Intro to OTTR - Opensource Tools for Training Resources

March, 2022

Table of Contents

# About this Course

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Recommended background information](#recommended-background-information)
* [How to contribute](#how-to-contribute)
* [Basic overview of what the publishing process looks like:](#X709578747fffe4fc45ed983c6ac16261d599dcd)
* [Table of contents](#table-of-contents)
  + [Getting started](#getting-started)
  + [The tools in this template](#the-tools-in-this-template)
  + [Platform specific set up](#platform-specific-set-up)
  + [Automation info and set up](#automation-info-and-set-up)
  + [Finishing touches](#finishing-touches)
  + [Release Notes](#release-notes)
  + [Help](#help)
  + [How to contribute to ITN courses](#how-to-contribute-to-itn-courses)
* Please take a look at the [code of conduct](https://github.com/jhudsl/OTTR_Template/blob/main/code_of_conduct.md).
* If you encounter any problems or have ideas for improvements to this template repository or this getting started guide, please [file an issue here](https://github.com/jhudsl/OTTR_Template/issues/new/choose)! Your feedback is very much appreciated.

## 0.1 Recommended background information

* If you are not familiar with **Git and Github**, we recommend going through these chapters from our Reproducibility courses for quick orientation:
  + [Why GitHub](https://jhudatascience.org/Reproducibility_in_Cancer_Informatics/making-your-project-open-source-with-github.html)
  + [Using GitHub](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/using-version-control-with-github.html)
* If you aren’t familiar with **markdown** [this is a nice introduction](https://www.markdownguide.org/getting-started/).
* For information on **RMarkdown** (a variation of a markdown file that we use here): [RStudio has their lessons here](https://rmarkdown.rstudio.com/lesson-1.html).

## 0.2 How to contribute

* If you are looking to write your own course go to: [“Start a new course”](https://github.com/jhudsl/OTTR_Template/wiki/Start-a-new-course) and start setting up your course repository. After creating your new template, issues will be filed automatically. Follow these issues and the links in them to get going.
* If you want to contribute in ways that are not your writing your own course, you can help us with our courses by:
  + [Reviewing existing content](https://github.com/jhudsl/OTTR_Template/wiki/Reviewing-existing-content)
  + [Proposing new content](https://github.com/jhudsl/OTTR_Template/wiki/Proposing-new-content)
  + [Writing new content](https://github.com/jhudsl/OTTR_Template/wiki/Writing-new-content)
  + [Fixing errors and bugs](https://github.com/jhudsl/OTTR_Template/wiki/Fixing-errors-and-bugs)

## 0.3 Basic overview of what the publishing process looks like:

## 0.4 Table of contents

#### 0.4.0.1 Getting started

* [Getting started](https://github.com/jhudsl/OTTR_Template/wiki/Getting-started)
* [Start a new course!](https://github.com/jhudsl/OTTR_Template/wiki/Start-a-new-course)
* [Start editing course files](https://github.com/jhudsl/OTTR_Template/wiki/Start-editing-your-course)
* [Content creation tips](https://github.com/jhudsl/OTTR_Template/wiki/Content-creation-tips)
* [Setting up images and videos](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-images-and-graphics)

#### 0.4.0.2 The tools in this template

* [About citations](https://github.com/jhudsl/OTTR_Template/wiki/Citations)
* [About Docker](https://github.com/jhudsl/OTTR_Template/wiki/Using-Docker)
* [Spell check](https://github.com/jhudsl/OTTR_Template/wiki/Spell-check)
* [Leanpub conversion tips](https://github.com/jhudsl/OTTR_Template/wiki/Leanpub-conversion-tips)
* [Generate docx output](https://github.com/jhudsl/OTTR_Template/wiki/Generate-docx-output)
* [Borrowing chapters between courses](https://github.com/jhudsl/OTTR_Template/wiki/Borrowing-chapters-between-courses)

#### 0.4.0.3 Platform specific set up

* [Choosing between platforms](https://github.com/jhudsl/OTTR_Template/wiki/Choosing-publishing-platforms)
* [Publishing with Bookdown](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-with-Bookdown)
* [Publishing on Leanpub](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Leanpub)
* [Publishing on Coursera](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Coursera)
* [Making quizzes private (Leanpub and Coursera)](https://github.com/jhudsl/OTTR_Template/wiki/Making-quizzes-private)

#### 0.4.0.4 Automation info and set up

* [GitHub actions](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots)
* [Receiving template updates](https://github.com/jhudsl/OTTR_Template/wiki/Receiving-template-updates)

#### 0.4.0.5 Finishing touches

* [Set up user feedback method](https://github.com/jhudsl/OTTR_Template/wiki/Set-up-feedback-method)
* [Change title and style](https://github.com/jhudsl/OTTR_Template/wiki/Change-Title-and-style)
* [Credits section](https://github.com/jhudsl/OTTR_Template/wiki/How-to-give-credits)
* [Adding Google Analytics Traffic Tracking](https://github.com/jhudsl/OTTR_Template/wiki/Google-Analytics)
* [Release a course for public viewing](https://github.com/jhudsl/OTTR_Template/wiki/Release-a-course-for-public-viewing)

#### 0.4.0.6 Release Notes

* [Template release notes](https://github.com/jhudsl/OTTR_Template/releases)

#### 0.4.0.7 Help

* [Getting help](https://github.com/jhudsl/OTTR_Template/wiki/Getting-Help-(Google-Group))

#### 0.4.0.8 How to contribute to [ITN](https://www.itcrtraining.org/) courses

* [Reviewing existing content](https://github.com/jhudsl/OTTR_Template/wiki/Reviewing-existing-content)
* [Proposing new content](https://github.com/jhudsl/OTTR_Template/wiki/Proposing-new-content)
* [Writing new content](https://github.com/jhudsl/OTTR_Template/wiki/Writing-new-content)
* [Fixing errors and bugs](https://github.com/jhudsl/OTTR_Template/wiki/Fixing-errors-and-bugs)

# 1 Getting started

For any OTTR questions we suggest joining the [OTTR Google Group](https://github.com/jhudsl/OTTR_Template/wiki/Getting-Help-(Google-Group)) and adding a discussion there about the problems you are encountering or questions you have.

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [1. Create a repository from this template](#X223c2422e355b36cd77013f32a0d49e438f9cb1)
* [2. Name your repository and fill in a short description.](#X865cce9853e28a7d74a6a39ad5d221cf87de826)
* [3. Your course must be set to public for Github actions to work.](#X8c9563f428d1f32872a82e5663e15ccf9807895)
* [4. Set up a Git client](#X745b2f07287e8195dadbe8a93231a4be1473cc2)
* [5. Add jhudsl-robot as a collaborator](#X2cfe5fabdc04448574bca37fb67a6b23eff2c51)
* [6. Set up your Github Personal access token](#X3ae5861a3e27273ffeb4dfd905d8f5e4dcb20b2)
* [7. Set up GitHub pages](#X30d74c876dc651e5188c77dcfd87e2ccb213e95)
* [8. Set up branch rules](#X81c7b0ee7532d01244ed45ce5cabadc0a9d8be9)
* [9. Enroll your repository for OTTR updates](#Xf6f7ca9d001ceed0abe73e8892f00976cbeaf4d)

Upon creating your repository, issues for getting your course started will be automatically filed in your repository. Go to Issues and you can follow them along to get set up.

### 1.0.1 1. Create a repository from this template

In the upper right of the [landing page for this repository](https://github.com/jhudsl/OTTR_Template/tree/1.8), click the green Use this template button and follow the steps to set up your course’s GitHub repository.

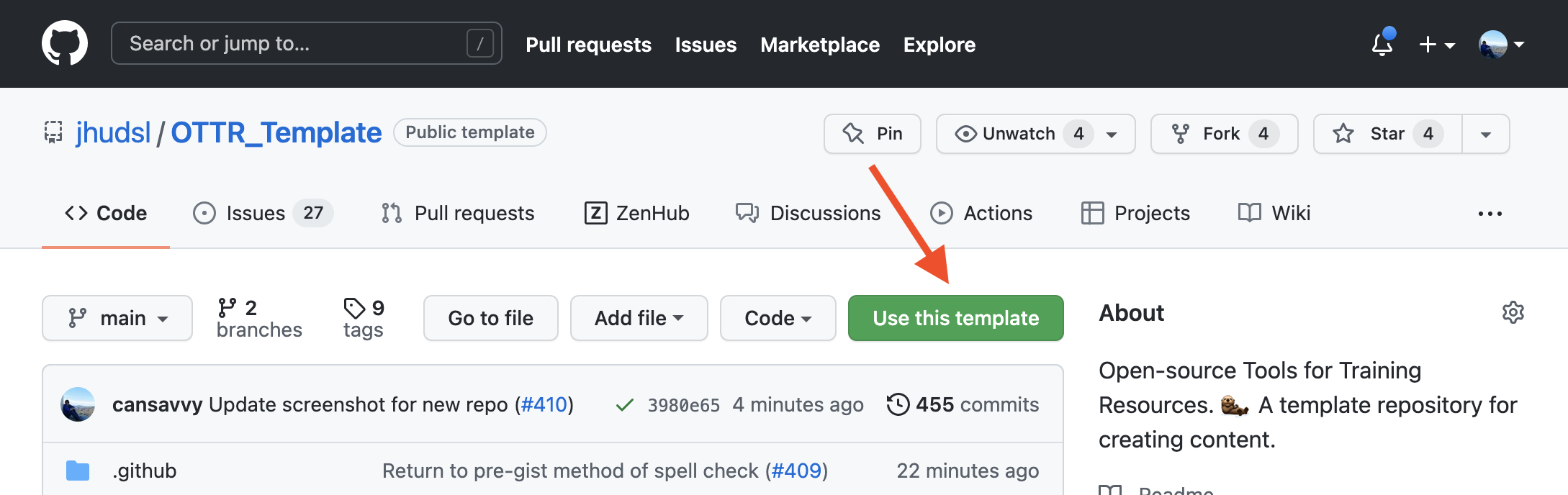


Figure : Where is the Use this template button?

### 1.0.2 2. Name your repository and fill in a short description.



Figure : Creating new course repository

### 1.0.3 3. Your course must be set to public for Github actions to work.

### 1.0.4 4. Set up a Git client

If you do not have a method of working with Git already it is recommended you use a git client to help you manage your branches more easily. Install [GitKraken](https://www.gitkraken.com/) for a handy way to manage your course locally.

### 1.0.5 5. Add jhudsl-robot as a collaborator

\*You can skip this step if your course is in the jhudsl organization.

The [Github actions](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots) in this repository need permissions to fully function. To this end, you need to add the jhudsl-robot as a collaborator on your repository with write permissions.

In your repository, go to your Settings > Collaborators & Teams and click on Add people. In the pop up window, search for and add jhudsl-robot.

Choose the write option then click Add jhudsl-robot to this repository.

### 1.0.6 6. Set up your Github Personal access token

The OTTR robots need permissions to run some of the actions.

To give them permission for all the actions, you need to set a GitHub secret we will call GH\_PAT. Go to Settings > Secrets and scroll down to Repository secrets. [Read more about GitHub secrets here](https://docs.github.com/en/actions/security-guides/encrypted-secrets) if you have general questions.

If you have organization admin privileges and plan on making multiple courses with OTTR, you can set GH\_PAT as an organization secret so you don’t have to do this again You only need to do this once as an organization (if this course is under jhudsl you don’t need to do this step).

* Click New repository secret/New organization secret button.
* Under name you must use GH\_PAT.
* For value: Create a personal access token [following these instructions](https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token#creating-a-token). Underneath Select scopes, check both repo and workflow. Then copy the PAT and save as the value. Your Github actions should all be able to run smoothly now.

### 1.0.7 7. Set up GitHub pages

Go to Settings > Pages.

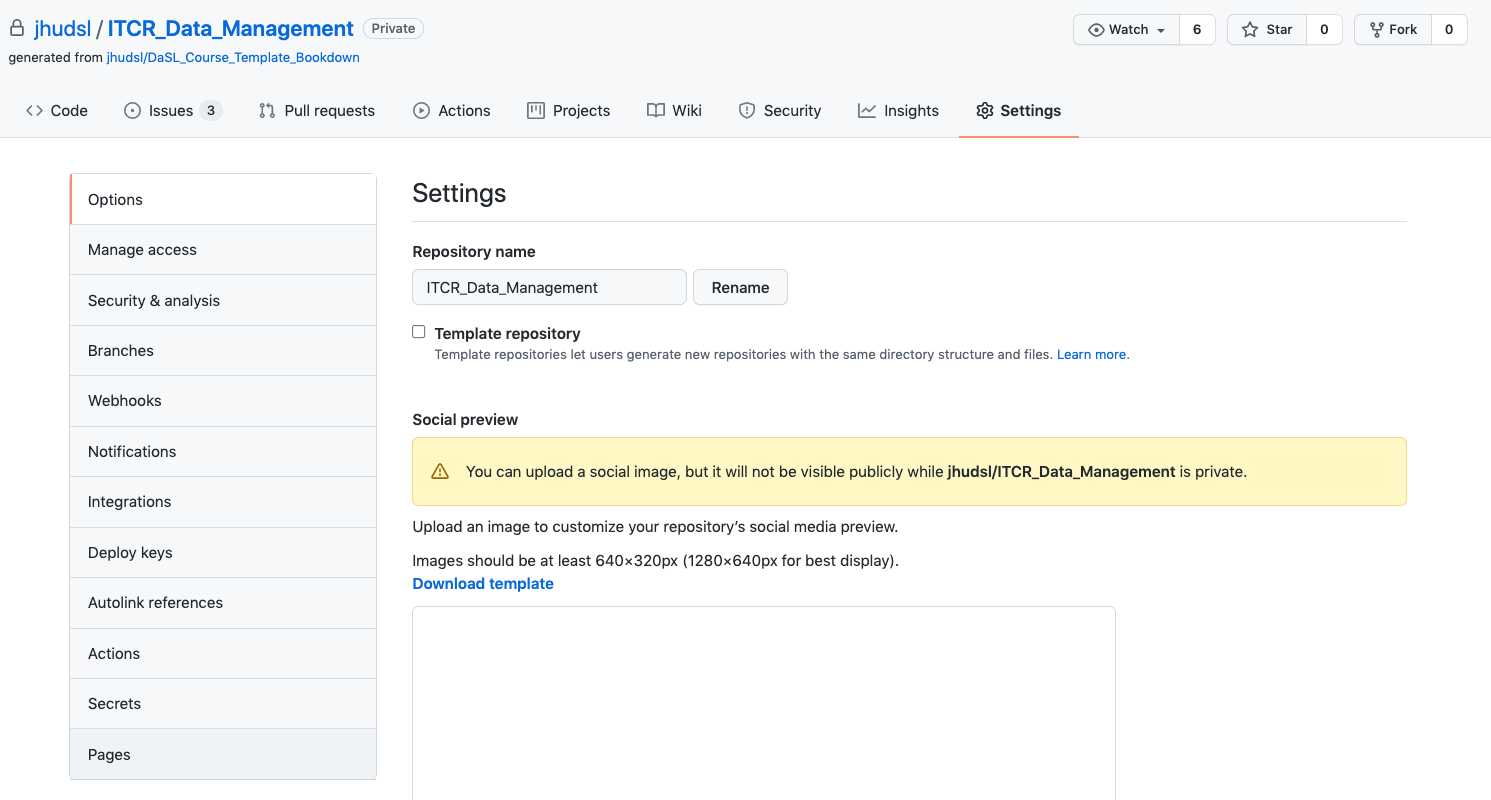


Figure : Find pages settings

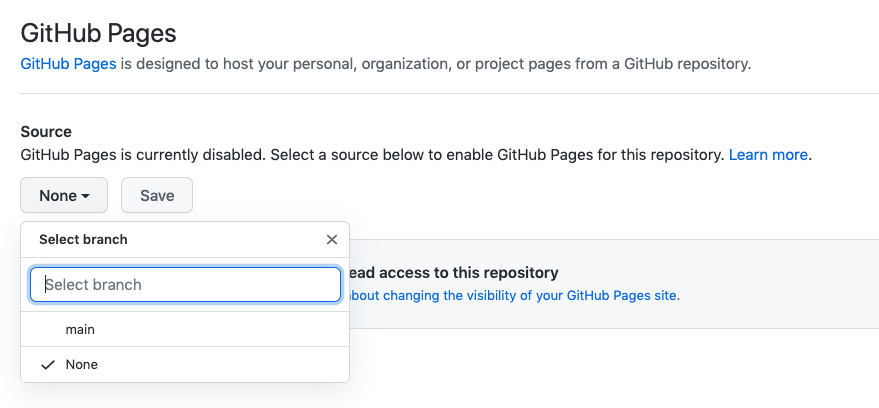


Figure : Change pages settings

* Under Source, pick the drop down menu and pick main and /docs.
* Then click Save.
* Lastly, check the box that says Enforce HTTPS at the bottom of this page.

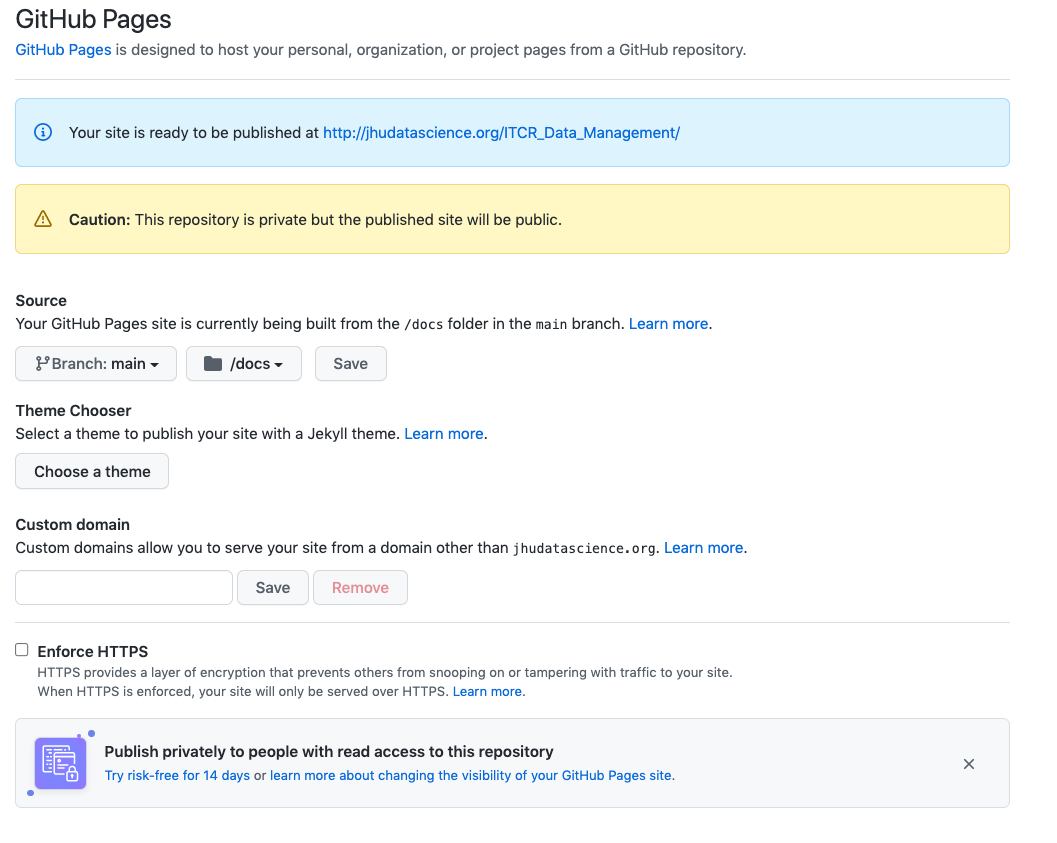


Figure : Enforce HTTPS

\* Note that if you go to your pages url right now (before you’ve pushed any file changes) you might see a 404 error because nothing has been triggered to be rendered. You can check your url pages after you file your first pull request (which we will guide you through [in the next section](https://github.com/jhudsl/OTTR_Template/wiki/Start-editing-your-course)).

### 1.0.8 8. Set up branch rules

\* Don’t set these settings right away! Wait a few minutes after starting your repository.

Go to Settings > Branches and click Add rule. This is only possible if you have GitHub Pro, Team, or Enterprise access. We recommend seeing if your organization can give you access if you do not already have access. To get this access through your organization, you will need to transfer ownership of your repo to your organization. See [here](https://docs.github.com/en/repositories/creating-and-managing-repositories/transferring-a-repository) about how to do this.

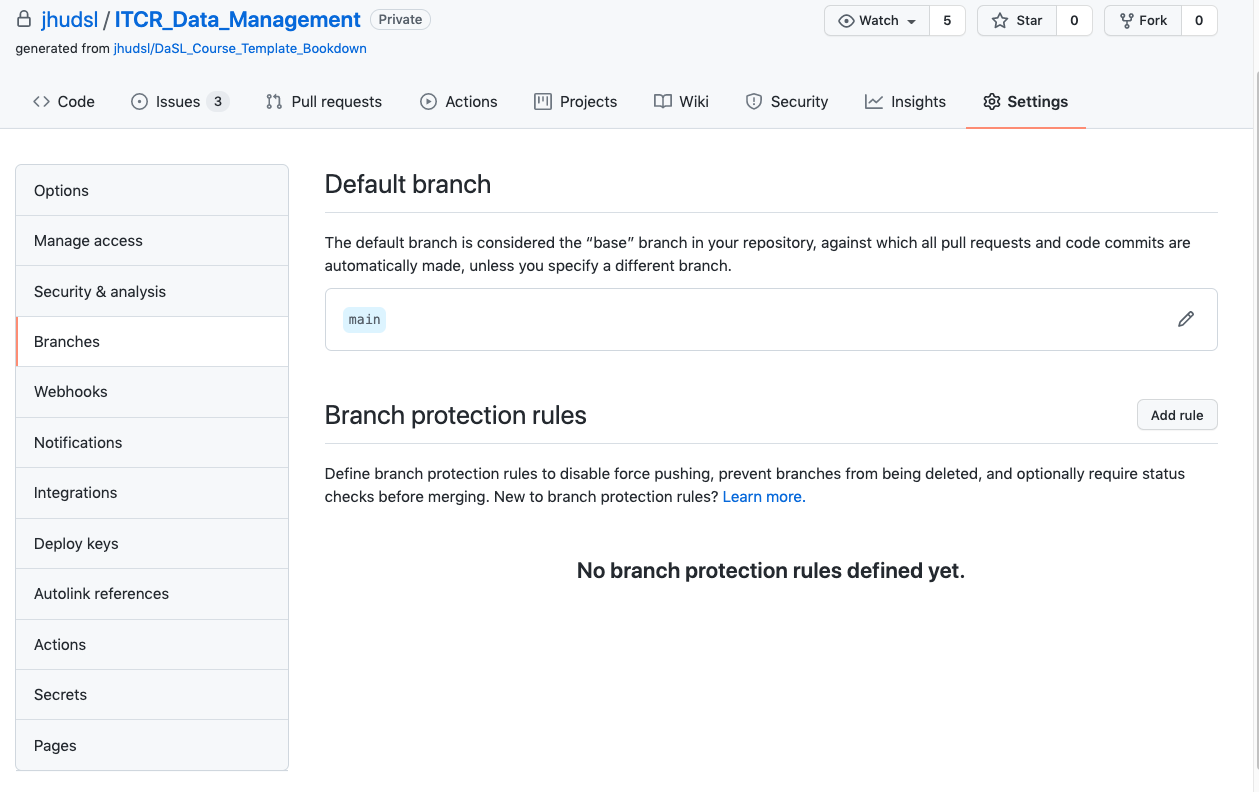


Figure : Add branch rule

For Branch name pattern, put main.

