

Styling with Markdown is supported

Create pull request

Now click **Create pull request**.

Finally we bring your changes together by merging your readme-edits branch into the master branch.

Click the green **Merge pull request** button to merge the changes into master.

✓ This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Click **Confirm merge**.

Merge pull request #1 from jamesconnollyit/readme-edits

Update to the readme file

Confirm merge Cancel

Now we can delete the branch, since its changes have been incorporated into our master branch. Click the **Delete branch** button.

Congratulations – you’ve now merged your pull request into the master branch.

## Creating a GitHub account

### Merging our local Git repo into GitHub

Earlier we worked on the Git command line to create a local repo. Now let's copy up our local Git repo to GitHub.

Open the Git Bash command prompt and navigate to the working directory of your local project.

Add the files to your local repo if you haven't already done so by using **git add**.

Commit these files that you've just staged in your local repo using the command **git commit -m "Copy files to GitHub"**. Use **git reset --soft HEAD~1** and commit and add the files again if you make a mistake during this process.

Go to your GitHub dashboard and create a new repo called **Local-Code**.

Optionally you can type in a description. Do not choose Initialise this repo with a README since you may already have a readme file in your local repo that you will copy to GitHub.

Click **Create repository**.

**Create a new repository**  
A repository contains all the files for your project, including the revision history.

Owner: jamesconnollyit / Repository name: Local-Code ✓

Great repository names are short and memorable. Need inspiration? How about **silver-rotary-phone**.

Description (optional):

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

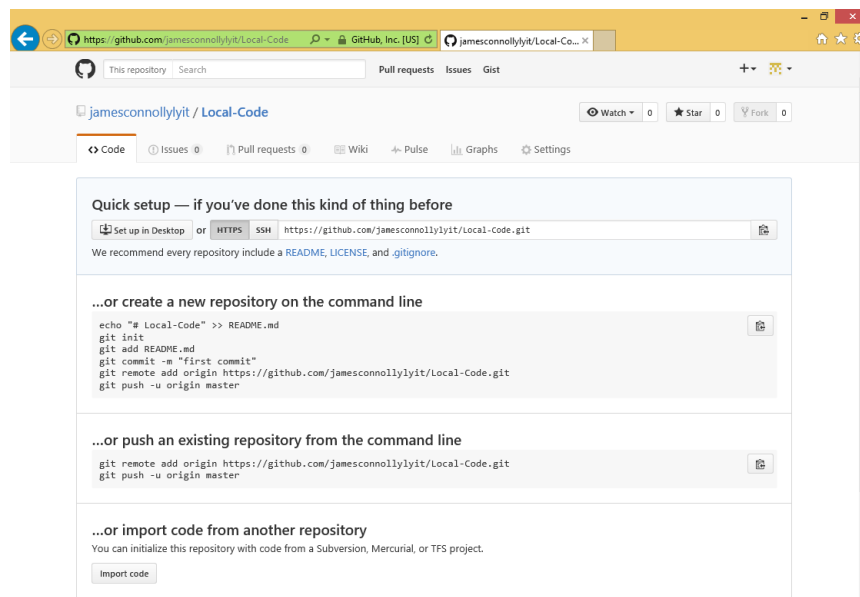
☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

**Create repository**

You will notice GitHub recognises that you've already created your first repo and so it now presents you with a different screen after the **Local-Code** repo is created.

## Creating a GitHub account



At the top of your GitHub repo is a **Quick Setup** page, click  to copy the remote repo URL.

Alternatively copy the code below the title **...or push an existing repository from the command line** to the clipboard.

Open the Git command prompt and key in **git remote add origin remote repo URL**. For example the URL to my newly created repo is <https://github.com/jamesconnollylyit/Local-Code.git>. The command I enter to the command line to remotely upload my local repo to GitHub is

```
git remote add origin https://github.com/jamesconnollylyit/Local-Code.git
```

If you get an error **fatal: remote origin already exists**, you have already connected your local repo to one in GitHub. You can verify this using the command **git remote -v**.

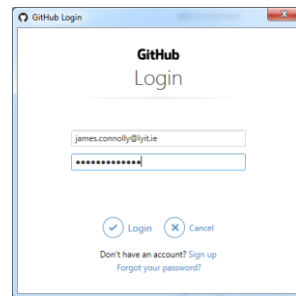
To fix the problem you need to remove this connection using the command **git remote rm origin**. Verify the remote connection has been deleted using **git remote -v**. Now you're ready to re-run the command shown above.

Verify the remote URL using **git remote -v**.

Push the changes in your local repo to GitHub using **git push origin master**.

You may need to authenticate to GitHub during this process. If so, key in your email address and password you used when you created your GitHub account and click Login.

## Creating a GitHub account



Once complete, your console will show a summary of the files you copied to the GitHub repo.

A screenshot of the Git CMD console window. The window title is 'Git CMD'. The command prompt shows the following sequence of commands and output:

```
c:\GitRepos>git remote add origin https://github.com/jamesconnollylyit/Local-Code.git
c:\GitRepos>git push origin master
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 957 bytes | 0 bytes/s, done.
Total 11 (delta 0), reused 0 (delta 0)
To https://github.com/jamesconnollylyit/Local-Code.git
 * [new branch]      master -> master
c:\GitRepos>
```

Click on the name of your repo in GitHub to verify the files now exist in your remote repo. The repo you copied to GitHub now holds an exact copy of your local repo, including all changes made to it since it was first initialised.

Click on the Commits button to see a history of repo commits.

Click on Graphs to view some of the visualisation options available to you.