
output:
pdf_document: default

html_document: default

Task 1: How to set up a repository on GitHub

This task is designed for students and researchers who want to create their first Open Source project (software or non-software) on GitHub. GitHub is a place for you to come and play and experiment with new research workflows, and is really just the beginning to help set the stage for your own pathways and ideas.

Don't forget you can join in the discussions over at our open [Slack channel](#). Please do introduce yourself at #module5opensource, and tell us a bit about who you are, your background, and how you ended up here!

Estimated time to complete: 30-45 minutes.

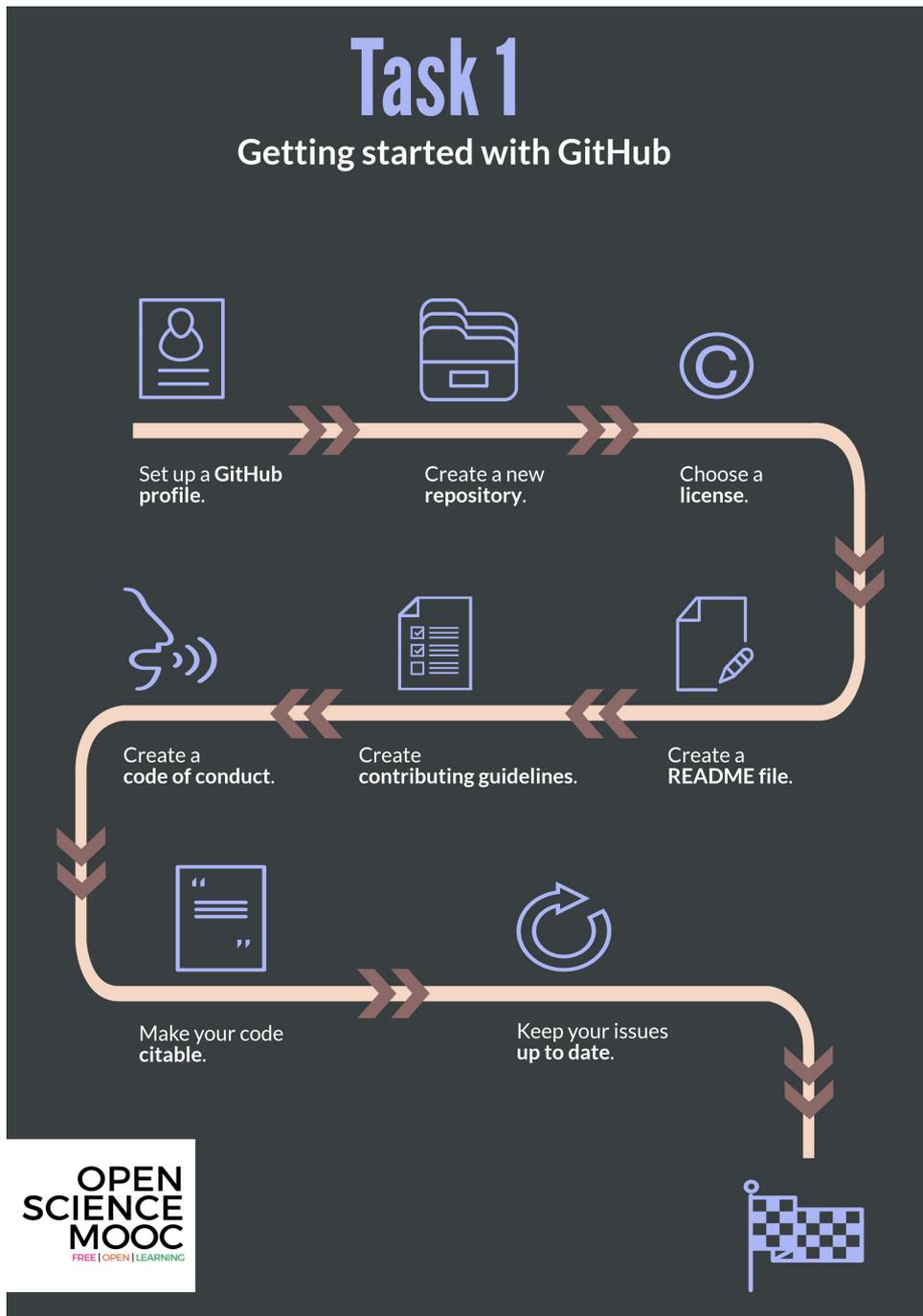
Estimate time saving once complete: Unimaginable..