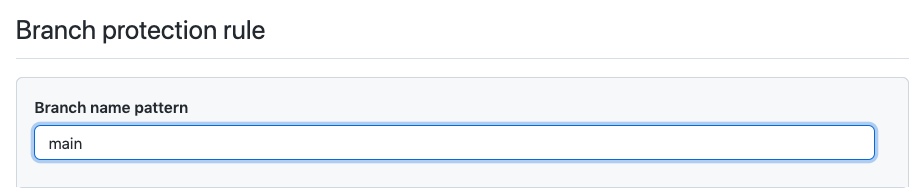


Figure : Put main for branch name pattern

*Protect the main branch*:  
Then check the box that says Require pull requests before merging.

*Make sure branches are updated*:  
- Check the box that says Require status checks to pass before merging. - Underneath this also check the box that says Require branches to be up to date before merging.

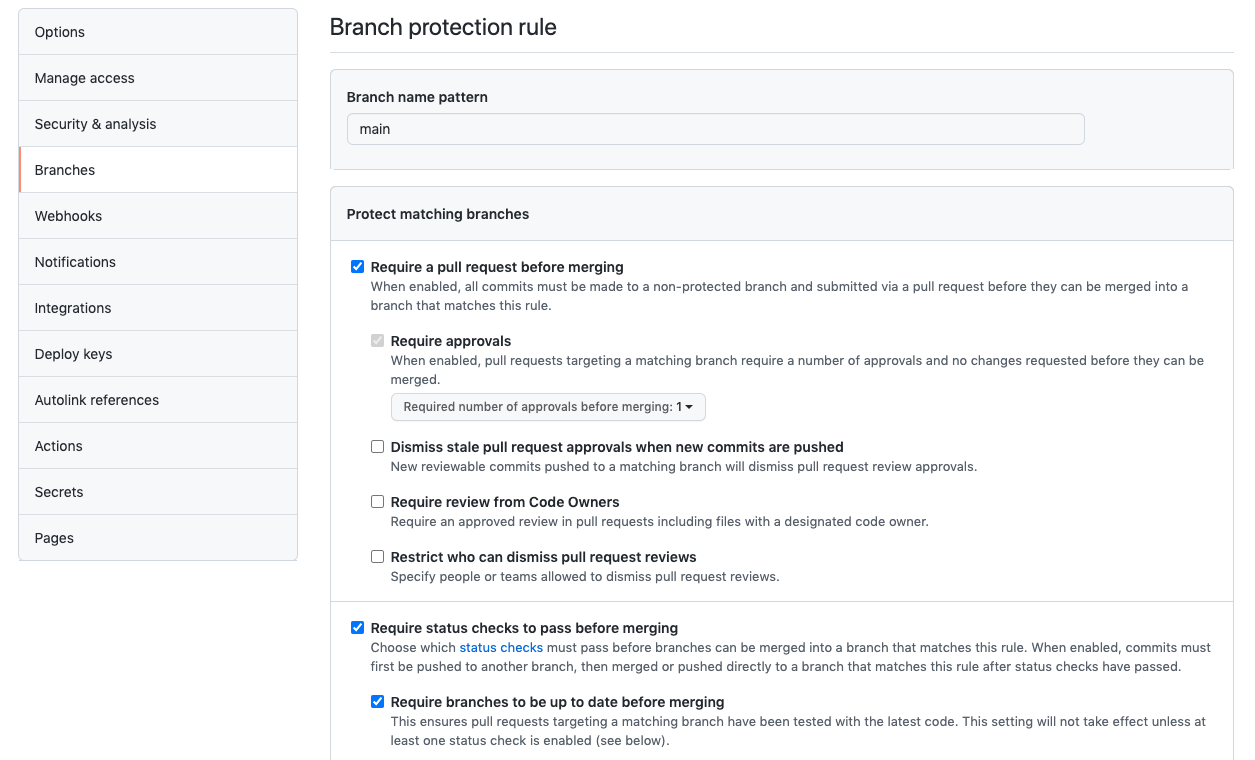


Figure : Branch settings

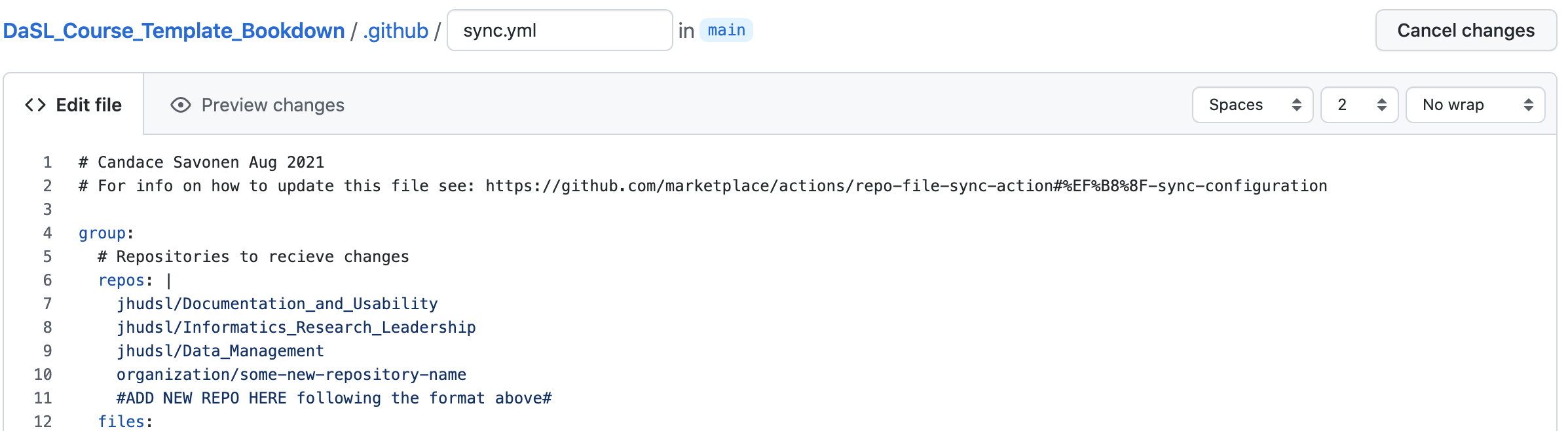
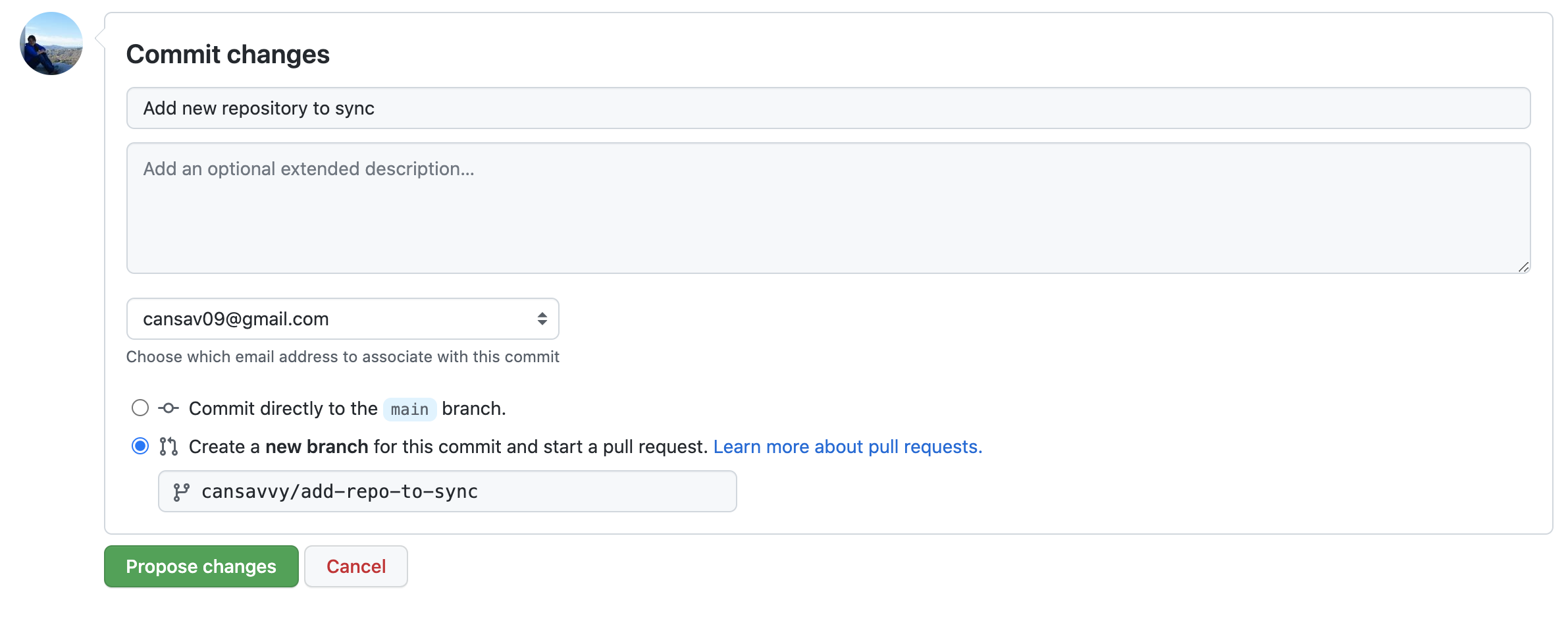
After setting up these new branch items, click Create and Save changes.

### 1.0.9 9. Enroll your repository for OTTR updates

This original course template: <https://github.com/jhudsl/OTTR_Template> is always a work in progress. We are working on adding more features and smoothing out bugs as we go (this is also why [your feedback is greatly appreciated](https://github.com/jhudsl/OTTR_Template/issues/new/choose)).

When updates are made to files that aren’t specific to the course content but instead run checks and other processes in the original repository, pull requests are filed automatically to any downstream repositories made from this template.

**To enroll in these automatic update pull requests, the new course’s repository name will need to be added to** [**this file in the original template**](https://github.com/jhudsl/OTTR_Template/edit/main/.github/sync.yml)

* Got to the respective file: .github/sync.yml of the OTTR template repository you created your repository from:
  + [This file for the main OTTR\_Template updates](https://github.com/jhudsl/OTTR_Template/blob/main/.github/sync.yml)
  + [This file for the OTTR\_Quizzes updates](https://github.com/jhudsl/OTTR_Quizzes/blob/main/.github/sync.yml)
* Add your repository’s name where it says #NEW REPO HERE#, being careful to indent the same amount as the other repositories listed. 
* Choose Create a new branch for this commit and start a pull request. Name the branch what you like. 
* Name the commit Add new repository to sync
* Click Propose changes.
* [See this guide](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request-from-a-fork) for any questions you have about creating a pull request from your forked repository.
* Write a short pull request description describing that you are adding a new repository to the sync.yml file.
* Click Create Pull Request
* Request @cansavvy as a reviewer.

If the your new course doesn’t need some of the functionality of these files or you find the automatic bothersome, you can feel free to use [this guide](https://github.com/marketplace/actions/repo-file-sync-action#sync-the-same-files-to-multiple-repositories) to tailor which files you want updates for. *If you have any questions about the implications of any of these updates or files, please tag @cansavvy.*

Your repository is all set up and you are ready to start editing your course and adding content! [Follow the next section’s instructions here](https://github.com/jhudsl/OTTR_Template/wiki/Start-editing-your-course)

# 2 Start editing your course

For any OTTR Questions we suggest joining the [OTTR Google Group](https://github.com/jhudsl/OTTR_Template/wiki/Getting-Help-(Google-Group)) and adding a discussion there about the problems you are encountering or questions you have.

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [File changes to be made](#file-changes-to-be-made)
  + [Files that need edited upon creating a new course.](#Xfa9b590d6981c7c23ffcc25a81de3afe224cbea)
  + [Files that need to be edited upon adding each new chapter (including upon creating a new course):](#X50ece7c8124005208f15347e686db2904f1b369)
  + [Files that need to be edited upon adding new packages that the book’s code uses:](#Xcce1edec24d14608b5c153ae4261a40bec232dd)
  + [Picking a style](#picking-a-style)
    - [Using a style set](#using-a-style-set)
* [Getting started with the Github workflow](#getting-started-with-the-github-workflow)
  + [git clone](#git-clone)
  + [Create a branch](#create-a-branch)
  + [Checks on the pull requests](#checks-on-the-pull-requests)
  + [More resources for learning GitHub](#more-resources-for-learning-github)

Now you have a course repository on GitHub and you’re ready to start editing files.

For any set of file changes, its recommended you file a pull request (which [the section below](#getting-started-with-the-github-workflow) will guide you on if you are unfamiliar). Filing a pull request initiates a series of checks done automatically by GitHub actions which will help you as you add content to your course (If you commit directly to the main branch, you will not reap the full benefit of these automatic checks).

These checks will do important things like: \* Check that the all the urls actually take learners somewhere (check-urls) \* Check that the code is styled using the styler package (style-and-sp-check) \* Check that the spelling is correct using the spelling package (style-and-sp-check) \* Create previews of the rendered versions of the course (render-preview) \* Check the formatting of any quizzes (check-quizzes)

You can adjust what checks are run by editing the config\_automation.yml file. This is further discussed in this section about [the GitHub Actions](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots).

## 2.1 File changes to be made

These are a list of changes you’ll want to make to get you started. This same list of changes is printed in an issue automatically on your repository for you to follow.

### 2.1.1 Files that need edited upon creating a new course.

* ☐ README.md - Fill in all the { } and delete the first part of the README that is referring to the OTTR\_Template.
* ☐ index.Rmd - title: should be updated.
* ☐ 01-intro.Rmd - replace the information there with information pertinent to this new course.
* ☐ 02-chapter\_of\_course.Rmd - This Rmd has examples of how to set things up, if you don’t need it as a reference, it can be deleted.

### 2.1.2 Files that need to be edited upon adding each new chapter (including upon creating a new course):

* ☐ \_bookdown.yml - The list of Rmd files that need to be rendered needs to be updated. See [instructions](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-with-Bookdown).
* ☐ book.bib - any citations need to be added. See [instructions](https://github.com/jhudsl/OTTR_Template/wiki/Citations).

### 2.1.3 Files that need to be edited upon adding new packages that the book’s code uses:

* If your course will require additional packages, docker/Dockerfile needs to have the new package added so it will be installed. See [instructions](https://github.com/jhudsl/OTTR_Template/wiki/Using-Docker#starting-a-new-docker-image) for more on this.

### 2.1.4 Picking a style

See more [about customizing style on this page in the guide](https://github.com/jhudsl/OTTR_Template/wiki/Change-Title-&-Customize-style). By default this course template will use the jhudsl data science lab style. However, you can customize and switch this to another style set.

#### 2.1.4.1 Using a style set

[Read more about the style sets here](https://github.com/jhudsl/OTTR_Template/wiki/Change-Title-and-style#customizing-the-style).

* ☐ On a new branch, copy the style-sets/<set-name>/index.Rmd and style-sets/<set-name>/\_output.yml to the top of the repository to overwrite the default index.Rmd and \_output.yml.
* ☐ Copy over all the files in the style-sets/<set-name>/copy-to-assets to the assets folder in the top of the repository.
* ☐ [Create a pull request](https://github.com/jhudsl/OTTR_Template/wiki/About-pull-request-review) with these changes, and double check the rendered preview to make sure that the style is what you are looking for.

## 2.2 Getting started with the Github workflow

If you do not have a method of working with Git already it is recommended you use a git client to help you manage your branches more easily. Install [GitKraken](https://www.gitkraken.com/) for a handy way to manage your course locally. These steps shown here will show you the GitKraken way of handling files.

### 2.2.1 git clone

It’s ideal to work on your course from your computer (you do not need to install any other software besides git if you don’t want to). To get going, you will need to clone the course repository it to your own computer. Cloning is just making a remote copy of the project local.

So to get started, you will need to clone your course’s repository you created that we will be using for the duration of this course.

To clone a GitHub repository, using GitKraken. First, click Clone a repo. Then, choose where you’d like the repository to be on your computer using the Browse button. You will need to Copy + Paste your new repository’s url to where it says URL.

Navigate to your repository on GitHub to copy the URL. Copying and pasting is advisable because any little typo will inhibit cloning.

Now you are ready to click Clone the repository! It will ask you if you’d like to Open Now, click that.

### 2.2.2 Create a branch

Handling branches is where you unleash the real benefit of GitHub, but it’s also the confusing part to get a hang of.

The best way to get a grasp on what the branches represent is to create one and start using it.

In GitKraken we can create a new branch; this will be your working copy. First, click the Branch button. Next, type in a branch name in the box that the cursor is blinking in. In our example, we are calling it a-new-branch. Now click Enter! Now you have a new branch!

Now we can edit our files and code however we normally would. Go ahead and make an edit to [some of the files listed in the above section](#file-changes-to-be-made).

If you’ve made a change to any file in your repository, it will appear in GitKraken and you can click on it to see the differences.

If we want to add these file changes to our current branch, we need to commit them.

Now that we have changes committed to our branch we are ready to also add them to the remote, internet copy! To do this, we will need to push our branch.

To push means to add changes that are on your new branch to the remote branch (internet version). The word origin just refers to where your branch is stored on the internet. Choose your origin in the dropdown menu and click Submit.

After a variable number of commits, your branch, called a-new-branch is a different version of the original code base that may have a nifty improvement to it. But our main goal is to add that nifty improvement to the main branch. To start this process of bringing in new changes to the main curated repository, we will create a pull request.

From GitHub:

Pull requests let you tell others about changes you’ve pushed to a GitHub repository. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.

Pull Requests are the meat of how code changes and improvements get reviewed and incorporated! A vast majority of the benefits of incorporating GitHub into your workflow centers around fully utilizing the power of pull requests!

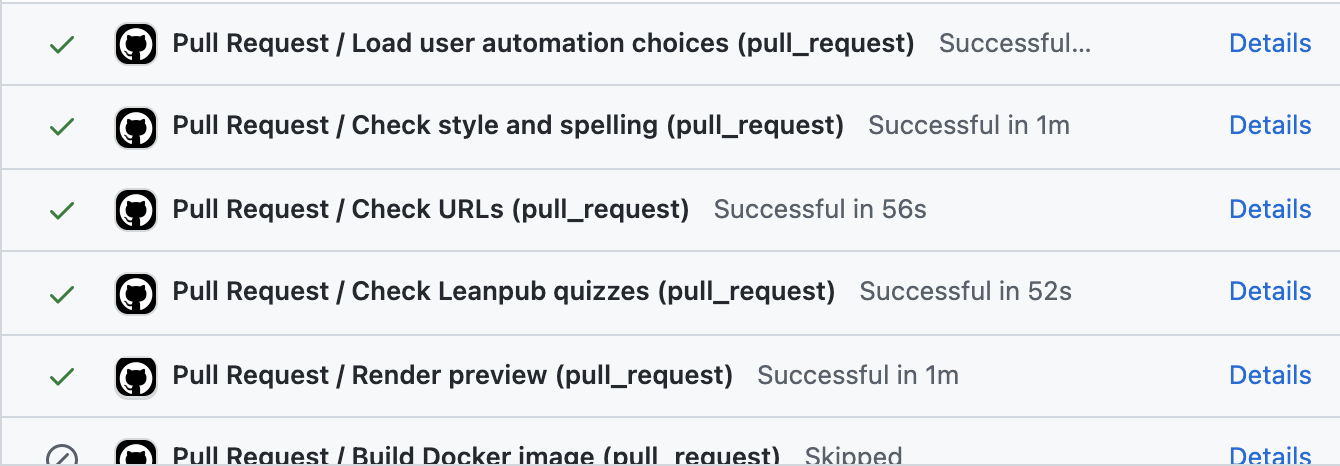
Now we can open up a pull request if we go to our GitHub repository on GitHub.

After you click on Compare & pull request you’ll be taken to a screen where you can add information about your changes. After you are done writing your description, click Create Pull Request! (If you don’t have your pull request description *perfect* don’t worry about it, you can always edit it later).

Congrats! You’ve just opened a pull request!

### 2.2.3 Checks on the pull requests

With your pull request open, the series of OTTR GitHub actions checks will commence! They will print out reports in comments on your pull request.



Read those comments to begin addressing the problems with more commits to your branch.

You can adjust what checks are run by editing the config\_automation.yml file. This is further discussed in this section about [the GitHub Actions](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots).

If you need more information on failed GitHub actions you can scroll to the bottom of your pull request where the status of the checks are shown and click on Details for more information. If you are unsure what the error message means and have trouble addressing it, please [file an issue on the OTTR\_Template repository to get help](https://github.com/jhudsl/OTTR_Template/issues/new?assignees=cansavvy&labels=bug&template=course-template-problem-report.md).

To summarize, this is what this workflow looks like:

For more on [what to put in a pull request’s description you can read this chapter](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/engaging-in-code-review---as-an-author.html)

For more on [how to review a pull request, see this chapter](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/engaging-in-code-review---as-a-reviewer.html)

### 2.2.4 More resources for learning GitHub

* [Using version control with GitHub](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/using-version-control-with-github.html)
* [Happy Git and GitHub for the useR](https://happygitwithr.com/)
* [GitHub for data scientists](https://towardsdatascience.com/introduction-to-github-for-data-scientists-2cf8b9b25fba)
* [Intro to GitHub](https://lab.github.com/githubtraining/introduction-to-github)
* [First Day on GitHub](https://lab.github.com/githubtraining/first-day-on-github)
* [First Week on GitHub](https://lab.github.com/githubtraining/first-week-on-github)
* [GitHub docs about creating a Pull Request](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request)
* [Making a Pull Request](https://www.atlassian.com/git/tutorials/making-a-pull-request)

# 3 Content creation tips

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Consider your motivation](#consider-your-motivation)
* [Consider your audience](#consider-your-audience)
* [Learning objectives](#learning-objectives)
* [Consider course and chapter length](#consider-course-and-chapter-length)
* [Graphics to emphasize and summarise](#graphics-to-emphasize-and-summarise)
* [Create quizzes with learning objectives in mind](#X779d26d39a8791282dcbceb240d56c57b15eb43)
* [Make courses engaging](#make-courses-engaging)
* [Get feedback](#get-feedback)
* [Example courses](#example-courses)

Creating course content is not always as straightforward as it may seem. Here we provide you with some tips about how to stay focused and write content that is useful for your intended audience.

### 3.0.1 Consider your motivation

First think about why people need your course. What gap in knowledge or expertise will your course help to fill? Think about what is uniquely useful about your course and write this down, as this will help guide you with the additional content creation steps.

### 3.0.2 Consider your audience

Think about who needs your course. Is it relatively homogenous group? Or do you have individuals of different backgrounds or expertise levels that might take your course. If this is the case, make sure that you create learning objectives and content that would cover the interests of both groups. For example, if many beginners might be taking your course be sure to be careful to define jargon and provide basic fundamental information. However, if there are are also more advanced learners that might take your course, consider making some content that will help this group of learners stretch beyond what they already know.

### 3.0.3 Learning objectives

Learning objectives help you to stay focused as you write you course content. These are actionable statements that describe what your students should be able to do after completing a course or chapter. Start each with a verb that is as specific as possible. Consider your audience as you create your learning objectives. Are they appropriate? Are they useful?

This [website](https://eclearn.emmanuel.edu/courses/1285497/pages/how-to-write-measurable-learning-objectives/) has some great descriptions about how to best write learning objectives, as well as examples of verbs that you can use to start objectives.

They point out that there are many levels of learning - with the highest form being the ability to create something new based on what you learn:

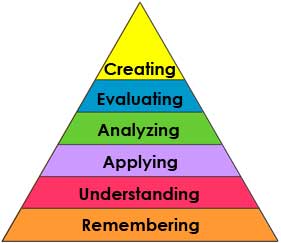


Figure : Levels of learning

[[source](https://wlclassroom.com/2013/03/12/bloomstaxonomy-2/)]

Although online content often makes higher levels of learning more difficult, it is not necessarily impossible.

For example, take a look at this section of a course about creating documentation: <https://jhudatascience.org/Documentation_and_Usability/creating-a-smooth-getting-started-section.html>

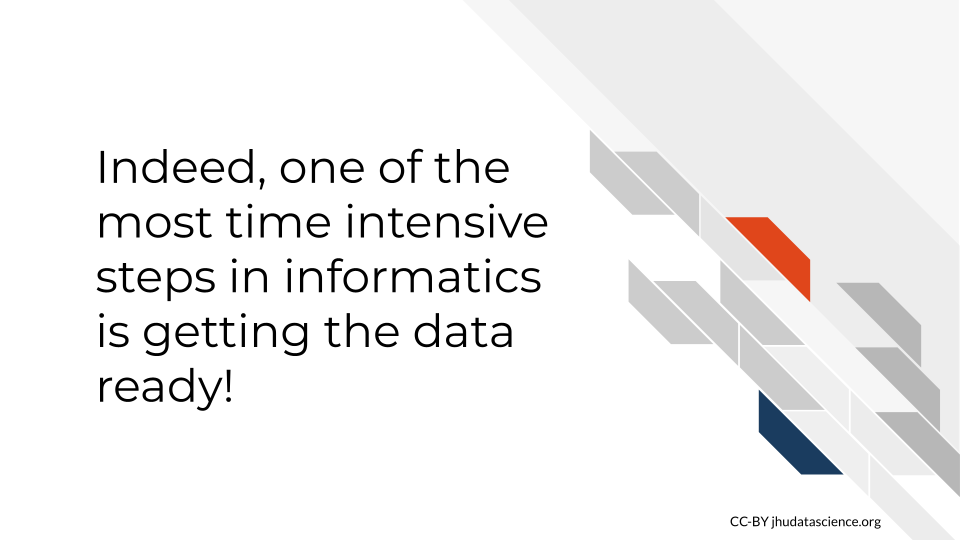
This provides learners with the background knowledge to actually create their own “getting started” documentation material.

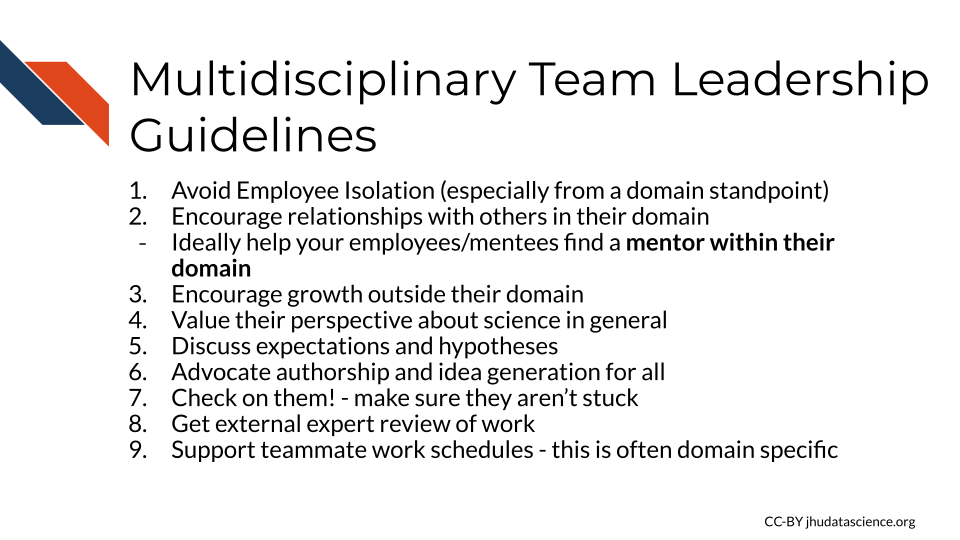
### 3.0.4 Consider course and chapter length

Often learners prefer smaller chunks of information at a time. Splitting longer chapters into shorter chapters or longer courses into smaller courses can help your learners find the information that they specifically need and keep them from feeling overwhelmed.

### 3.0.5 Graphics to emphasize and summarise

Graphics can be really helpful to emphasize specific information or to summarise the take-home messages of a particular section. We suggest that if possible make such summarising graphics for each section of your course.

Here is an example of a graphic to emphasize a particular piece of information: 

Here is an example of a graphic to summarise a section: 

### 3.0.6 Create quizzes with learning objectives in mind

Quizzes are also a great step for assessing if your course is really accomplishing your instruction goals. As you create your quizzes, take a moment to think about the major take-home messages of your course. Use this as an opportunity to emphasize what you want students to know, especially for concepts that are often misunderstood. Use your quiz creation time to reflect on if your material really captures what you want to teach and emphasize.

Here is a great example of a quiz question that requires some higher level learning:

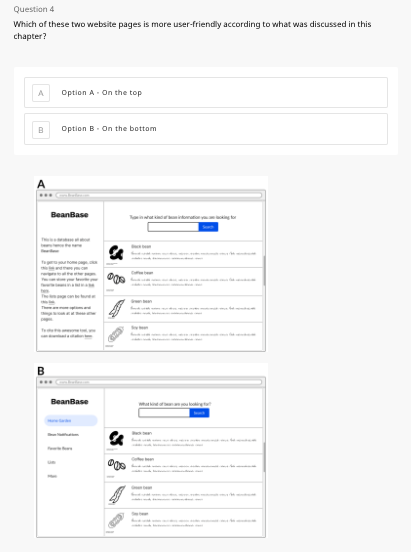


Figure : Quiz question example

Students are taught general information about good documentation. This now requires the students to apply this knowledge in a new context to recognize which option is better.

Note that this question uses images. This is a feature that is coming soon for the template.

### 3.0.7 Make courses engaging

Consider adding cartoons or other images/graphics that can make your course more relatable and entertaining. This can be really important for keeping learners interested and motivated. Using concrete examples that are relatable can also be very helpful.

### 3.0.8 Get feedback

If possible find people in your target audience demographic to give you feedback. Especially ask them to try the quizzes to see if your content clearly describes what they need to know to be able to successfully pass the quizzes. This is also a great time to catch any typos! Also consider getting feedback from experts about topics that you are less familiar with. We hope that using our template will make it easier to incorporate feedback from others over time to allow courses to be living documents that are simple to adapt. Make sure that you give credit to people that give you feedback! Check out our [credits section](https://github.com/jhudsl/OTTR_Template/wiki/How-to-give-credits) for more information on how to do this.

### 3.0.9 Example courses

Beyond our template, check out these courses we have made using this template for additional examples of how we approached content creation: 1) [Documentation and Usability](https://github.com/jhudsl/Documentation_and_Usability/) 2) [Leadership for Cancer Informatics Research](https://github.com/jhudsl/Informatics_Research_Leadership/) 3) [Reproducibility in Cancer Informatics](https://github.com/jhudsl/Reproducibility_in_Cancer_Informatics) 4) [Advanced Reproducibility in Cancer Informatics](https://github.com/jhudsl/Adv_Reproducibility_in_Cancer_Informatics)

# 4 Setting up images and graphics

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Themes for non-ITCR projects (JHU users):](#themes-for-non-itcr-projects-jhu-users)
* [Themes for ITCR project (nonJHU users and JHU users):](#X58f0d83838df71febc34053765939139d0de12a)
* [Accessibility](#accessibility)
* [Adding images and graphics in text](#adding-images-and-graphics-in-text)
* [Adding videos in text](#adding-videos-in-text)
* [Adding embedded files to text](#adding-embedded-files-to-text)
* [Learning Objectives Formatting](#learning-objectives-formatting)

To maintain style and attributions for graphics and images, as well as to enable easy updates in the future, please start a new Google Slide document for your course. This also allows you to make videos of your slides that can be added to your course.

Each Rmd with images that is a part of your Bookdown needs to have this chunk at the beginning so that images are stored properly for Leanpub conversion.

If you are unfamiliar with [how R Markdown code chunks work, read this](https://rmarkdown.rstudio.com/lesson-3.html).

```{r, include=FALSE}  
ottrpal::set\_knitr\_image\_path()

Next, import the appropriate theme (see [this video](https://youtu.be/pNbwF263yY8) for assistance):

Note that you will need to [change your Google Slide permissions](https://support.google.com/drive/answer/2494893?hl=en&co=GENIE.Platform%3DDesktop) to “Anyone with a link” in order for some functionality to work.

### 4.0.1 Themes for non-ITCR projects (JHU users):

* If you are not creating a course as part of a larger collaboration but simply just as a DaSL course, import the theme from this [template](https://docs.google.com/presentation/d/1YmwKdIy9BeQ3EShgZhvtb3MgR8P6iDX4DfFD65W_gdQ/edit?usp=sharing). Once you have done this add a slide with the “Title slide” layout under layouts - and add your course title.



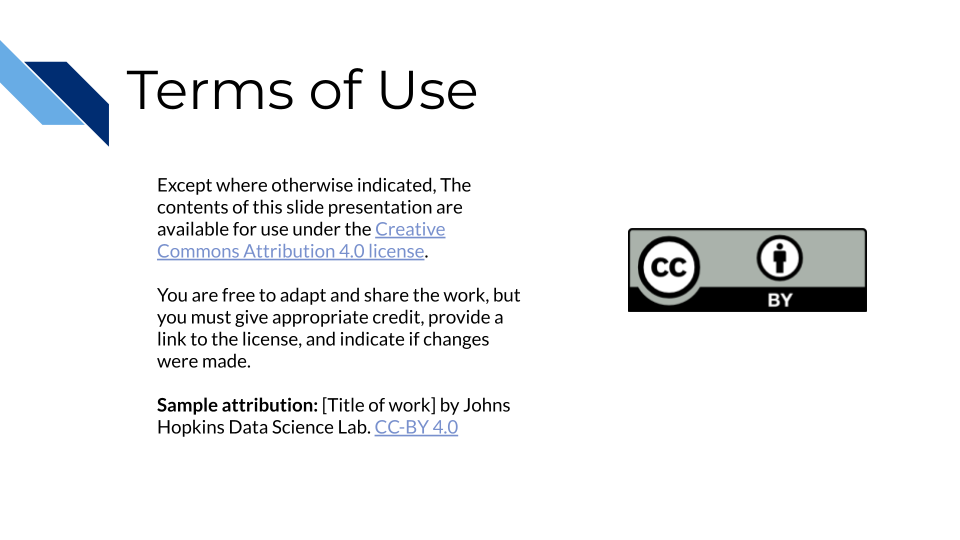
* If you are working on a project (more likely) with the [Johns Hopkins Data Science Lab](http://jhudatascience.org/) than import the theme from this [template](https://docs.google.com/presentation/d/1qa3v_zlNedCKHQ6IYF5MM1ZLZLgHUs0U4usgDRRe3W8/edit?usp=sharing).

Add a title slide and update it with your project’s logo.



In either case:

Be sure to add a terms of use slide to let people know about our licensing.



For any major point, please select this layout:



Figure : Major\_point\_layout

For itemized lists, please select this layout:

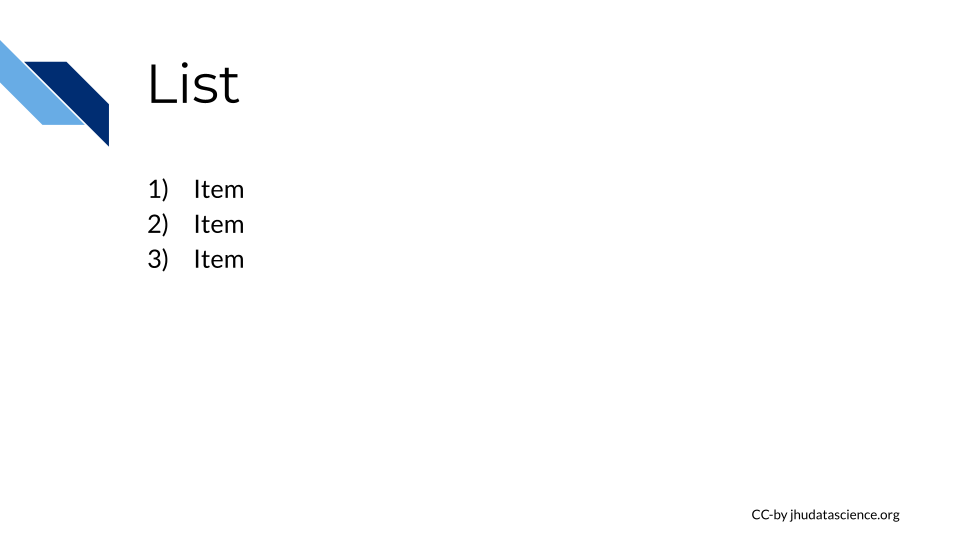


Figure : list\_layout

Please select the layout that looks like this for any images/content from outside sources:



Figure : source\_layout

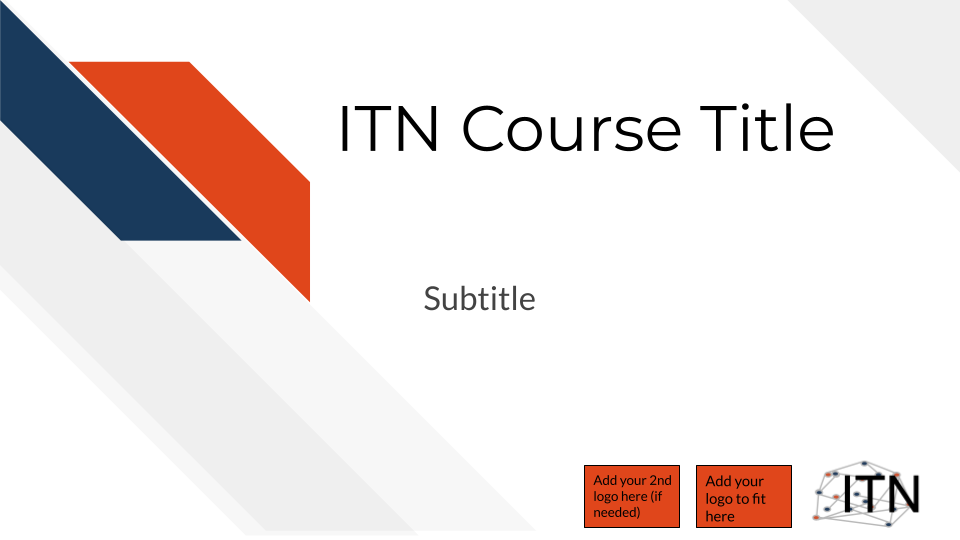
Please update the text at the bottom to describe the source. If it is not from JHU, remove the JHU information.

Also please type text describing the image in the notes section of the slide, this can be used to give visually impaired learners audio explanations of the images.

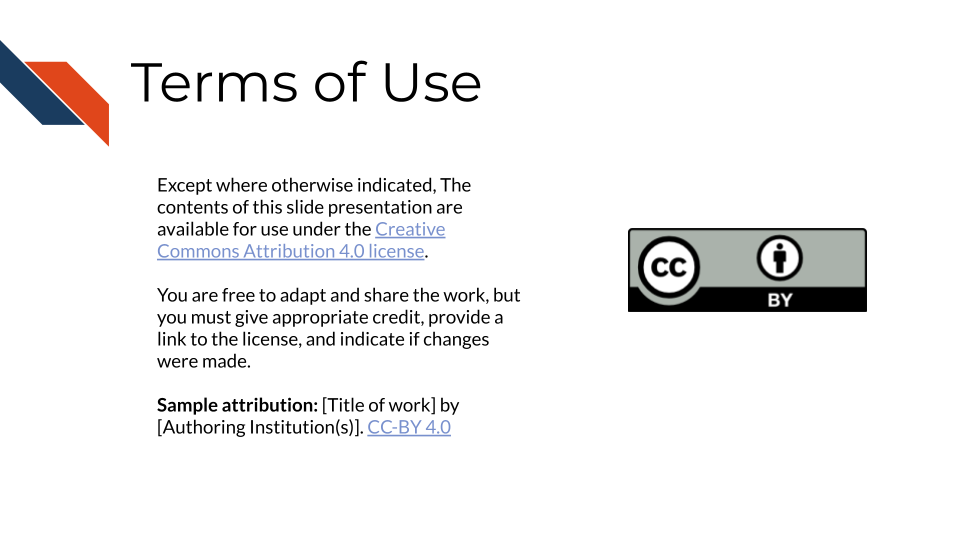
Once complete, you can incorporate slides into your course using the instructions in the [Adding images and graphics in text](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-images-and-graphics#adding-images-and-graphics-in-text) below.

### 4.0.2 Themes for ITCR project (nonJHU users and JHU users):

* If you are not part of the [Johns Hopkins Data Science Lab](http://jhudatascience.org/), import the theme from this [template](https://docs.google.com/presentation/d/1lMNTlY8S21cNG4hL6NoOmXYZBQ30BtCQ4UdUJ7J_Wvs/edit?usp=sharing). Once you have done this add a slide with the ITN2 layout - and add your course title and any logos for your organization(s) that may be appropriate like so:



Also be sure to add a terms of use slide to let people know what licensing that you have chosen. You can use ours (that looks like the slide below) if you like or create your own with a different license.



* If you are part of the [Johns Hopkins Data Science Lab](http://jhudatascience.org/) than import the theme from this [template](https://docs.google.com/presentation/d/1-7UvgVq5tP1pasTEErUM3bJFH2fU_pilH6i6_81CCXU/edit?usp=sharing).

Be sure to add a Johns Hopkins terms of use slide.

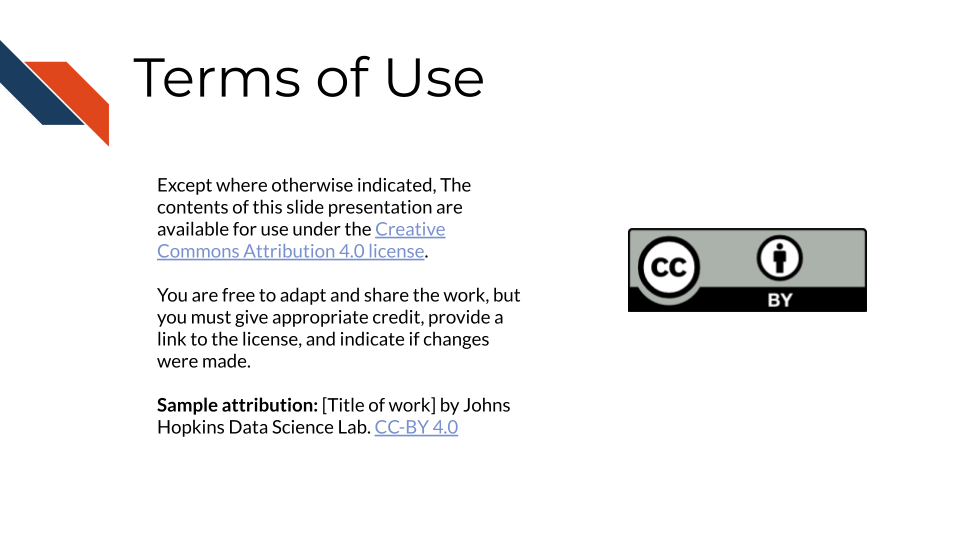


Figure : Terms of Use

For any major point, please select this layout:



Figure : Major\_point\_layout

For itemized lists, please select this layout:

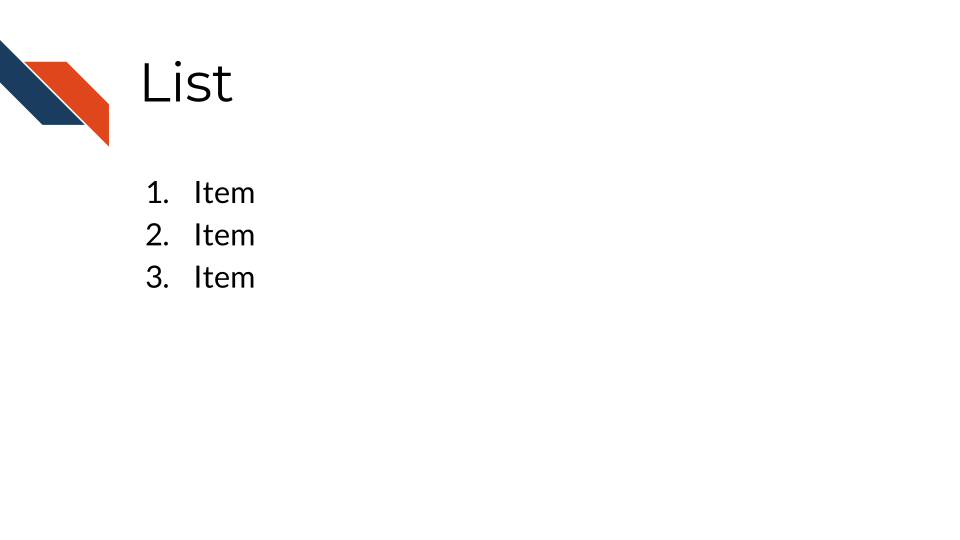


Figure : list\_layout

Please select the layout that looks like this for any images/content from outside sources:



Figure : source\_layout

Please update the text at the bottom to describe the source. If it is not from JHU, remove the JHU information.

Also please type text describing the image in the notes section of the slide, this can be used to give visually impaired learners audio explanations of the images.

Once complete, you can incorporate slides into your course using the instructions in the [Adding images and graphics in text](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-images-and-graphics#adding-images-and-graphics-in-text) below.

### 4.0.3 Accessibility

Each slide and image added to the courses needs to be accessible. There are two things to check for each slide:

* ☐ Each slide is described in the notes of the slide so learners relying on a screen reader can access the content. See <https://lastcallmedia.com/blog/accessible-comics> for more guidance on this.
* ☐ The color palette choices of the slide are contrasted in a way that is friendly to those with color vision deficiencies. You can check this using [Color Oracle](https://colororacle.org/).

### 4.0.4 Adding images and graphics in text

All images should be included in your Google Slides with the captions we discussed above. To add images in the text in your Rmd, use the following function within an [R code chunk](https://bookdown.org/yihui/rmarkdown/r-code.html).