Table of contents

- [Getting started](#)
 - [Setting up a GitHub profile](#)
 - [The GitHub language](#)
 - [Creating a new repository](#)
- [The foundational steps](#)
 - [Choosing a license](#)
 - [Creating a README file](#)
 - [Creating contributing guidelines](#)
 - [Creating a code of conduct](#)
 - [Making your code citable](#)
- [Keeping your issues up to date](#)
- [Check-list for launching your project](#)

Task 1

Getting started with GitHub



The workflow for Task 1. Keep this handy as you work through the task!

Getting started

A 'repository' is really just a fancy name for a project on GitHub. GitHub is a place online where you can manage projects, store files, and openly collaborate with others. This is all achieved by using version control to track projects as they progress. As such, GitHub is a powerful tool for both software and non-software projects.

One of the most important things to consider at this early stage is to think about how you want the wider community to interact with your project. As you are working in the open, you want to make sure others feel comfortable in accessing, viewing, and engaging with your work. Setting up a repository in a way that lowers the barriers to entry, and the fear of being an 'outsider' is the first step towards maintaining a successful project.



Octocat, GitHub's little mascot

Setting up a GitHub profile

To set up a GitHub profile, simply head to the main page and click [Sign Up for GitHub](#). Here, you can create your personal account, with a username, email, and password as standard.

Username
Pick a username

Email
you@example.com

Password
Create a password

Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

Sign up for GitHub

The next step is to set up a personal plan. For now, simply select the 'Unlimited public repositories for free' plan, unless you are concerned about privacy, in which case select the private plan. If you intend to set up a project for an organisation, you can select that option too.

The GitHub language

This is possibly the most confusing and off-putting aspect of GitHub. Here are some of the most commonly used terms and their definitions:

- **Initialise:** Create an empty repository.
- **Checkout:** Create a working copy of a local repository.
- **Clone:** Copy the repository into a local directory on your computer.
- **Fork:** Create a personal offshoot of a repository to work on it in parallel.
- **Branch:** An independent and parallel version of a repository. Changes do not affect the master branch.
- **Master:** The main and default branch for a repository.
- **Clean:** No commits pending on the branch.
- **Stage:** Add updates ready to be committed to a branch.
- **Commit:** A revision to a repository, like a versioned 'save' function.
- **Commit message:** A description of changes accompanying a commit.

- **Check:** A status check.
- **Fetch:** Nothing to do with dogs. Refers to getting the latest changes from an online repository without merging them.
- **Index:** The 'tree' which acts as a staging area.
- **Working Directory:** The 'tree' where the files are kept.
- **Head:** The 'tree' which indicates the last commits made.
- **Push:** Add committed changes to the head of your remote repository.
- **Merge:** Combining the changes made in one branch back with the master branch upon completion.
- **Pull:** Update your repository by fetching and merging the newest commits.
- **Pull request:** A request to merge an updated branch into the master branch.
- **Issue:** Suggested improvements, tasks, or questions related to a repository.

Whew! Don't worry about memorising *all* of these for now. Like any new skill, familiarity comes with experience.

You can probably see how some of these are fairly similar to things like save, copy, paste - standard workflow operations, but adapted for a software management process. There are a [few more](#) too, but this should do for getting started.

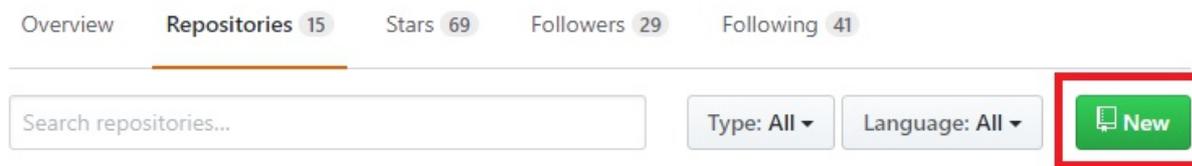
If you are interested, most of these terms come from the underlying [Git system](#). Git was built to allow developers to manage different versions of source code in a distributed manner, which is great. It has lots of features and the ability to do lots of complex stuff, written by a [very clever guy](#). However, the [user interface was not designed with new users in mind](#), so it can be hard to learn.