```{r, fig.alt="Alternative text", echo = FALSE, outwidth = "100%"}  
ottrpal::include\_slide(<google\_slide\_url>)  
```

You can obtain the by viewing the slide with the image you want to add and copying the address from the browser search bar.

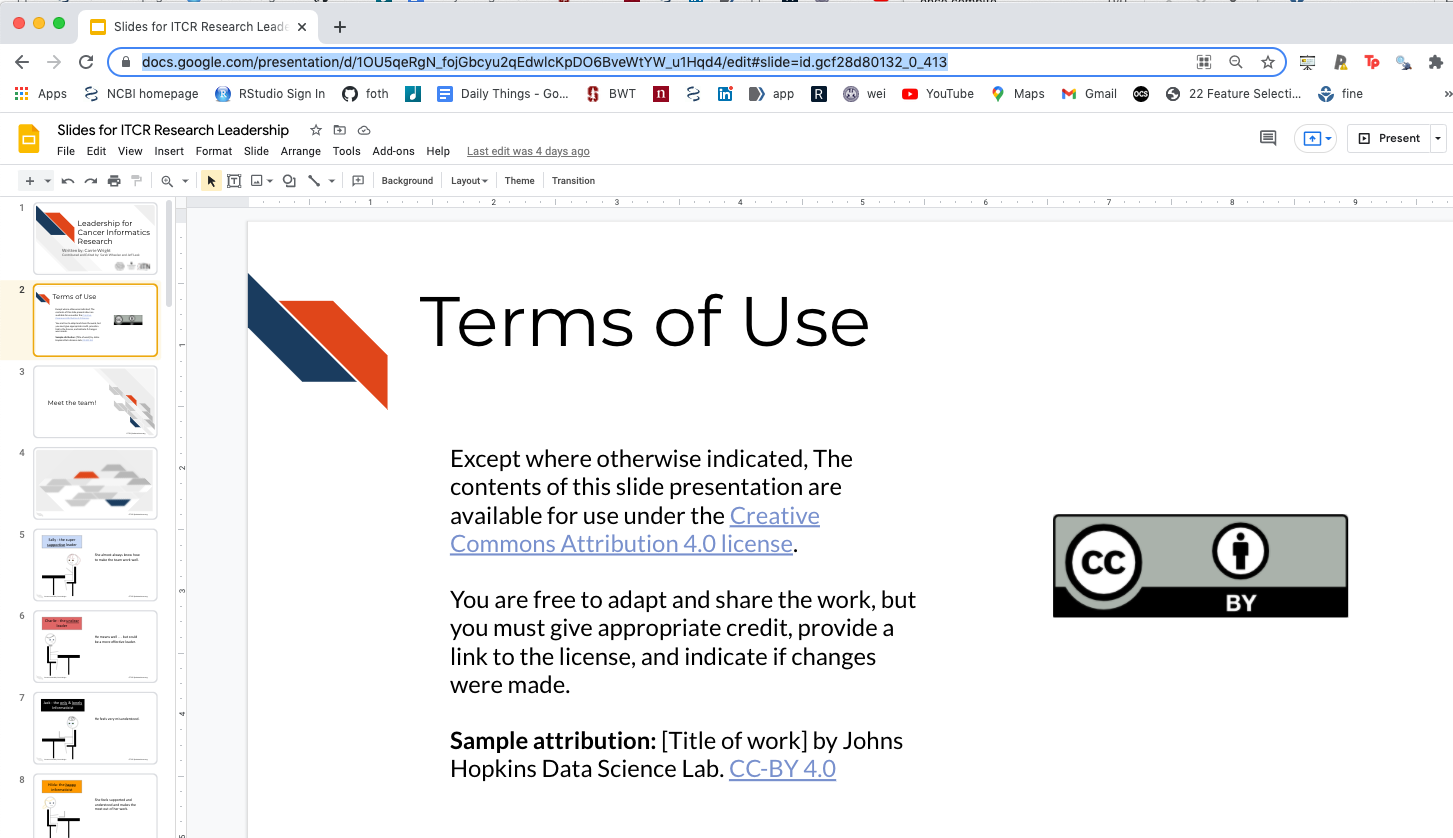


Figure : Google slide url

Also add notes (the same as the fig.alt text) to each slide in the google slide presentation describing the text or images of the slide to allow for the content to be accessible to vision impaired individuals, as this can be converted to audio. Note that you can’t have any line breaks within the fig.alt text. If there are no line breaks the text should appear blue within the code chunk.

The echo=FALSE ensures that the r code is hidden from your course, while the out.width = "100%" is used to size the image. We generally recommend going with larger images.

*You must define fig.alt in the code chunk options/parameters to pass to knitr.* You can adjust the size(fig.hight, fig.width, out.width, out.height), alignment (fig.align), or caption (fig.cap) of the image you can use these arguments in the code chunk tag:

```{r, fig.alt="Alternative text", fig.height=4, fig.align='center', fig.cap='...'}

Google Slides must be **public**. Share settings must be set to “Anyone on the internet with this link can view”. Note that “Private” is the default setting when you make a new presentation.

See [Chapter 2](https://github.com/jhudsl/OTTR_Template/blob/main/02-chapter_of_course.Rmd) of the template course for examples.

### 4.0.5 Adding videos in text

To add a youtube video to your Rmd files use the following:

```{r, fig.align="center", fig.alt = "video", echo=FALSE, out.width="100%"}  
knitr::include\_url("https://www.youtube.com/embed/yiZQaE0q9BY")

To get the appropriate youtube url do the following: 1) click on the **SHARE** button on the lower right corner of the video on youtube 2) click on the **Embed** option on the far left 3) copy just the part after "src =" and paste the url into the knitr::include\_url() function

Again, it is important to use the echo=FALSE option so that only the video is shown and not the code to generate it.

See [Chapter 2](https://github.com/jhudsl/OTTR_Template/blob/main/02-chapter_of_course.Rmd) of the template course for examples.

### 4.0.6 Adding embedded files to text

Sometimes it is useful to include an embedded version of a website or file on a website, if there is a particularly important link and you don’t want to rely on learners clicking the link.

To include such a file or website do the following:

```{r, fig.align="center", echo=FALSE}  
knitr::include\_url("https://www.messiah.edu/download/downloads/id/921/Microaggressions\_in\_the\_Classroom.pdf", height = "800px")

Again you will need to include echo = FALSE to ensure that the code to generate the preview of the website or file is not included in your course material.

If you want to include a file that is not hosted online, consider hosting it on GitHub using the method described for hosting your Bookdown version of the course. See the [Set up GitHub pages](https://github.com/jhudsl/OTTR_Template/wiki/Starting-a-new-course-with-this-template#set-up-github-pages) section.