Unbeatable guide to using Git. (Source: XKCD)

Creating a new repository

On your GitHub profile, click the 'Create new repository'. The first step is to create a name as the brand for your project. Ideally, it should be memorable and give some indication of what the project does.



Open-Scholarship-Strategy

This document aims to agree on a broad, international strategy for the implementation of open scholarship that meets the needs of different national and regional communities but works globally.

★ 6 5 CC-BY-4.0 Updated a day ago

[Create a new repository](#)

Make sure not to duplicate names, infringe upon other trademarks, or name it anything that could be considered to be offensive.

The foundational steps

Any GitHub repository requires 4 key elements to get started and to begin developing a welcoming community:

1. An Open Source license;
2. A README file;
3. Contributing guidelines; and
4. A Code of Conduct.

These are critical aspects and best practices of any project for users to understand their legal rights, their expectations, the purpose of the project, and to improve the overall user experience.

All four of these files should be kept in the root directory for your project repository. It is convention to use markdown file formats (`.md`) for most of these files (though the license file is most often plain text (`.txt`)), and capitalise all file names. Instead of spaces in file names, make sure to use underscores `_` .

So you should end up with a foundational file selection like this:

1. LICENSE.md
2. README.md
3. CONTRIBUTING.md
4. CODE_OF_CONDUCT.md

The screenshot shows the GitHub repository page for 'Protohedgehog / Open-criteria'. At the top, there are navigation tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. Below the repository name, there are statistics: 10 commits, 2 branches, 0 releases, 1 contributor, and GPL-3.0 license. A table of recent commits is visible, with a red box highlighting the first four commits: 'Create CODE_OF_CONDUCT.md', 'Create CONTRIBUTING.md', 'Rename LICENSE to LICENSE.md', and 'Update README.md'. Below the commit history, there is a section for 'README.md' with an edit icon.

The basic repository structure

Choosing a license

Choosing an appropriate license is what will differentiate your Open Source repository from publicly available software. While you are not obliged to choose a license, doing so guarantees that others will be able to modify, share, re-use, and build upon your project within a legal framework.

To start with, you want to check [Choose A License](#) to find a license that best suits your intentions for the repository.

The three primary ones to choose from are:

- **MIT License**: A permissive license that lets people do whatever they want with your code as long as they provide appropriate attribution to you, and do not hold you liable.
- **Apache License 2.0**: Similar permissions to the MIT License, but also provides an express grant of patent rights from contributors to users.
- **GNU General Public License (GPL) v3**: A copyleft license that requires anyone who redistributes your code, or a derivative work, to make the source available under the same terms as the original license; also provides an express grant of patent rights from contributors to users.

Thankfully, when you start a new repository on GitHub, you are given the option to select an existing license from a drop-down menu. You should always (with very few exceptions) use an existing license, since this is what potential users and contributors will see before they choose to use or contribute to your software.

Protohedgehog / Open-criteria

Unwatch 3 Unstar 3 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master Open-criteria / LICENSE.md Find file Copy path

Protohedgehog/Open-criteria is licensed under the **GNU General Public License v3.0**

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

This is not legal advice. [Learn more about repository licenses.](#)

Permissions	Limitations	Conditions
<ul style="list-style-type: none"> ✓ Commercial use ✓ Modification ✓ Distribution ✓ Patent use ✓ Private use 	<ul style="list-style-type: none"> ✗ Liability ✗ Warranty 	<ul style="list-style-type: none"> ① License and copyright notice ① State changes ① Disclose source ① Same license

Choosing an example license

If they don't have one you want, you can add one you like manually. To do this, simply click 'Create new file' in the repository, and copy and paste an existing license text in. Name the file something like LICENSE.txt or LICENSE.md to make it clear, and keep it in the main repository folder (i.e., the root). Make sure to add a clean commit message, and you're done!

Helping hand: This MOOC uses a different combination of licenses for code content and non-code content. Here you can find an example of the [MIT License](#) that we apply for all code and software generated as part of the MOOC production.

Creating a README file

When you initialise your new repository, there should be an option to do so with a README file. Just like Alice in Wonderland, these do exactly what they say - provide key information about the project. These are typically the first thing outside contributors will see when they come to your repository, so making them informative and welcoming is key.

Code Issues 9 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master Module-5-Open-Research-Software-and-Open-Source / README.md Find file Copy path

protohedgehog Updated readme, added first images to task 1 95f5292 34 seconds ago

3 contributors

62 lines (39 sloc) | 4.14 KB Raw Blame History

Module 5: Open Research Software and Open Source

Welcome to Module 5 of the Open Science MOOC! Here you can find important information on the development of the module, including the latest updates to the content.

- **Content Development** - This is where you can find all content currently being developed for the MOOC, and also contribute yourself.
- **Production Toolkit** - This is where the basic protocols and outline for the module development are kept. It includes a tracking scheme as the content development progresses.

Don't forget to join us in our open [Slack group](#)! The channel for this module is #module5opensource. You can also sign up to our mailing list [here](#).

Part of the README file for this module

The file will originally be in markdown (.md) format. This is a lightweight markup language with a plain text format. To learn some basic markdown, see [this cheatsheet](#). But for now, we can just use plain text.

There are several things you will want to include in your `README` file:

- What is this project about and what does it do.
- Why should people care, and why is it useful.
- How can someone get started contributing to the project.
- Who can be contacted in case someone needs help.
- A link to the license, contributing guidelines, and code of conduct.
- A description of the project structure.
- Who is involved, and what are their roles.
- The current status of the project.

Pro-tip: Later on as your project develops, you might want to add FAQs based on community feedback, or a tutorial to help users understand how your project works.

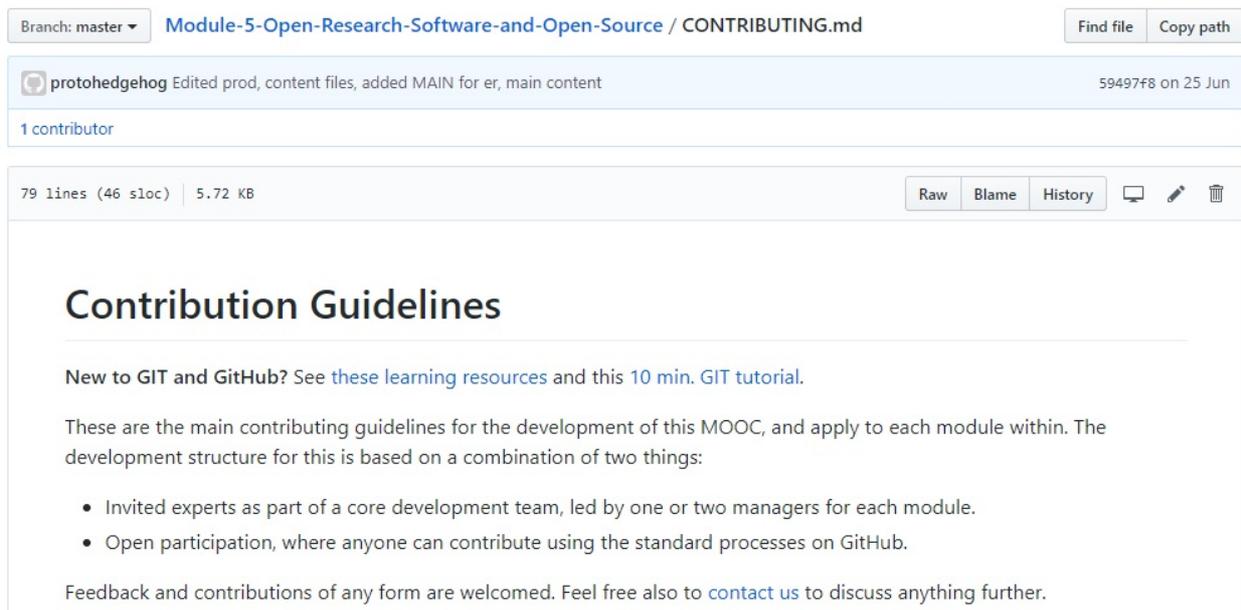
Remember that not everyone coming to your project will be an expert, or understand what it is you are doing and why. Having a well-documented `README` file will enhance the user experience for people with a range of prior knowledge.

When the `README` file is included in the root directory, GitHub will automatically display this on the homepage of your repository. This means it is the first thing people will often see, so make it count!