Then you would do the following, where the url is that of your hosted file:

```{r, fig.align="center", echo=FALSE}  
knitr::include\_url("https://carriewright11.github.io/stringr\_RLadies/index.html", height = "800px")

See [Chapter 2](https://github.com/jhudsl/OTTR_Template/blob/main/02-chapter_of_course.Rmd) of the template course for examples.

### 4.0.7 Learning Objectives Formatting

Each chapter should start with Learning objectives! You can use [this website](https://eclearn.emmanuel.edu/courses/1285497/pages/how-to-write-measurable-learning-objectives/) to help you craft learning objectives.

Learning objectives should be stated both in the slides and in the beginning of each bookdown chapter. Because of this, you may find it most handy to use the [List layout](https://docs.google.com/presentation/d/1-7UvgVq5tP1pasTEErUM3bJFH2fU_pilH6i6_81CCXU/export/png?id=1-7UvgVq5tP1pasTEErUM3bJFH2fU_pilH6i6_81CCXU&pageid=gcf0c1d8548_0_141) slide for stating your Learning objectives and then embed that in the book from your GoogleSlides [the instructions here](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-images-and-graphics#adding-images-and-graphics-in-text).

# 5 Choosing Publishing platforms

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Choosing publishing platforms](#choosing-publishing-platforms)
* [Links to the guides for each platform:](#links-to-the-guides-for-each-platform)

## 5.1 Choosing publishing platforms

This template helps you prepare material to publish, but there are certain steps that cannot be automated. In those instances we’ve provided step-by-step guides to get you started on getting your material published on the platforms.

Depending on your goals and intended audience you may want to publish just one or all three of the platforms supported:

## 5.2 Links to the guides for each platform:

* [Publishing with Bookdown](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-with-Bookdown)
* [Publishing with Leanpub](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Leanpub)
* [Publishing with Coursera](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Coursera)

# 6 Publishing with Bookdown

Bookdown is published using Github pages which you can set up [following the set up repository instructions](https://github.com/jhudsl/OTTR_Template/wiki/Starting-a-new-course-with-this-template#set-up-github-pages).

As you modify the names of the chapters of your course and add more chapters (using the .Rmd files), you need to update the \_bookdown.yml file accordingly.

For example let’s say that we added another chapter and named the file 03-new\_chapter\_of\_course.Rmd. We would update our \_bookdown.yml to look like this:

book\_filename: "Course\_Name"   
chapter\_name: "Chapter "   
repo: https://github.com/jhudsl/OTTR\_Template/ ##Make sure you update this for your GitHub Repo!!   
rmd\_files: ["index.Rmd",   
 "01-intro.Rmd",   
 "02-chapter\_of\_course.Rmd",   
 "03-new\_chapter\_of\_course.Rmd", ##Only this is new!  
 "about.Rmd"]   
new\_session: yes   
delete\_merged\_file: true   
language:   
 ui:   
 chapter\_name: "Chapter "   
output\_dir: "docs"

Notice how only one line is different - the one that says 03-chapter\_of\_course.Rmd", Be careful about quotation marks: "" as well as commas: ,!

Once we do this we can preview the book!

You can do so by typing: bookdown::serve\_book() in the RStudio Console.

Note that when you run bookdown it will create an .rds file; you can generally ignore this file.

You will then see a live version of your book in your RStudio viewer.

Note that When a pull request is merged to main, bookdown::render\_book() will be re-run by the [GitHub actions](#github-actions) and the results added to main.

# 7 Publishing with Coursera

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Setting up your Coursera course](#setting-up-your-coursera-course)
* [Converting your files for upload to Coursera](#X80d0781f07bfd49776b3905193d2af26e2ddde1)
* [Navigating to your course on Coursera](#navigating-to-your-course-on-coursera)
  + [Adding new chapters to Coursera](#adding-new-chapters-to-coursera)
  + [Adding new quizzes to Coursera](#adding-new-quizzes-to-coursera)
* [Add programmed messages](#add-programmed-messages)
  + [Welcome template message:](#welcome-template-message)
  + [Completion template message:](#completion-template-message)
* [Grading formula](#grading-formula)
* [Content Schedule](#content-schedule)
* [Module Descriptions](#module-descriptions)
* [Landing Page](#landing-page)

Once your content has been largely developed you may want to add your course to Coursera.

This guide was prepared specifically for those developing courses for the [ITN project](https://www.itcrtraining.org/) at Johns Hopkins University or other Coursera partner institutions. It reflects the needs and goals of ITN course developers. Coursera’s Educator Resource Center is the authoritative source of documentation for using the Coursera platform.

## 7.1 Setting up your Coursera course

*If you are from Johns Hopkins:* You will need to [follow this document](https://docs.google.com/document/d/1aZeOSFLkK4hZne4Vb1iaP_0H4zyhIwvbnw9sbdCFq1Y/edit?usp=sharing) and send information Ira Gooding who will create a course shell for you.

*If you are not from Johns Hopkins:* You will need to set up your own educator profile and course shell through your institution.

## 7.2 Converting your files for upload to Coursera

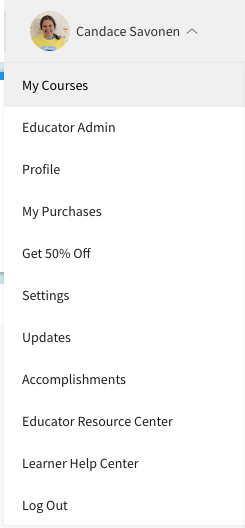
The Github actions set up in the render-all.yml render your course material in a format suitable for linking to Coursera. This Coursera version is identical except that the table of contents bar on the left side has been dropped so as to not confuse Coursera users about navigating the course. These files can be previewed in the docs/no\_toc folder and a link to the Coursera version is printed out in a GitHub comment in your pull request by render-preview.yml.

If you have written your quizzes in Leanpub format first, render-all.yml github action will convert those quizzes so they are ready for upload to Coursera.

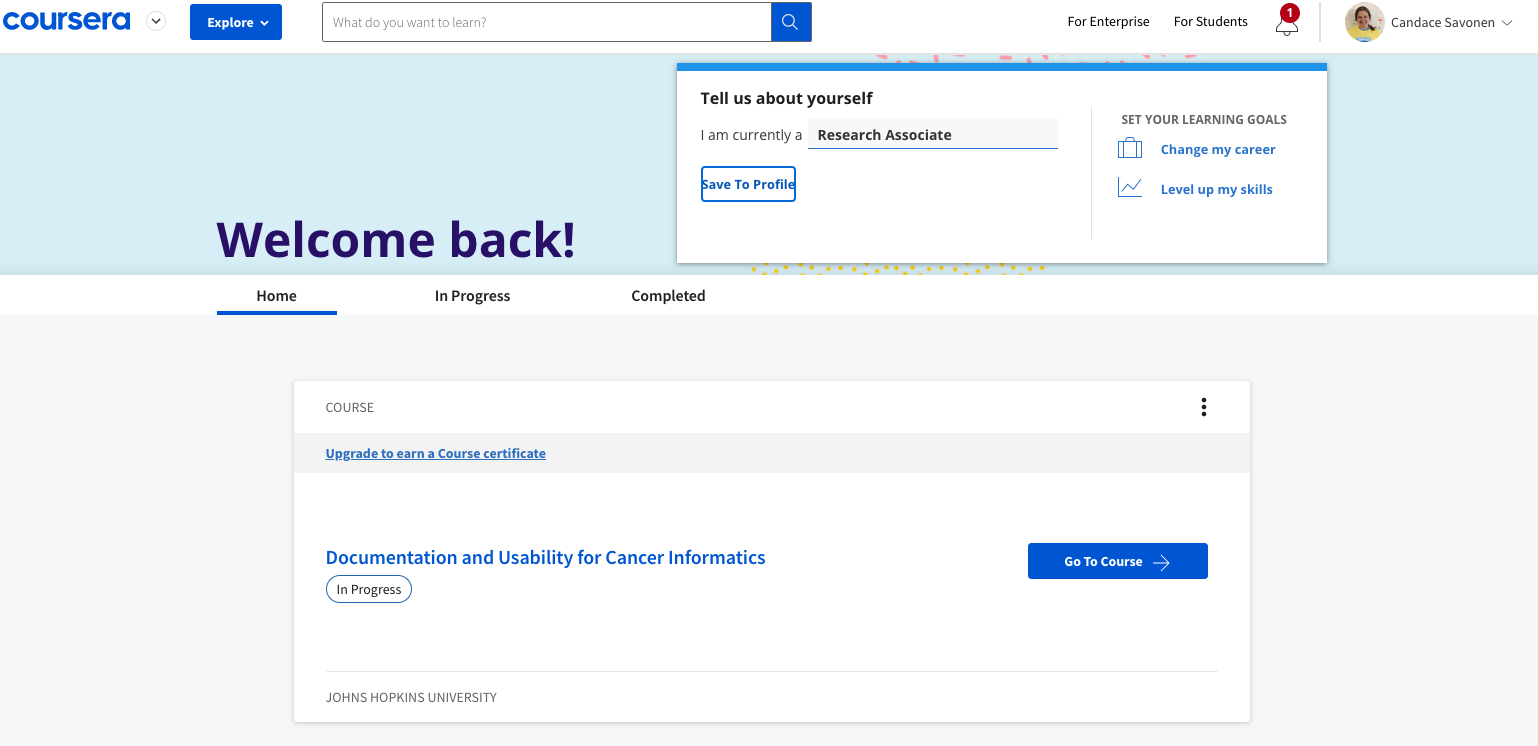
Or alternatively if you do not wish to have a Leanpub version of your course, you can write your quizzes directly on Coursera’s website. You can download the quiz you write on their website as a yaml and store it on your repository as well.

## 7.3 Navigating to your course on Coursera

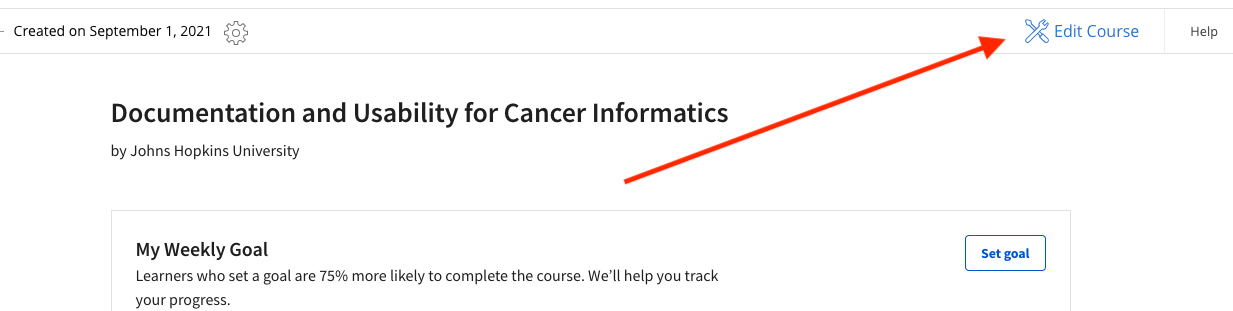
After your [course shell has been added to your account](#setting-up-your-coursera-course) you can navigate to it by going to My Courses:



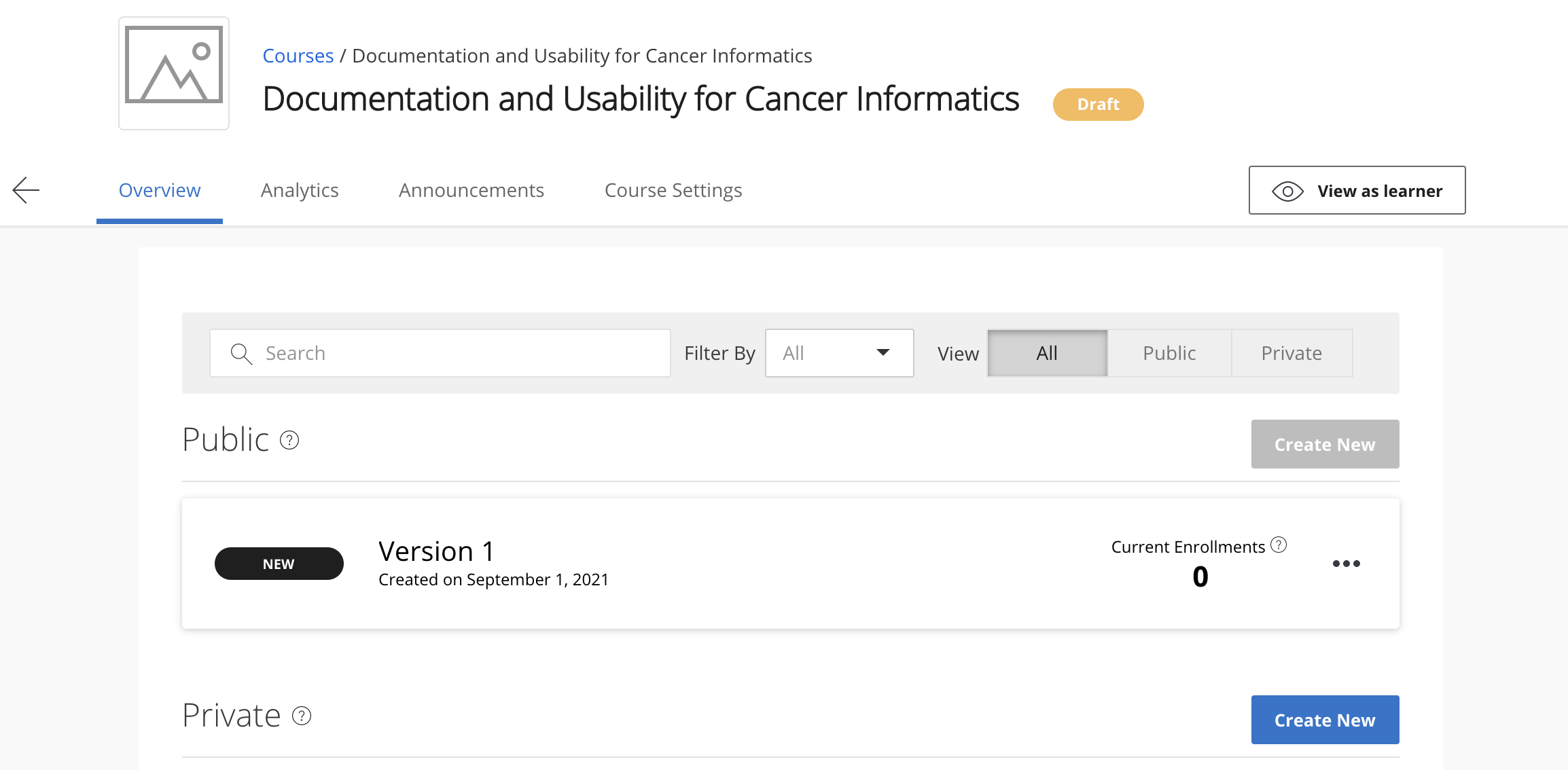
And you should see it listed:



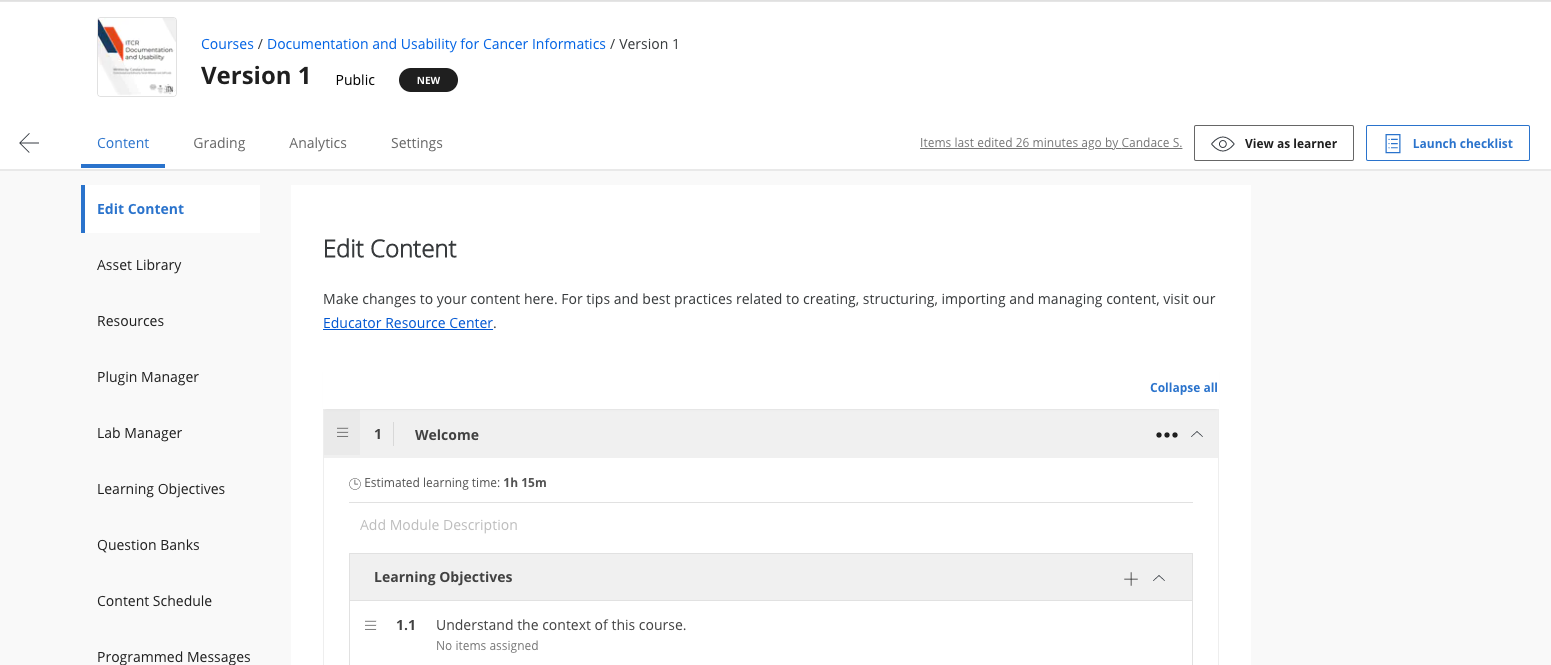
You can click on Go to Course. And on your main course page click Edit Course in the right corner.



This will bring you to your version page (at this point you should only have a Version 1, so go to that one).



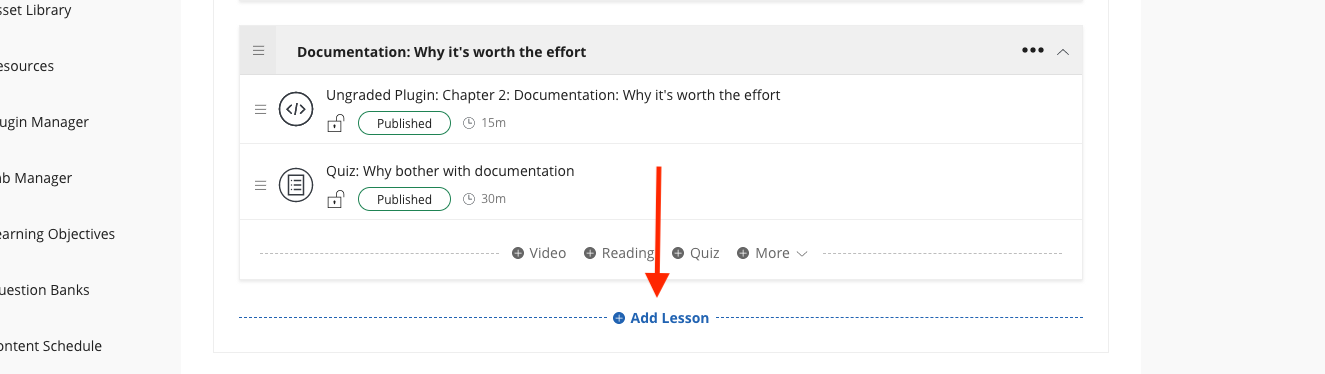
Now you are on the Edit content page and ready to add lessons, modules and quizzes!



In general, you will want each lesson to contain at least a chapter and a quiz.

Each module is supposed to take a learner one week, so depending on the length of your chapters and quizzes you may want one or two lessons/chapters/quizzes a module.

To add a new lesson, click the + Add Lesson button:



Now you are ready to add a chapter!

### 7.3.1 Adding new chapters to Coursera

You’ll need to link out each chapters in Coursera by providing a url built from your Github pages for your course.

You can see a preview of these chapters you will link to in Coursera by going to the [Github pages](https://guides.github.com/features/pages/) for your course, but adding /no\_toc at the end of your url before the chapter file name.

So the urls for each of your chapters which you will supply to Coursera in the next steps will look like this:

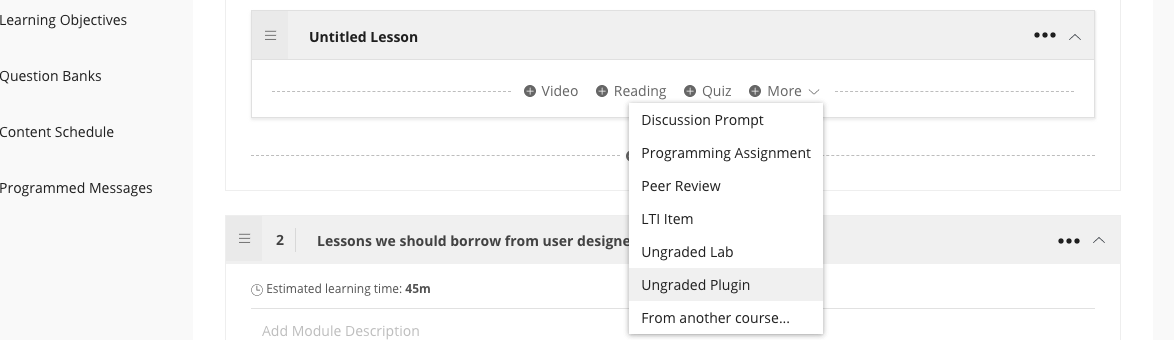
<main-github-pages-url>/no\_toc/<html\_filename>

If you are unsure what the main url for your Github pages is, in your github repository, go to Settings > Pages and you’ll see Your site is published at:

So for example, the introduction chapter of this template would be linked by using this url:

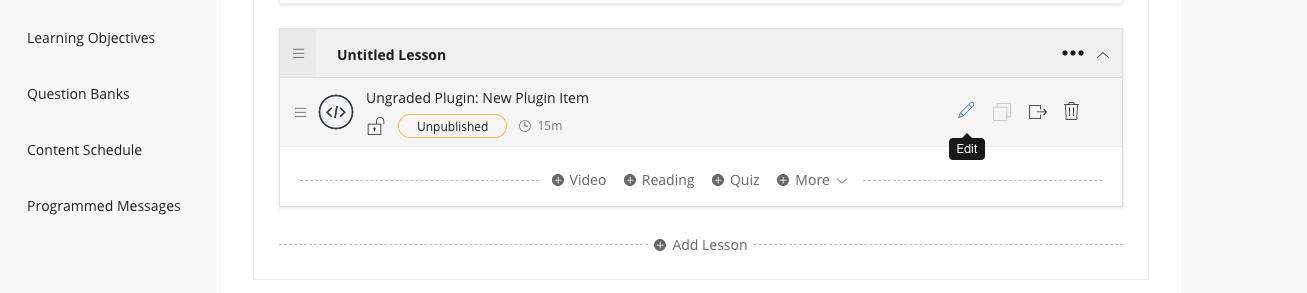
https://jhudatascience.org/OTTR\_Template/no\_toc/introduction.html

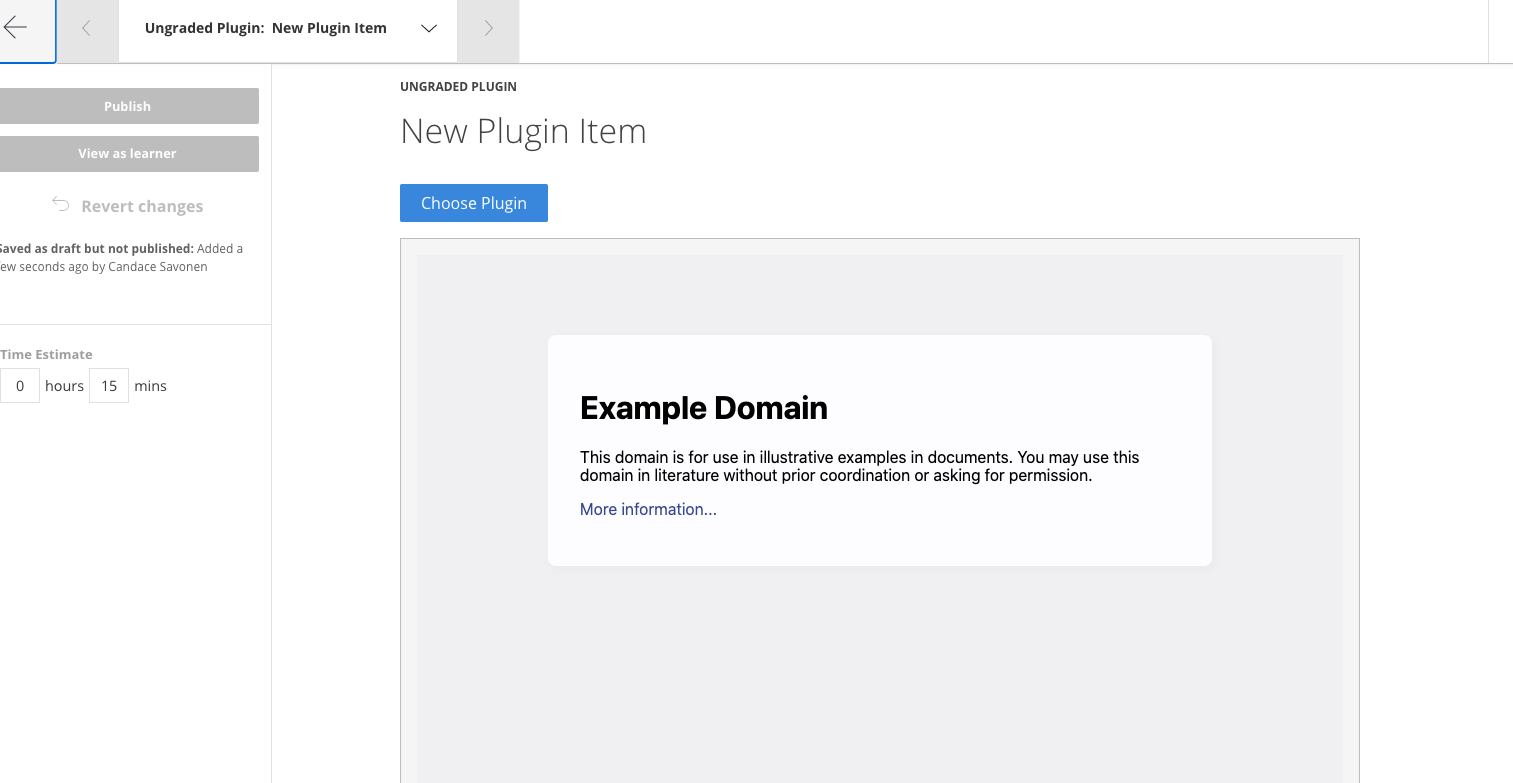
To add this URL to Coursera, add a new lesson, click + More and choose Ungraded Plugin.



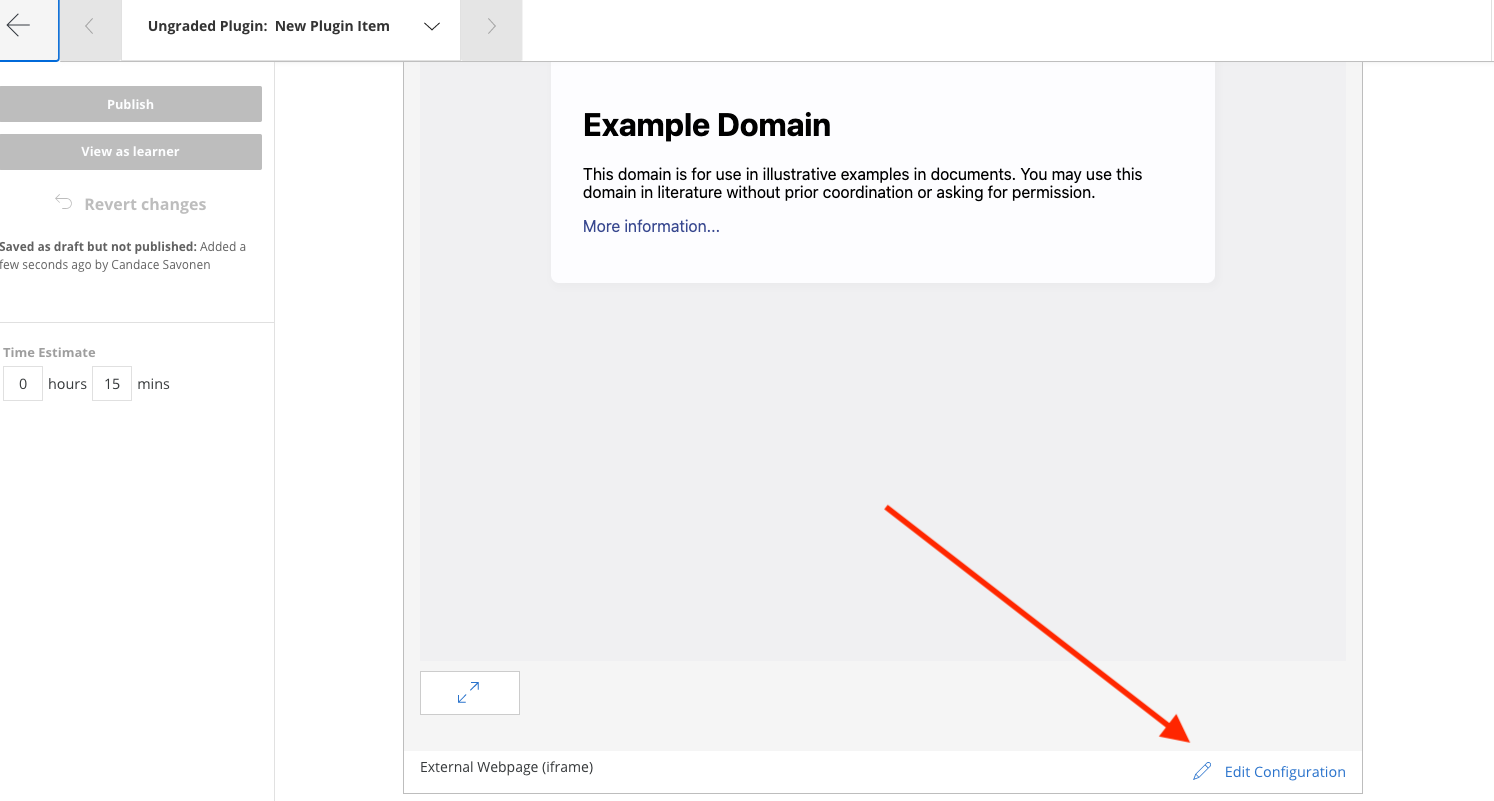
This will add an unpublished ungraded plugin to your lesson: 

Now click the edit button on the new Ungraded Plugin.

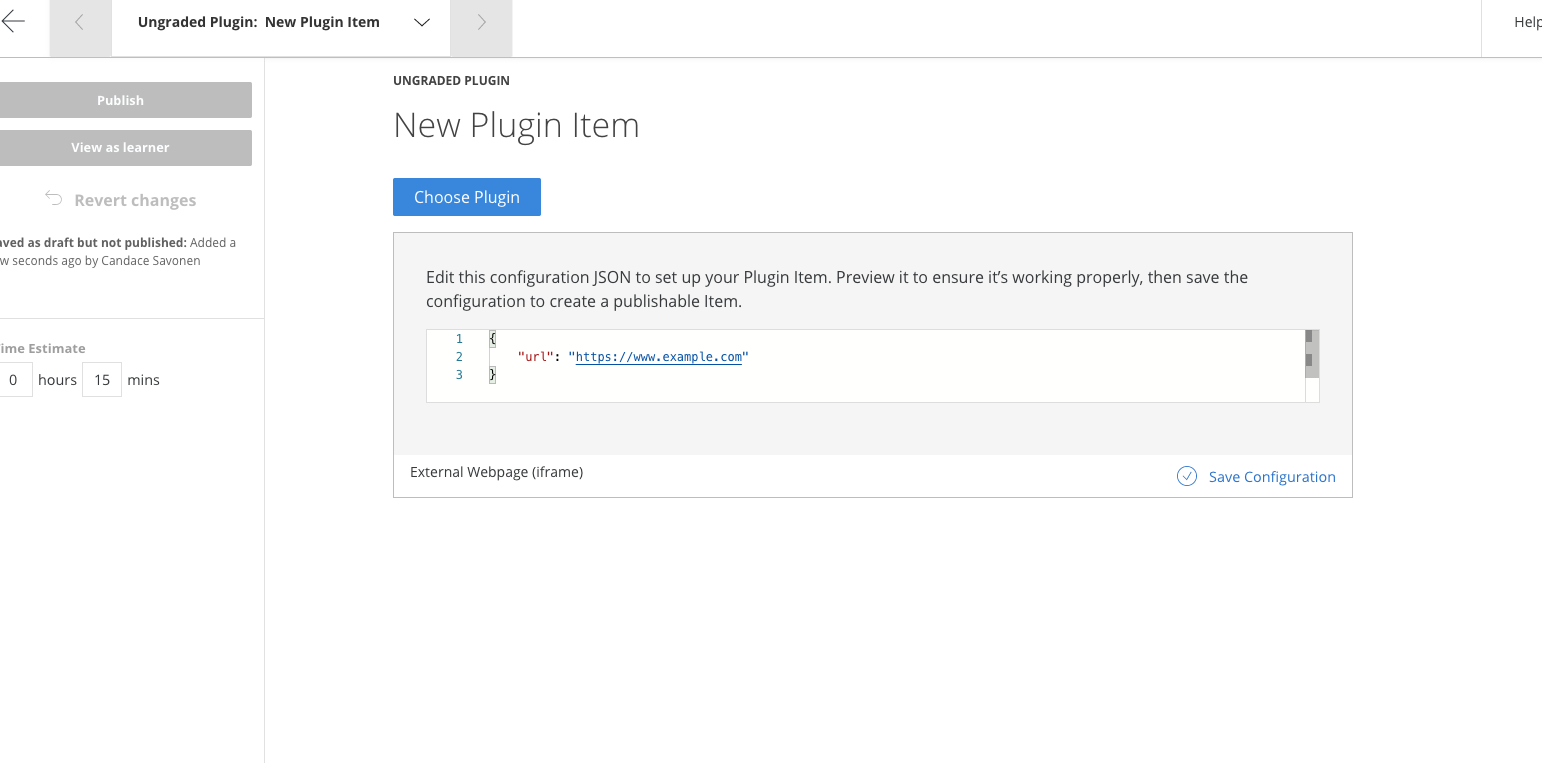


This will bring you to the standard example plug in page. 

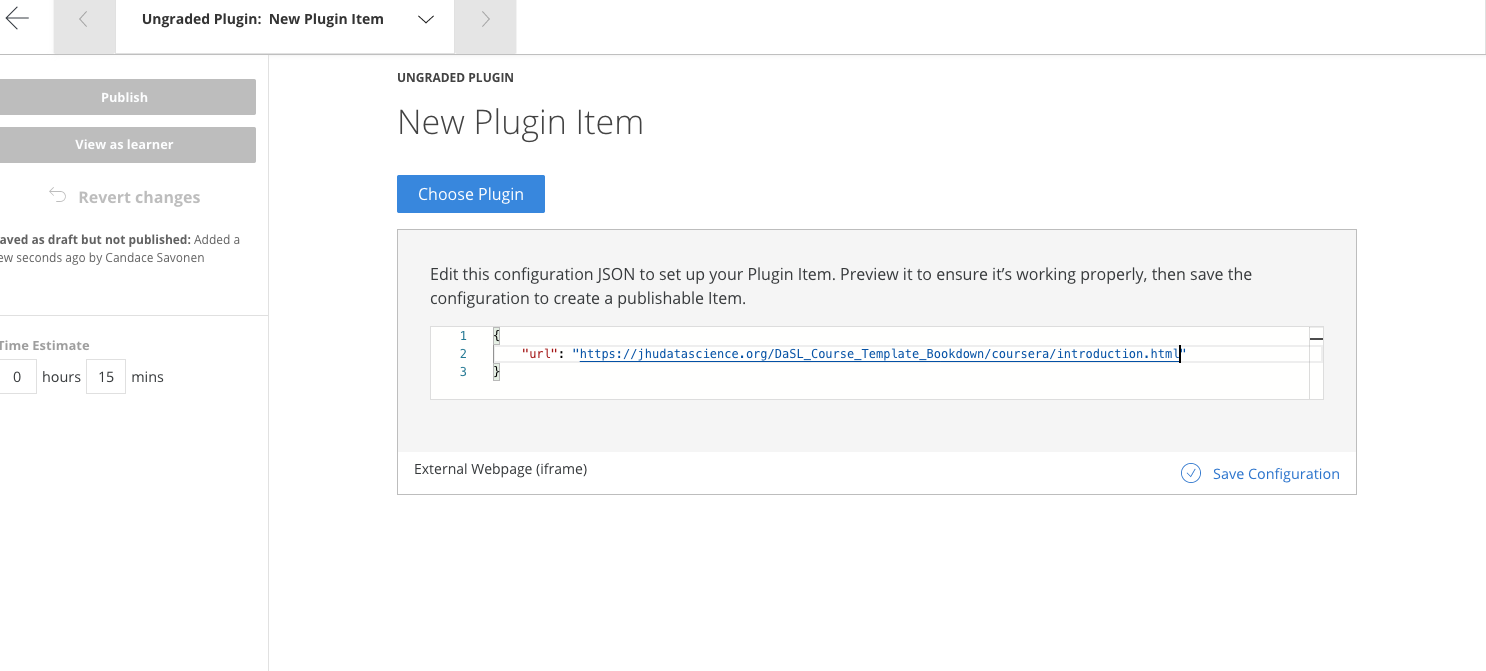
Scroll down and click Edit Configuration:



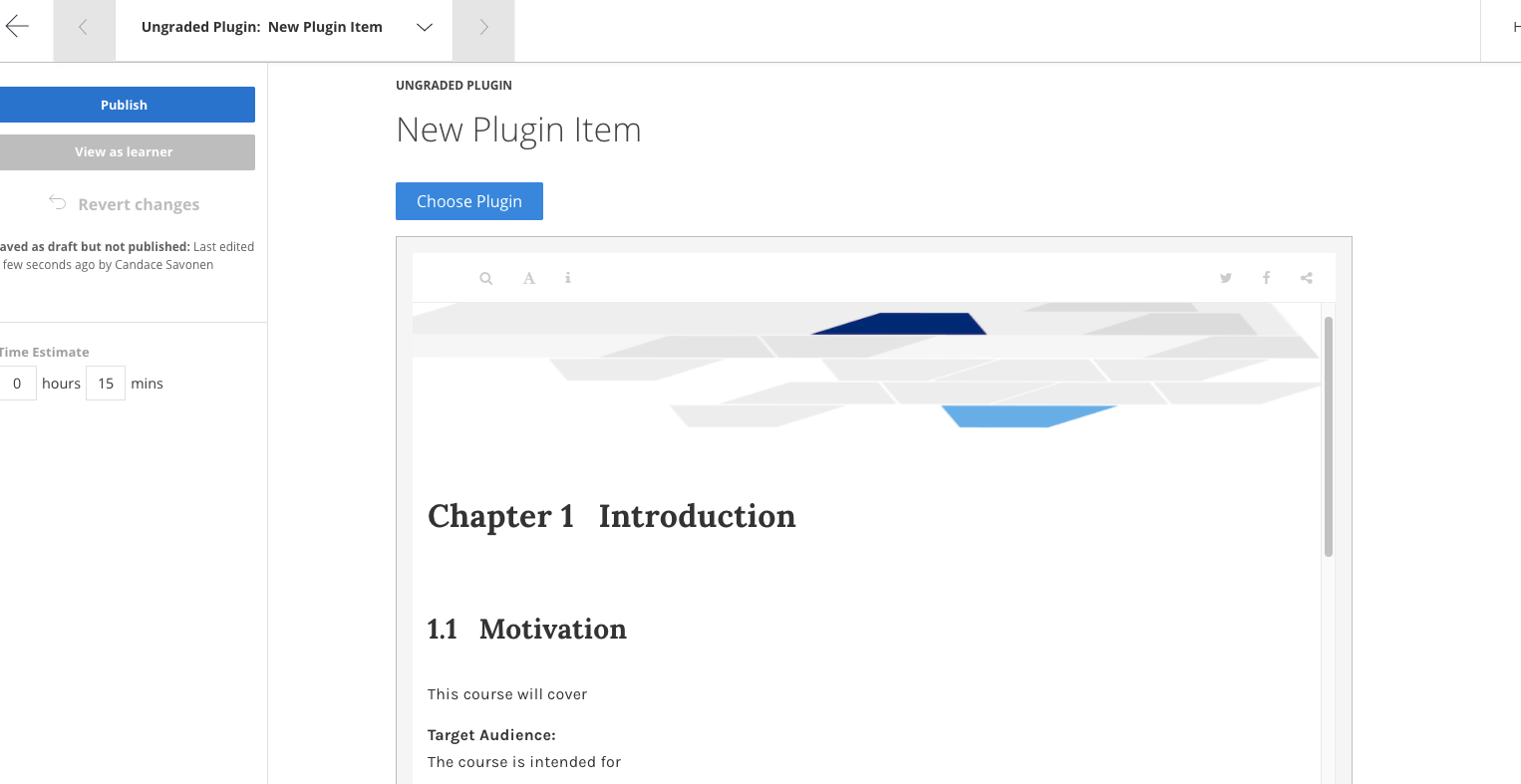
Replace the example url with the url of your individual chapter we discussed above.



Then click on Save Configuration:

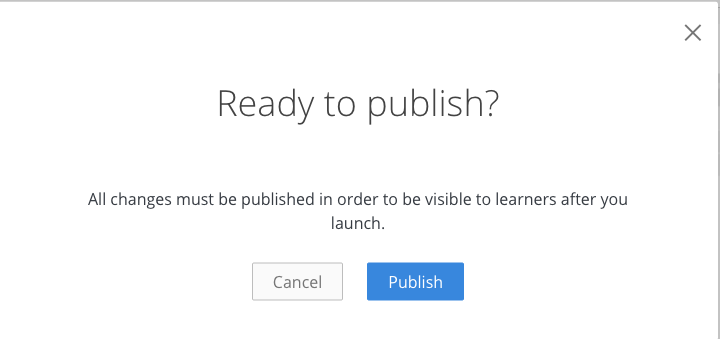


If the url you provided was correct, you should see a preview of your chapter:



You may want to edit the title at the top of this page to indicate the material being linked.

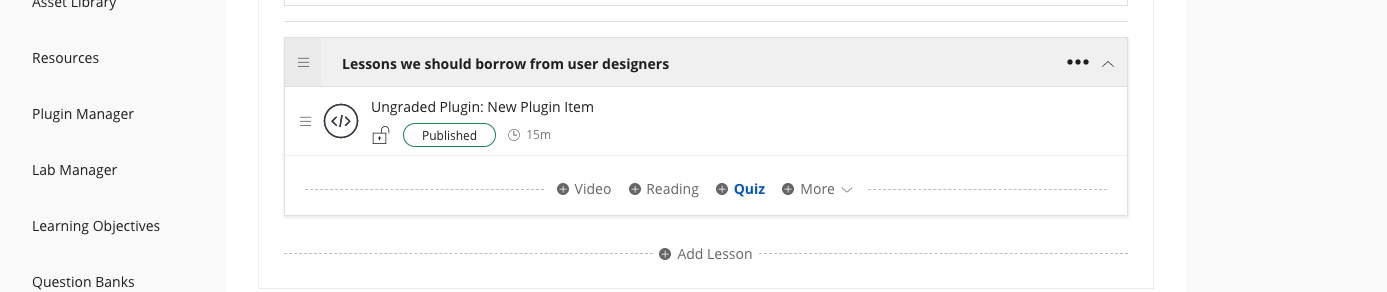
Now click Publish in the left upper corner (don’t worry it’s not actually publishing publicly, it is just saving your changes). And it will ask you again to really scare you, but yes, click Publish.



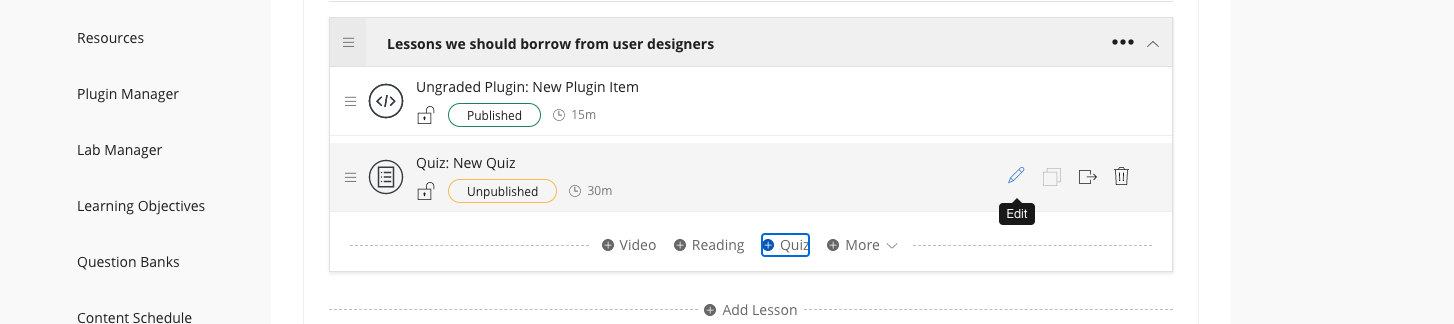
Congrats you added a chapter to your course! Now do the same thing for all your chapters!

### 7.3.2 Adding new quizzes to Coursera

To add a new quiz start from your Edit content page and on whatever lesson you would like to add the quiz to and click the + Quiz button.

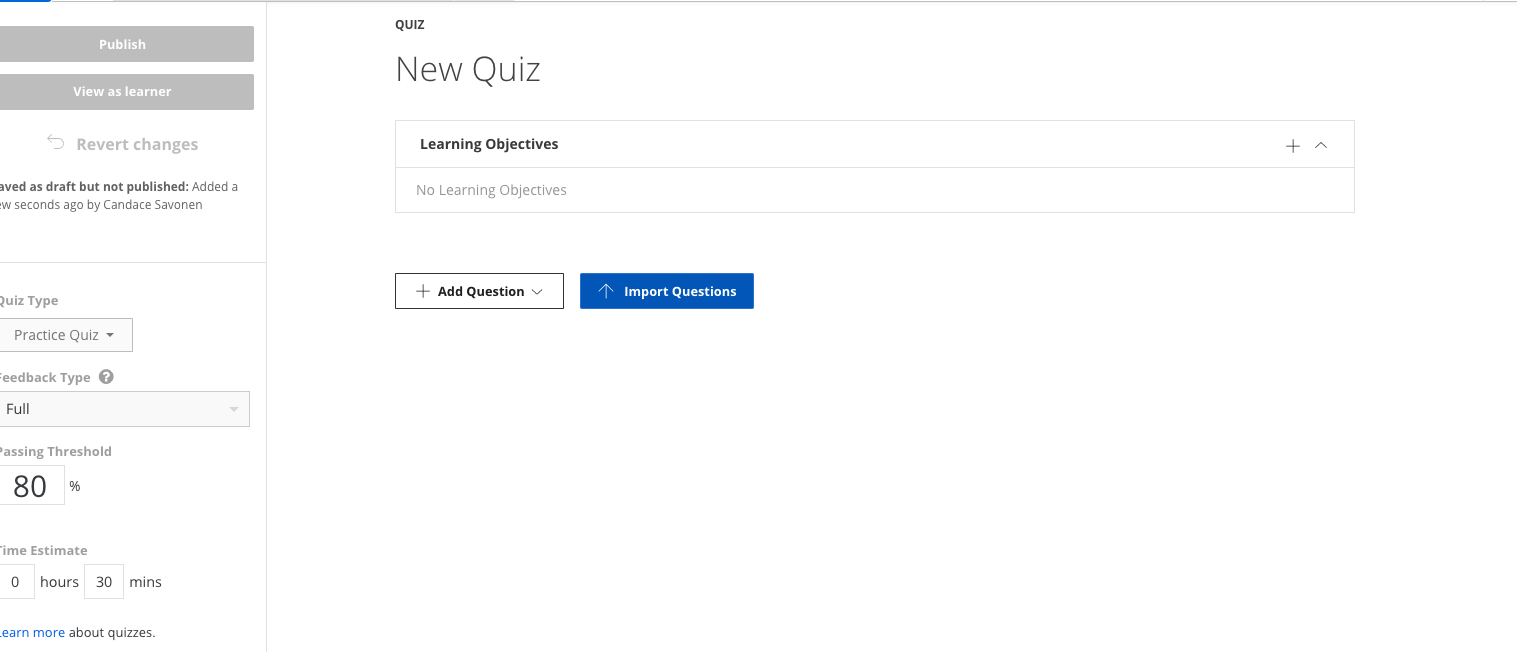


Now go to Edit on the new quiz.

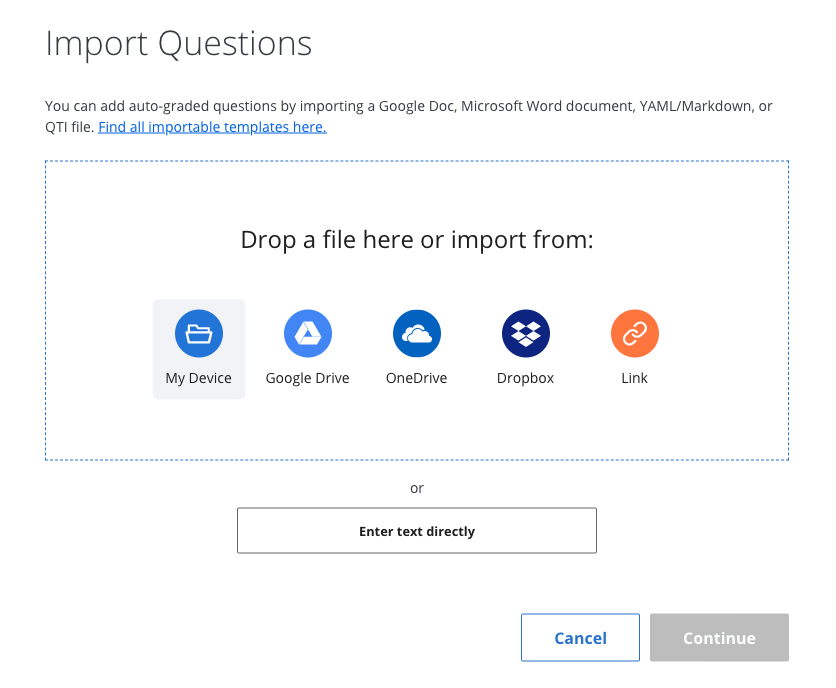


If you are choosing to manually enter your quizzes, you can continue to add each question from this page.

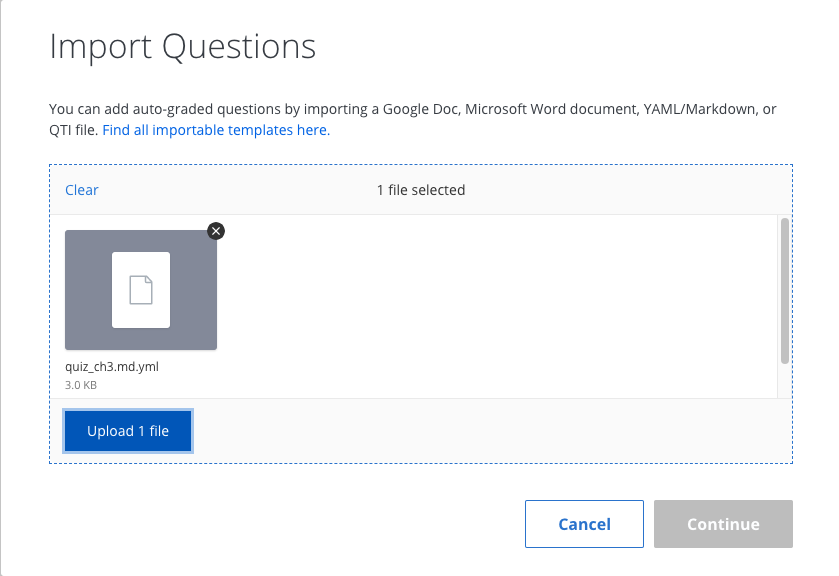
If you are importing a Leanpub converted quiz click the Import Questions button and choose your converted yml files. (If you’ve not yet converted your leanpub quizzes, then [refer to that section](#converting-quizzes-from-leanpub-format).)



Upload your Coursera quiz yml file from your device:



After you’ve selected the file click Upload:

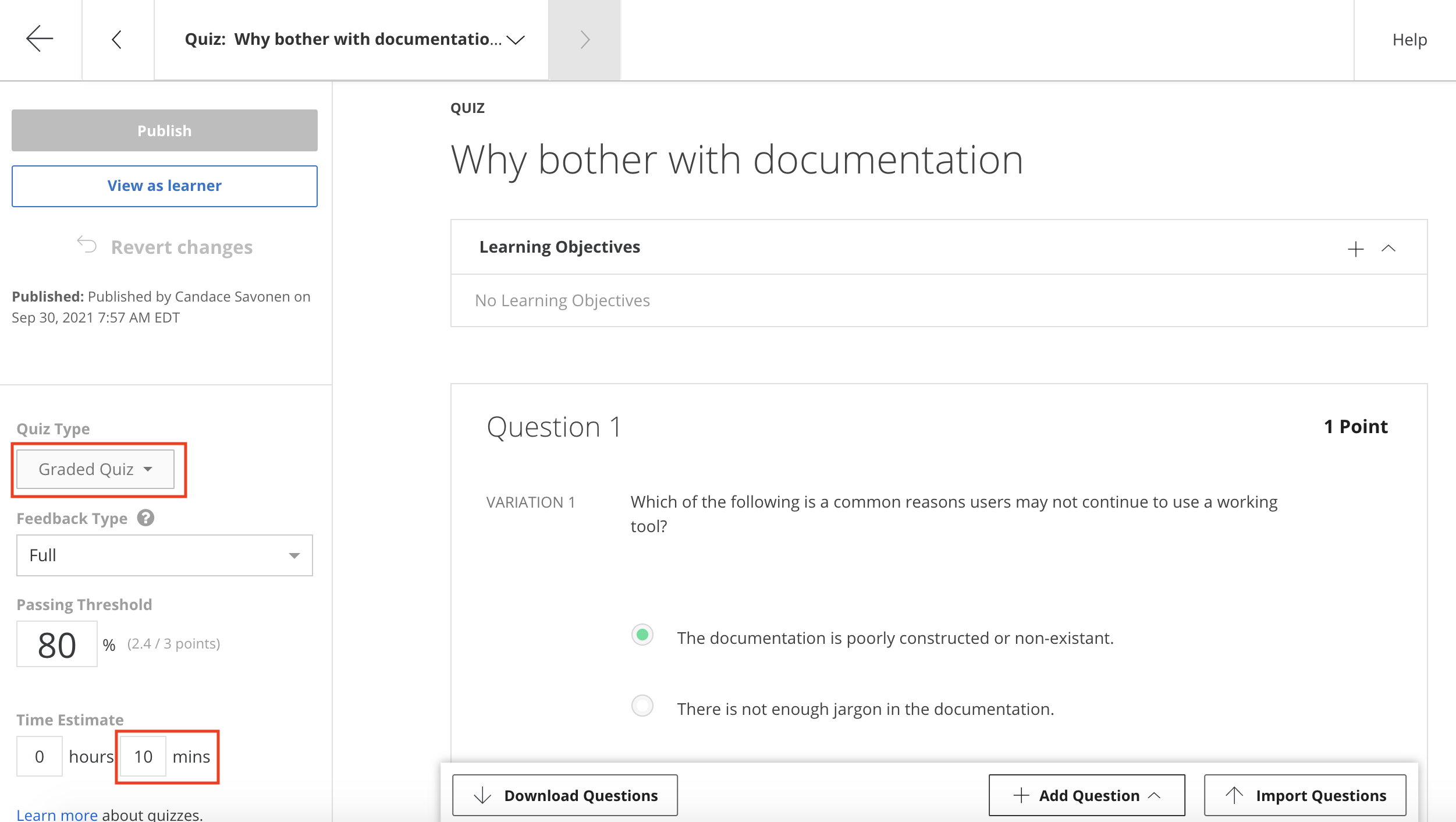


It will let you know if there are any problems uploading the quiz. If the coursera\_quiz\_conversion.R script failed to render something properly, please [file an issue here](https://github.com/jhudsl/OTTR_Quizzes/issues) describing how it failed and including a file that shows the instance it failed to convert.

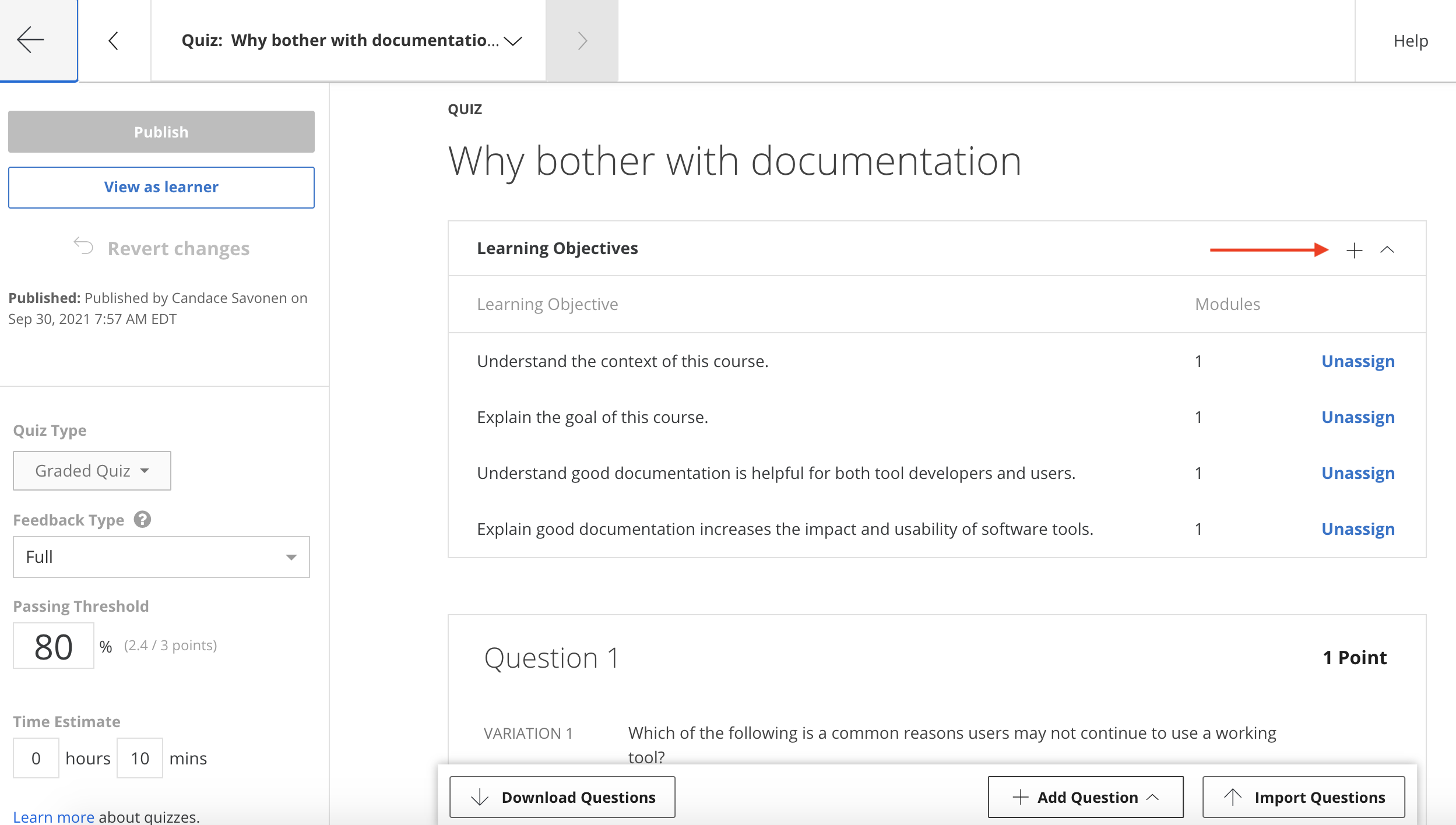
Note that images and links are not currently supported in that conversion script. You also cannot have : in your prompts or answers.

But if the quiz uploads without issue, then click Continue.

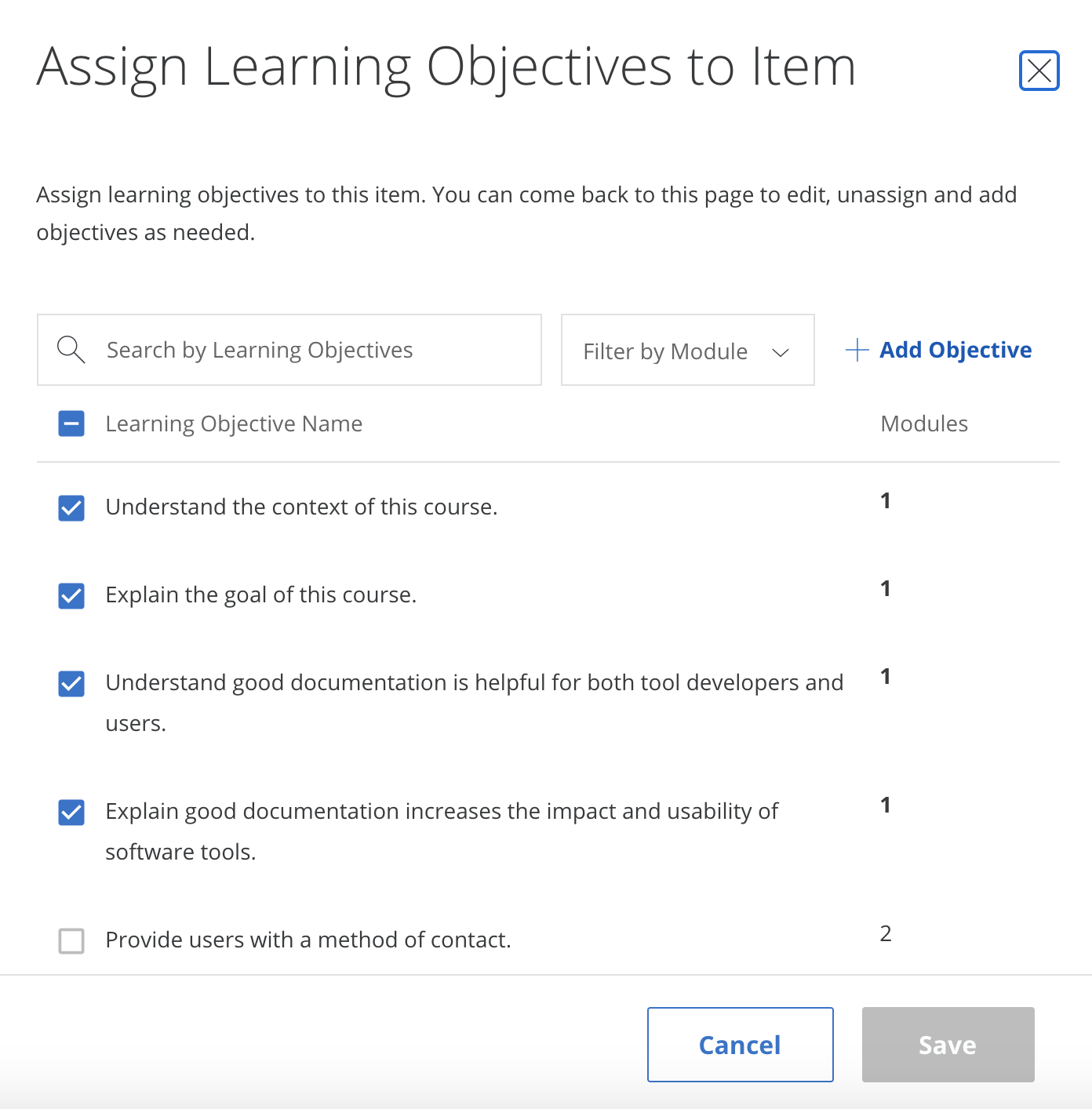
By default quizzes are set to Practice Quiz but you will need at least some of your quizzes to be used for Grading. On the left side of quiz editing page, there’s a drop down menu to select Graded Quiz. Also in this location you can edit the estimated time it should take to complete the quiz (by default it is set to 15 minutes).



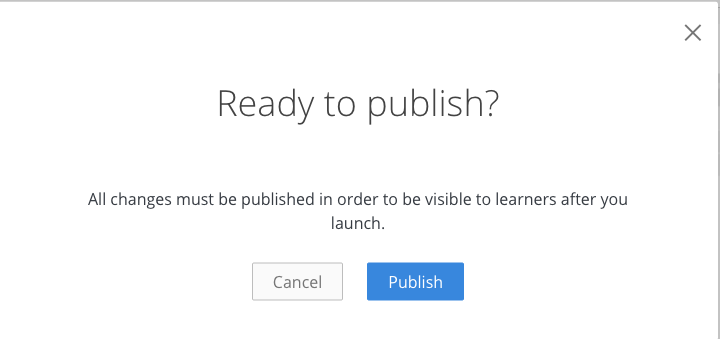
You also will want to assign the learning objectives that apply to this quiz. You can do this by clicking on the plus sign underneath the title.



From here it will suggest the module’s learning objectives. Click all that apply and then Save.



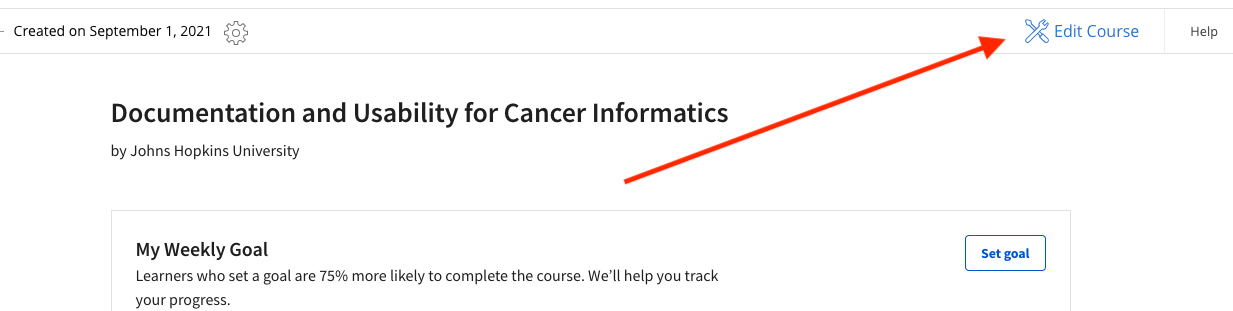
You can edit the title of your quiz and then click Publish (don’t worry it’s not actually publishing publicly, its just saving your changes). And it will ask you again to really scare you, but yes, click Publish.



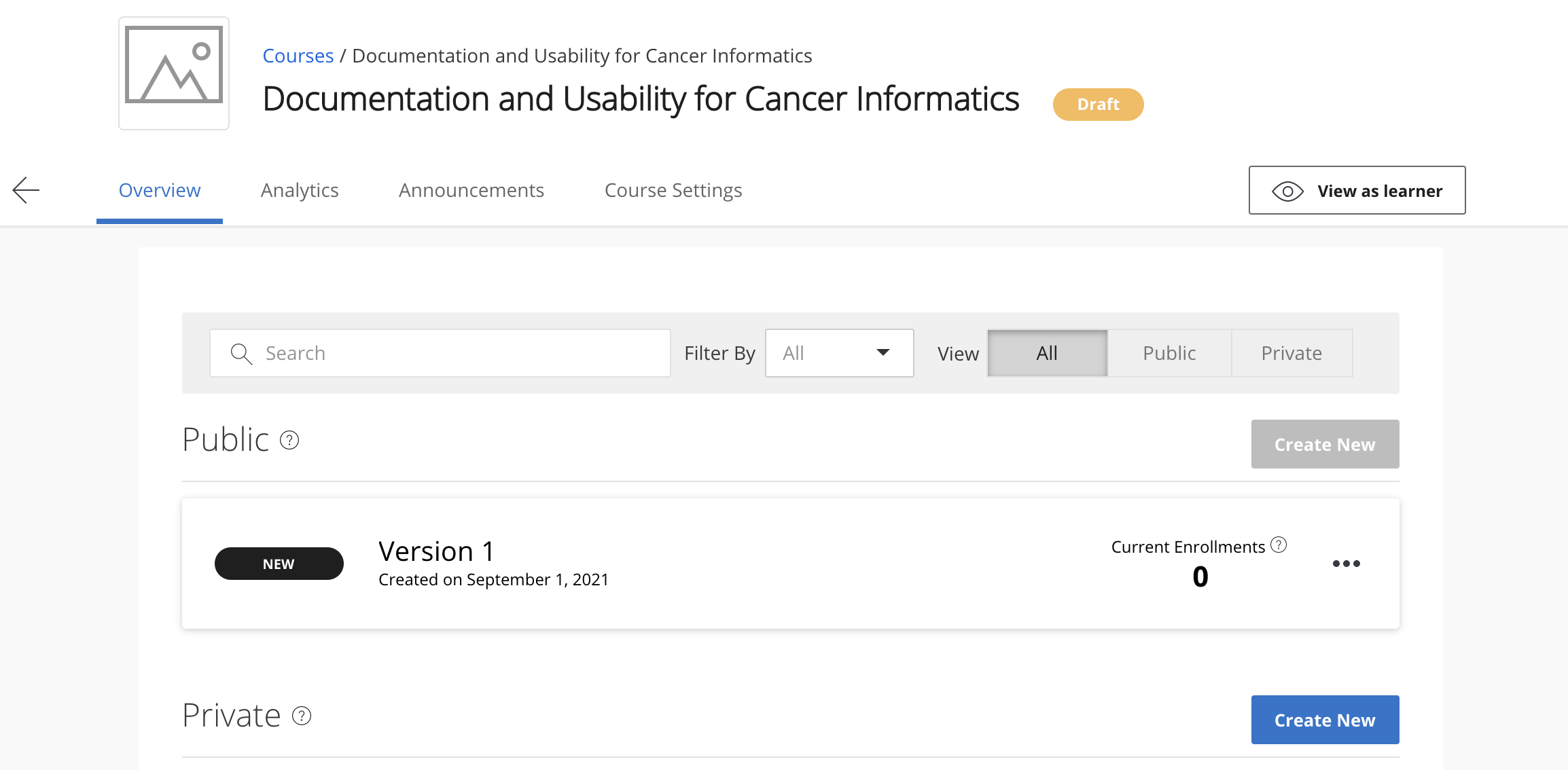
Congrats you added a quiz to your course!

## 7.4 Add programmed messages

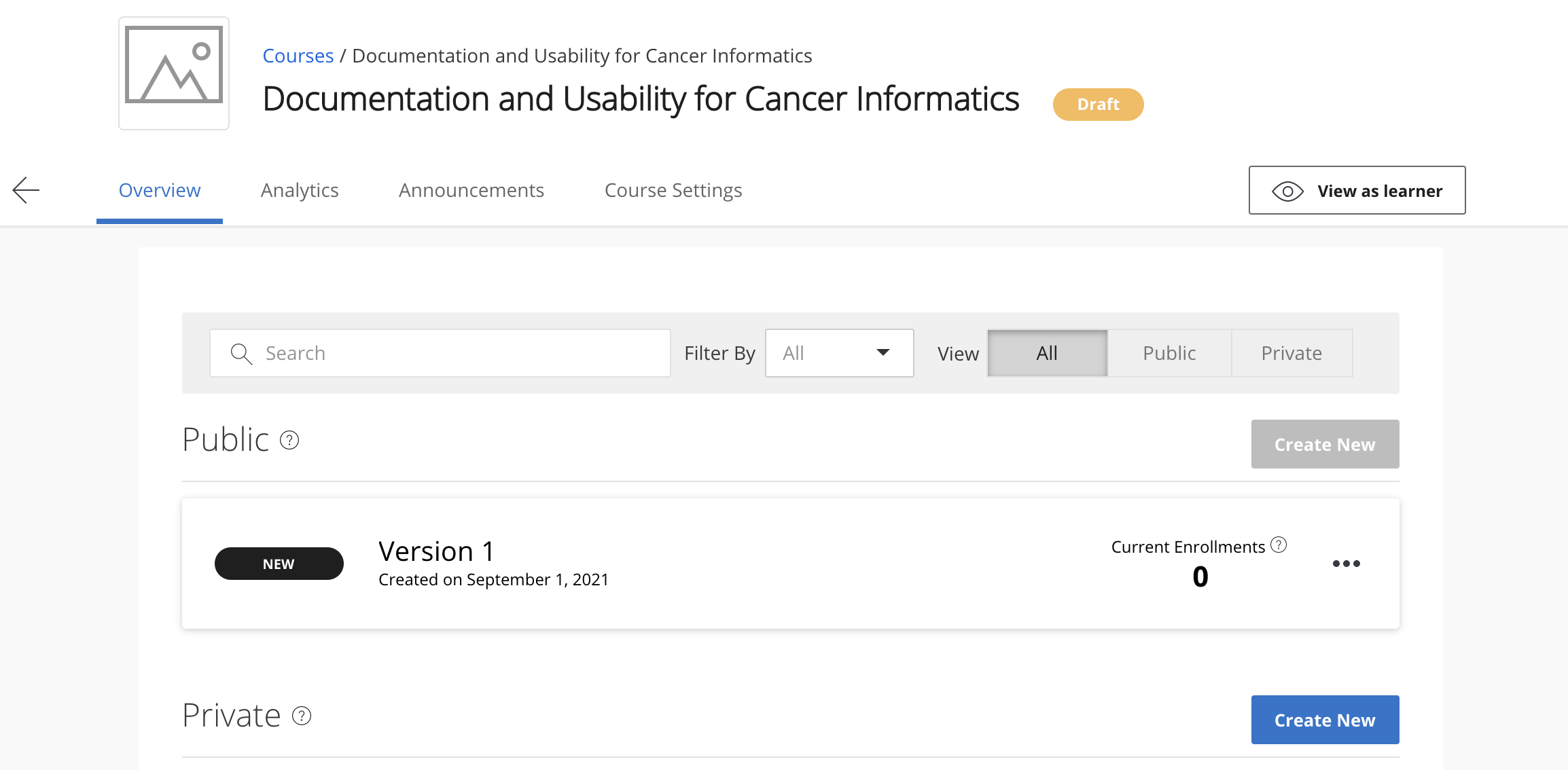
You can click on Go to Course. And on your main course page click Edit Course in the right corner.



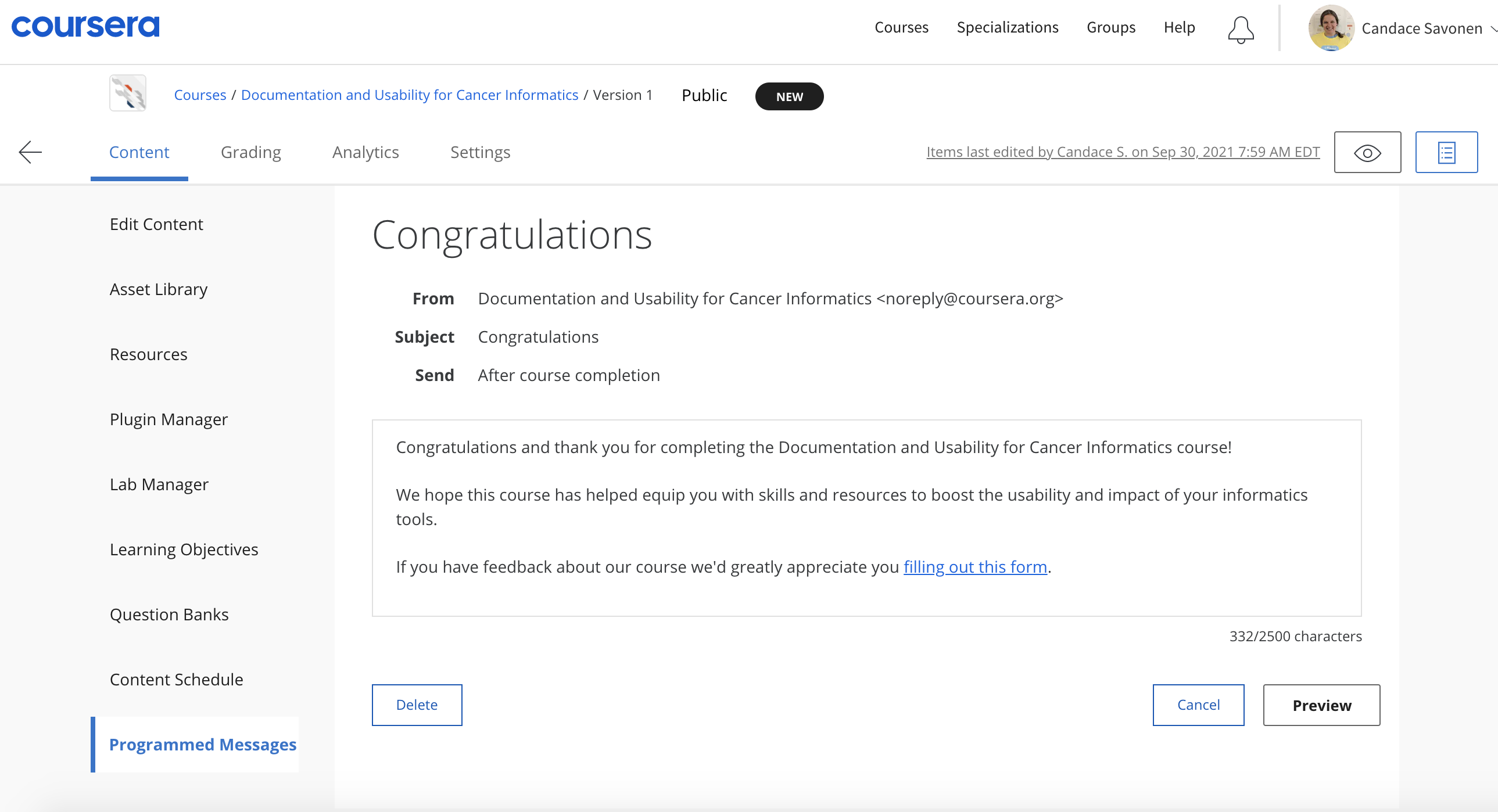
This will bring you to your version page.



Underneath Content, on the left side bar, at the bottom, there is the Programmed Messages.



At the very least you should add a Welcome to the course and a Completion programmed message. Click on each of these to edit them. Click Preview and then Publish to save your edits.



Here are some templated messages to get you started:

#### 7.4.0.1 Welcome template message:

Welcome to {Course Name}  
  
We hope this course will {What will they learn?}  
  
To get the most out of the course {What do you advise?}  
  
If you ever encounter any problems with the course, have questions or ideas, please let us know using this feedback form.

Add a link to your feedback form by clicking the link icon. Click Preview and then Publish to save your edits.

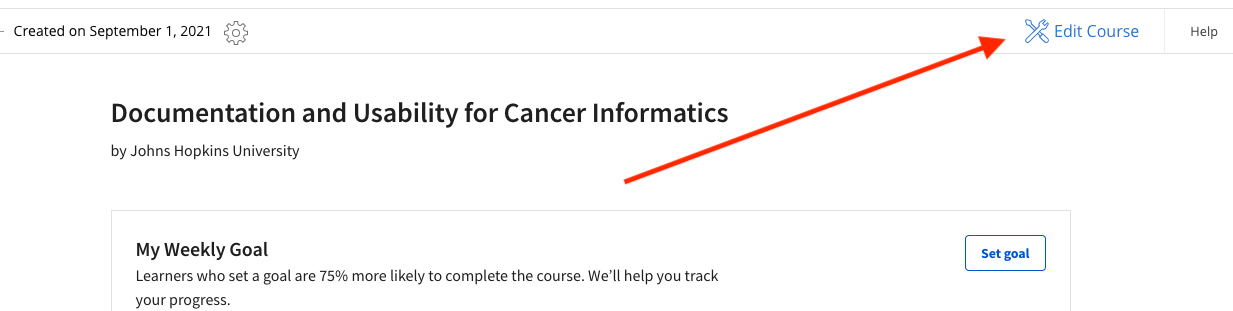
#### 7.4.0.2 Completion template message:

Congratulations and thank you for completing {name of course}!  
  
We hope this course has {What do you hope they learned}.  
  
If you have feedback about our course we'd greatly appreciate you filling out this form.

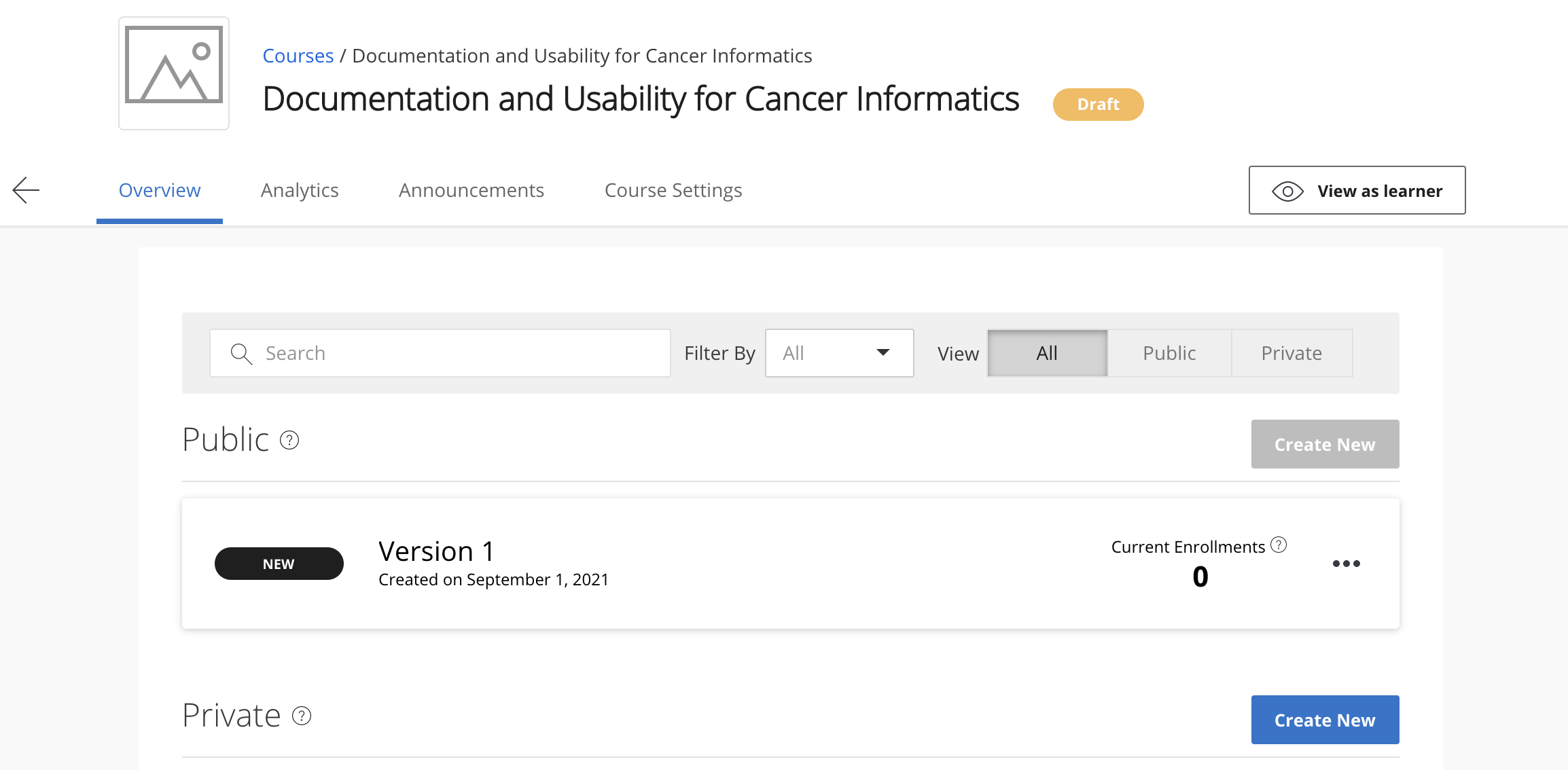
Add a link to your feedback form by clicking the link icon. Click Preview and then Publish to save your edits.

## 7.5 Grading formula

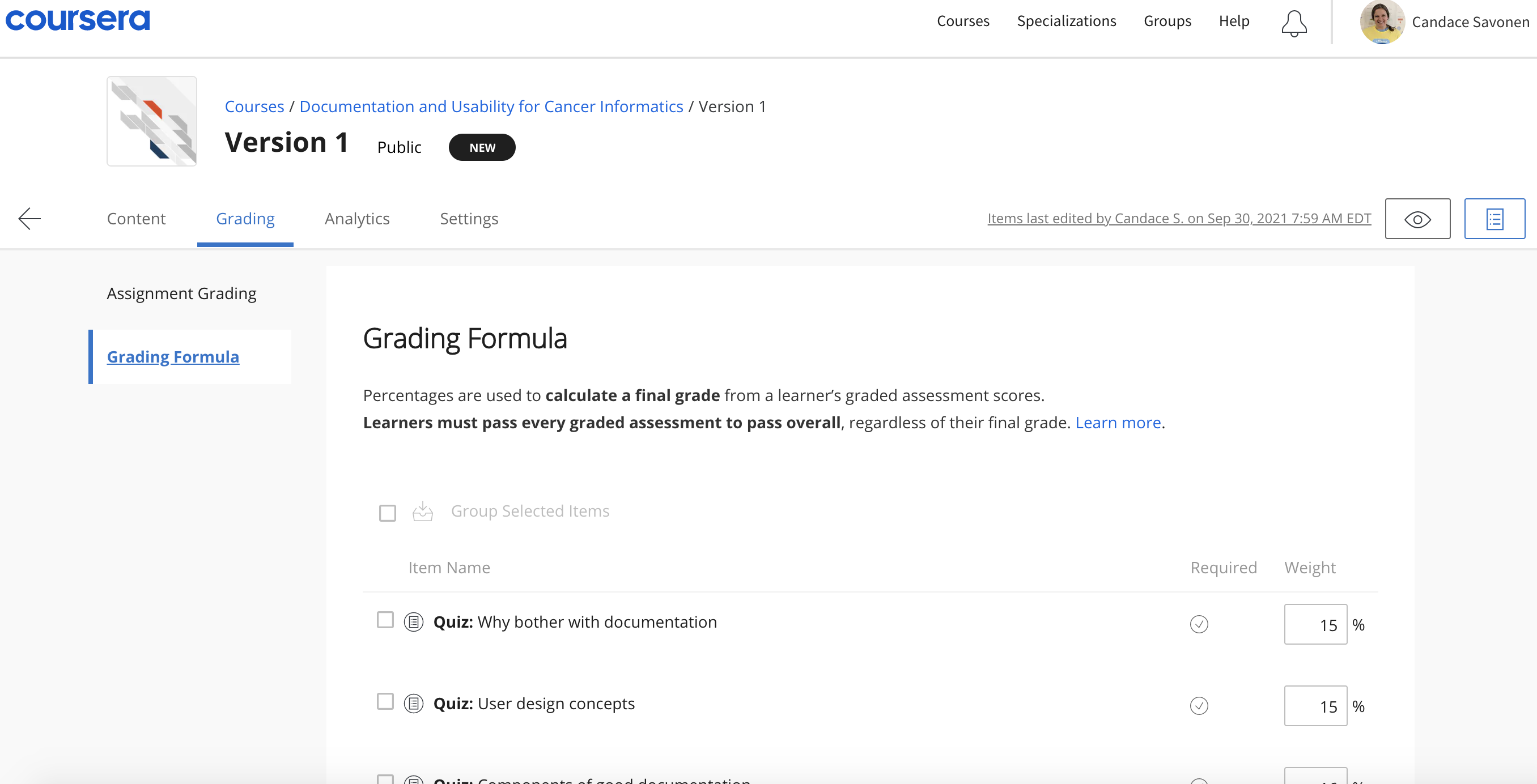
For final grades in the course, you will need to set the Grading Formula. To get to this page, you can click on Go to Course. And on your main course page click Edit Course in the right corner.



This will bring you to your version page.



Now you can set the percentage that each module’s assignments are worth in the course.



## 7.6 Content Schedule

You can find the settings for this under the Content tab. Here you can modify how each of your modules should be assigned to each week of your course.

## 7.7 Module Descriptions

Module descriptions can be found right before learning objectives when you are editing content. Adding these helps learners know a bit more about what to expect.

## 7.8 Landing Page

Add an image for your course and consider adding the following to polish your course: \* Estimated workload \* Skills \* Recommended background \* Items students will learn

# 8 Publishing with Leanpub

## Warning in readLines(dest\_file): incomplete final line found on 'resources/  
## other\_chapters/Publishing-on-Leanpub.md'

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Leanpub rendering](#leanpub-rendering)
  + [Setting up Leanpub-related GitHub actions](#X2a42f996d119c50a3e21a27b4bb8f5c551acc2b)
* [Hosting your course on Leanpub](#hosting-your-course-on-leanpub)
* [Setting up quizzes](#setting-up-quizzes)
  + [Leanpub quizzes](#leanpub-quizzes)
    - [Standard multiple choice:](#standard-multiple-choice)
    - [Choose answers](#choose-answers)
  + [Leanpub quiz formatting ‘rules’](#leanpub-quiz-formatting-rules)
* [Converting quizzes from Leanpub to Coursera format](#Xaa586d26b7961545fac959cf1009dfc1a42826f)
* [Setting up the \_Quizzes repository checklist:](#X113e783184f9581afe0e69bef11b87a7d43ca99)

OTTR contains all the files you need to publish your course to [Leanpub](https://leanpub.com/). The [ottrpal package](https://github.com/jhudsl/ottrpal) does the handling and automatic conversion to a rendered version that is ready for Leanpub.

These converted files are stored in the manuscript folder. As a general rule, don’t edit files in manuscript folder. This folder should be autogenerated by the render ottrpalGitHub action (in file.github/workflows/render-leanpub.yml`) and you in general shouldn’t make edits to it.