Helping hand: Here, you can find the `README` file used for this [MOOC module](#). This includes information on the status, rationale, learning outcomes, development team, key documents, and license to help. You can copy and adapt this structure for your own projects as needed.

Creating contributing guidelines

Contributing guidelines are designed to communicate to potential contributors a short guide on how to engage with your project and community. You want to make sure to be welcoming, and indicate that you are eager for participants to engage with your project. Whenever a participant opens a new pull request or creates a new issue, they will see a link to your contribution file.



Branch: master | [Module-5-Open-Research-Software-and-Open-Source](#) / CONTRIBUTING.md | Find file | Copy path

protohedgehog Edited prod, content files, added MAIN for er, main content | 59497f8 on 25 Jun

1 contributor

79 lines (46 sloc) | 5.72 KB | Raw | Blame | History | [Icons]

Contribution Guidelines

New to GIT and GitHub? See [these learning resources](#) and this [10 min. GIT tutorial](#).

These are the main contributing guidelines for the development of this MOOC, and apply to each module within. The development structure for this is based on a combination of two things:

- Invited experts as part of a core development team, led by one or two managers for each module.
- Open participation, where anyone can contribute using the standard processes on GitHub.

Feedback and contributions of any form are welcomed. Feel free also to [contact us](#) to discuss anything further.

Part of the `CONTRIBUTING` guidelines for this module

Sticking with the all caps file names, the next step is to create a `CONTRIBUTING` file. Click 'Create new file', and make sure to save it in markdown format as before. This file will tell other users how they can engage with and participate in your project. This is the first step towards establishing a community around your project, so make it engaging, concise, and informative.

The `CONTRIBUTING` file should include information on:

- What sort of contributions you are looking for.
- How to suggest updates or new features.
- How to interact with the project using GitHub's functions (e.g., the pull request protocol).
- How to file a bug report or create an issue.
- The ultimate goal, vision, or roadmap for the project.
- How to contact those in charge of the project.

- Links to any external documentation or websites.

Pro-tip: Consider starting off with a short thank you note for people taking the time to consider contributing - they have clicked on the file to learn more after all! If there are other methods of recognition that you have in mind, make sure to include them in here too.

Here, you are essentially trying to encourage people to volunteer their time to advance your project. Make sure to be welcoming and friendly, and be precise about how people can engage. When writing this, make sure to think about it from the user perspective - how can you make their life easier when submitting pull requests and opening issues to make the whole project run more smoothly.

Helping hand: The [Contributing guidelines](#) for this MOOC module include some very specific things: an introduction to using Git and GitHub, tips for getting started, contact information, how to alter the content and report issues, a link to the `README` file, and information on the preferred content and code styles. Feel free to copy and adapt this for your own project as needed.

Creating a Code of Conduct

A code of conduct is important for setting the ground rules for expected behaviour and participation for project contributors, and is an easily referenced document for showing that your project team takes constructive dialogues seriously. Therefore, it is a critical element for creating and maintaining a healthy community that engages in a constructive and productive manner within a positive social atmosphere.

A code of conduct not only provides expectations of behaviour, but also describes who those expectations apply to, when they apply, what to do should a violation of the code occur, and what the action items for this will be. As such, points of contact need to be made clear in the code of conduct. Typically, this should be in a private way such as an email address.

Pro-tip: In case a violation needs to be reported about the person who receives those reports, make sure to include an option to contact a secondary party.

To add a code of conduct, you can create your own from scratch by adding a new markdown file, or use existing templates such as the [Contributor Covenant](#). Name your file `CODE_OF_CONDUCT.md`, and make sure it is visible in the `README` file.

Helping hand: This MOOC also has a [Code of Conduct](#) based on the Contributor Covenant. As you can see, it includes information on expected standards of behaviour, responsibilities of those in the community, and enforcement of the CoC including contact details. Feel free again to re-use and adapt this to your project as you see fit.



The screenshot shows a GitHub repository interface. At the top, it indicates the branch is 'master' and the file path is 'Module-5-Open-Research-Software-and-Open-Source / CODE_OF_CONDUCT.md'. There are buttons for 'Find file' and 'Copy path'. Below this, a commit by 'Protohedgehog' is shown, creating the file 'CODE_OF_CONDUCT.md' with commit hash 'a721b40' on '18 Apr'. It lists '1 contributor'. The file details show '75 lines (56 sloc)' and '3.27 KB'. There are buttons for 'Raw', 'Blame', 'History', and icons for a monitor, edit, and delete. The main content of the file is displayed, starting with the title 'Contributor Covenant Code of Conduct' and a section 'Our Pledge' which reads: 'In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.'

Part of the CODE OF CONDUCT file for this module, based on the Contributor Covenant

Making sure to enforce the code of conduct is important, as it shows that not only do you value the code, but you respect the influence that it has on your community. It is important to treat each member of the community with the respect, courtesy, and importance that they deserve. Should a violation occur, or a repeat offender makes consistent violations, it is best to refer to the [Open Source Guide](#) to see how to enforce the code of conduct.

Making your code citable

If you want to make your code citable from the start, you should store the metadata needed for a citation from the start, by creating a `[codemeta.json]`

(<https://codemeta.github.io>) file or a [CITATION.cff](<https://citation-file-format.github.io>) file. Both will allow tooling that is currently being developed to automatically create citation information, rather than asking you to type it in a form later.

If you're interested, cite.research-software.org provides further background information about software citation in academia.

Keeping your issues up to date

Issues are not necessarily problems with a project, but also suggestions for improvement, things to develop in the future, and comments and feedback about the project to work through. They can be openly shared and discussed with contributors as needed, sort of like a forum.

If you are a project lead, it is important to maintain a list of issues that make it clear to contributors what aspects of the project need attention. It is also important to engage with as many issues as possible from others in a positive manner, to show that you take their contributions seriously.

Key elements for issues include:

- An informative title and description;
- Coloured labels/tags to help categorise and filter;
- Milestones to associate issues with specific features or project phases;
- Assignees indicate who is responsible for working on an issue;
- Comments for providing feedback.

The screenshot shows the GitHub interface for the 'Open-Scholarship-Strategy' repository. At the top, there are navigation links for 'Code', 'Issues' (5), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Insights', and 'Settings'. On the right, there are buttons for 'Unwatch' (5), 'Unstar' (9), and 'Fork' (10). Below the navigation, there is a search bar with the filter 'is:issue is:open' and buttons for 'Labels' and 'Milestones'. A 'New issue' button is also present. The main content area displays a list of 5 open issues, each with a title, a label (e.g., 'second draft', 'enhancement'), and a comment count. The issues are: '#27 New text for pharma industry status' (second draft, 1 comment), '#26 Could we add a strong finish?' (second draft, 1 comment), '#13 Resistance to change, thinking about behavior change more generally' (second draft, 1 comment), '#8 Better visualisation of the site' (enhancement, 16 comments), and '#2 Spelling and grammar' (enhancement, 0 comments).

The issue tracker for the Open Scholarship Strategy project

Within issues it is possible to use @ mentions to notify other contributors about the issue, and to get the right people engaged in an effective manner. GitHub has an internal system of notifications, just like Facebook or Twitter, and can also send emails to people who are mentioned in the issue tracker. This can all be customised for individuals within the user settings.

Checklist for launching your project

So now you are ready to launch your project, begin advertising it, and getting contributions! Before continuing, make sure that you have:

- Project has a memorable and informative name
- Project has a LICENSE file that is an exact copy of an Open Source license
- Complete documentation including a README, CONTRIBUTING, and CODE_OF_CONDUCT files
- Project has at least one clearly labelled issue
- Any code included at this stage is clearly structured and annotated

CONGRATULATIONS!

You have now launched an Open Source research project! Hopefully, from here on out, your work will act to benefit the wider community, forge new collaborations, and create new and fantastic opportunities for you all. Try and think about ways in which these skills can be applied to future projects, and how they might also have helped with some in the past.

From now on, it is all up to you! Some advice is to:

- Write clean code;
- Have a well-structured project;
- Make frequent commits with clean messages;
- Keep projects well-documented;
- Have clear contributing guidelines;
- Make use of the description and tag functions;
- Don't fork someone else's repository unless you intend to work on them;
- Make sure to contribute to other people's projects too.

Know a way this content can be improved?

Time to take your new GitHub skills for a test-run! All content development primarily happens [here](#). If you have a suggested improvement to the content, layout, or anything else, you can make it and then it will automatically become part of the MOOC content after verification from a moderator!