Note that if you wish your quizzes and their answers to remain hidden, there is a bit more set up involved with this process and you will need to create a separate private OTTR\_Quizzes repository to store these quizzes. [Start with these instructions first if you want your quizzes and answers to be hidden.](https://github.com/jhudsl/OTTR_Template/wiki/Making-quizzes-private)

## 8.1 Leanpub rendering

The ottrpal package can do most of the formatting of links and etc for you (so long as you followed the formatting prescribed by the [Setting up images section of this Wiki](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-images-and-graphics).

Github actions in this repository will attempt to do the Bookdown to Leanpub conversions for you by running ottrpal::bookdown\_to\_leanpub() function at the top of the repository.

### 8.1.1 Setting up Leanpub-related GitHub actions

When you are ready to begin publishing to Leanpub, you will want to turn on a few Github Actions that are included in the template but by default, turned off.

In your [config\_automation.yml file](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots), you will need to turn on two actions in particular: - render-leanpub: yes - check-quizzes:yes

File a pull request with these changes and merge these to main. When you file a pull request, the quiz formatting will be checked automatically and reported on in a link on a comment in your pull request.

If you encounter issues with the ottrpal package, please file an issue on its [Github repository](https://github.com/jhudsl/ottrpal/issues).

## 8.2 Hosting your course on Leanpub

To host your course on Leanpub follow these steps:

1. Make a Leanpub account here: <https://leanpub.com/> if you don’t already have one.
2. Start a course

* Click on the 3 line menu button
* Click the author tab on the far left
* Click Courses
* Click the text that says create a new course
* Fill out all the necessary information
* Select using Git and GitHub (if you work with us at JHU there is a different protocol and additional settings you need to set which you should follow - [see this document](https://docs.google.com/document/d/18UQicXwf8d25ayKGF2BrinvRgB_R2ToVn5EDOUcxyoc/edit?usp=sharing) )
* press the add to plan button

1. Preview a new version

* Click on the 3 line menu button
* Click the author tab on the far left
* Click Courses
* Click on your course name/icon
* Click “Preview New Version”
* Click Create Preview button

If you have errors in your course (typically from a quiz formatting issue) the render will fail and you will need to fix your quizzes. Also note, that occasionally the preview might fail and you should just try again, as it will work the second time. This might be due to a lag in GitHub and Leanpub communicating.

## 8.3 Setting up quizzes

Quizzes need to be stored in the quizzes/ directory. Edit and add quizzes to this directory. See and copy this [template quiz](https://github.com/jhudsl/OTTR_Quizzes/blob/main/quizzes/quiz_ch1.md) to get started. All quizzes need to be written in the Markua format. Refer to their [documentation](https://leanpub.com/markua/read#leanpub-auto-quizzes-and-exercises) (but note that it is sometimes vague or out of date). The example question types in the template are ones that are verified to work.

After you add each new quiz to the quizzes/ directory, it’s filename needs to be added in its respective spot in the Book.txt file (remember do not edit the one in manuscript but the one at the top of the repository); this ensures its incorporated by Leanpub in the correct order.

If you wanted two quizzes (one called quiz\_1.md and one called quiz\_2.md) you could duplicate and modify quiz\_1.md for your needs and then you could make the Book.txt file look like this (assuming you created a chapter called "03-chapter\_of\_course.Rmd" and you wanted quiz\_1 to be after 02-chapter-of\_course and quiz\_2 to be after 03\_chapter\_of\_course:

01-intro.md   
02-chapter\_of\_course.md   
quiz\_1.md   
03-chapter\_of\_course.md   
quiz\_2.md   
about.md

Note that any .md files with an # in front of the name in the Book.txt file will be ignored by Leanpub. We have included an example of this in the Book.txt file.

See an [example quiz here](https://github.com/jhudsl/OTTR_Quizzes/blob/main/quizzes/quiz_ch1.md) Note that you cannot have two quizzes with the same quiz\_id.

### 8.3.1 Leanpub quizzes

Leanpub is specific about how quizzes should be formatted and it won’t create a preview if any question or quiz doesn’t follow these rules. Our [GitHub action for checking quizzes](#X2a42f996d119c50a3e21a27b4bb8f5c551acc2b) (In file, .github/workflows/pull-request.yml underneath check-quizzes section) will print out a list of errors for you which you can retrieve from a pull request comment that is automatically printed. This check is only run when changes are made to the quizzes/. At this time, two types of questions are supported. (Short answers are not yet supported by our checks but will be added in the future).

##### 8.3.1.0.1 Standard multiple choice:

The answer choices are not randomized. And it looks like this:

? A question is here  
a) A wrong answer  
B) A correct answer has a capital letter  
c) A wrong answer  
d) A wrong answer

##### 8.3.1.0.2 Choose answers

You can use choose-answers option which will randomize the multiple choices or you can use a standard quiz question that doesn’t randomize. The notation for the answer choices are: C) for correct answers m) for mandatory incorrect answers and o) for optional incorrect answers.

Note that the number given for the number of answers has to be at least equal to the number of correct and mandatory incorrect answers listed.

Here’s an example:

{choose-answers: 4}  
? A question is here  
C) The correct answer is signified with a capital C  
m) A mandatory incorrect answer   
m) A mandatory incorrect answer  
o) An optional incorrect answer  
o) An optional incorrect answer

Upon merging to the main branch, a GitHub action will automatically run check-quizzes.yml that will use the ottrpal package to check that your quiz conforms to these rules. It will report the quiz formatting errors on your pull request should there be any problems.

### 8.3.2 Leanpub quiz formatting ‘rules’

* Quizzes start and end with the {quiz} and {} tags.
* Don’t have exclamation points or colons in answers.
* Make sure there’s at least one right answer.
* Check that the question and quiz attributes used are attributes recognized by Leanpub. -
* Make sure all quizzes are listed in Book.txt
* The number of choose answers is at least as big as the number of mandatory and correct answers listed.
* For choose-answers questions, use C) for correct answers m) for mandatory incorrect answers and o) for optional incorrect answers.
* Check that the question and quiz attributes used are attributes recognized by Leanpub.

## 8.4 Converting quizzes from Leanpub to Coursera format

[See more instructions on how to port a course made from these templates to Coursera](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Courseraa).

You can convert your quizzes to a nicely upload-able yaml file in this repository by running this command in the docker image

ottrpal::convert\_quizzes()

Note that currently images and links are not supported and if your quizzes contain those, you will have to manually add them at this time.

You can run this same command locally if you wish to test something. This render the chapters without the table of Contents. If you do not wish to publish to Coursera and prefer this do not run, you may delete this section (but it shouldn’t hurt anything to keep as is – unless for some reason it causes you some troubles).

Additionally, the [ottrpal](https://github.com/jhudsl/ottrpal) package has a ottrpal::convert\_quizzes() function if you choose to create quizzes and publish on both Leanpub and Coursera.

## 8.5 Setting up the \_Quizzes repository checklist:

* ☐ A [GH\_PAT has been set in the main Template and the \_Quizzes repository if you are using a separate \_Quizzes repository](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-GitHub-secrets).
* ☐ [Your Leanpub course as been created on Leanpub](#hosting-your-course-on-leanpub)
  + ☐ Writing mode in this course is set to GitHub and your main OTTR\_Template repository is linked.
* ☐ Repository is created and set to private.
* ☐ main branch has been set up:
  + ☐ Require pull request reviews before merging box is checked.
  + ☐ Require status checks to pass before merging box is checked.
    - ☐ Underneath that Require branches to be up to date before merging box is checked.
* ☐ This repo has been enrolled in [automatic updates from the original template](https://github.com/jhudsl/OTTR_Template/wiki/Receiving-template-updates).
* ☐ Your Book.txt file is up to date.
* ☐ The [ottrpal::bookdown\_to\_leanpub() ran successfully](#leanpub-rendering).

# 9 Publishing quizzes with answers hidden

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [How to use these repositories:](#how-to-use-these-repositories)
* [Setting up your OTTR\_Quizzes Github repository](#X2d603bfd7fcdaef593ffeff33788a3930a7d230) - [Set up branches](#set-up-branches)
* [Linking to your OTTR\_Template course Github repository](#X1040b71f097dcc4fd918721d63ed76ed110dc5e)

If you prefer that your quizzes and the answers remain hidden, you can use a companion [OTTR\_Quizzes](https://github.com/jhudsl/OTTR_Quizzes) repository to store them.

## 9.1 How to use these repositories:

The Leanpub repository is where quizzes are stored because it is meant to be kept private so answers are hidden.

Keep in mind that in order to keep your Leanpub and Bookdown versions of your course concurrent, you should not make chapter edits in this repository! Only quiz edits should be in your \_Quizzes repository. Chapter changes should be made in your OTTR\_Template course repository and the [transfer-rendered\_files.yml](https://github.com/jhudsl/OTTR_Template/blob/main/.github/workflows/transfer-rendered-files.yml) Github action should be used to copy over the changes here.

## 9.2 Setting up your OTTR\_Quizzes Github repository

Go to the [OTTR\_Quizzes repository](https://github.com/jhudsl/OTTR_Quizzes) and click Use this template. You must name this repository identical to your main OTTR repository but ending in \_Quizzes if you would like them to be linked. For example, the jhudsl/Documentation\_and\_Usability course has a corresponding quiz repository named jhudsl/Documentation\_and\_Usability\_Quizzes.

Make sure to set your new repository to Private so your quiz answers will be hidden.

##### 9.2.0.0.1 Set up branches

\*These settings are the same as we used in the Bookdown repository so we will need to set them up in the same way.

Go to Settings > Branches and click Add rule. For Branch name pattern, put main.

*Protect the main branch*:  
Then check the box that says Require pull request reviews before merging.

*Make sure branches are updated*:  
- Check the box that says Require status checks to pass before merging. - Underneath this also check the box that says Require branches to be up to date before merging.

*Use automatic spell and URL checks*:  
After the first pull request, a couple of checks will automatically happen and then appear here in settings. Then, you can require these checks to pass before merging pull requests by returning here and selecting them - they are url\_check and sp-check they will check that the urls work and that the quizzes do not have spelling errors. See the [Github Actions section in the Bookdown repository](#github-actions) for more details on these.

After setting up these new branch items, click Create and Save changes.

## 9.3 Linking to your OTTR\_Template course Github repository

In order to link your \_Quizzes and OTTR\_Template course repositories (so you only have to edit material in one place), you need to name your GitHub repository with an identical name to your OTTR\_Template course repository except end it in \_Quizzes. So for example, if your Bookdown repository is called: Cool\_Course; it’s Leanpub quiz repository **must** be called Cool\_Course\_Quizzes.

The GitHub actions that are responsible for content transfer is in the .gihub/workflows/ folder and called [transfer-rendered-files.yml](https://github.com/jhudsl/OTTR_Template/blob/main/.github/workflows/transfer-rendered-files.yml)

Once build-all is run, the docs/ folder where the rendered files are place are copied over to the Leanpub repository and filed as a pull request. When you are sure that you want the changes from your main OTTR\_Template repository, you can merge that pull request.

Note if you haven’t set a [GH\_PAT git secret](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-GitHub-secrets) and you are not a part of jhudsl organization, you will need to set that by following [these instructions](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-GitHub-secrets).

\_Note that any content changes to non-quiz material needs to be done your course’s Bookdown repository! Do NOT change them in your *Quizzes repository, otherwise your OTTR\_Template course will not be updated.*

# 10 Citations

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Citations](#citations)

## 10.1 Citations

You can generally follow the [Bookdown instructions about citations](https://bookdown.org/yihui/rmarkdown-cookbook/bibliography.html), but you don’t need to add the additional bibliography argument at the top of the Rmds.

To add a new reference source, add to the book.bib file, keeping your new entry in alphabetical order.

For articles (or anything with a DOI), go to [doi2bib.org](https://www.doi2bib.org/) or [ZoteroBib](https://zbib.org/) to get a BibTex-formatted reference that you can copy and paste the reference to the book.bib file.

You can also use programs like Zotero or Endnote to export a .bib file and either combine with the book.bib or manage your references from there.

Other sources can be added using this template:

@website{citekey,  
 author = {First Last},  
 title = {Title},  
 url = {http://www.someurl.html},  
}

To reference the citations in your writing follow the [bookdown instructions](https://bookdown.org/yihui/rmarkdown-cookbook/bibliography.html):

Items can be cited directly within the documentation using the syntax @key where key is the citation key in the first line of the entry, e.g., @R-base. To put citations in parentheses, use [@key]. To cite multiple entries, separate the keys by semicolons, e.g., [@key-1; @key-2; @key-3]. To suppress the mention of the author, add a minus sign before @, e.g., [-@R-base].

See [Chapter 2](https://github.com/jhudsl/OTTR_Template/blob/main/02-chapter_of_course.Rmd) of this template course for examples.

# 11 About Docker

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Using the jhudsl/course\_template Docker image](#X99796ef2ebe8b5c7e1cda8871ed4945c552e155)
  + [Optionally run RStudio from the Docker container](#X7b9c2cafe538c2a4b794512dd7d45a041c620a2)
* [Starting a new Docker image](#starting-a-new-docker-image)
  + [Set Dockerhub related secrets](#set-dockerhub-related-secrets)
  + [Updating workflows for new Docker image](#updating-workflows-for-new-docker-image)
  + [Modifying the Dockerfile for a new image](#modifying-the-dockerfile-for-a-new-image)
    - [Template commands for adding packages to the Dockerfile](#X01401151311a902a20264e45390db4bd18a1aca)
      * [CRAN Packages:](#cran-packages)
      * [Bioconductor Packages:](#bioconductor-packages)
      * [Python Packages:](#python-packages)
  + [Testing a modified Docker image](#testing-a-modified-docker-image)
    - [Pushing the Docker image](#pushing-the-docker-image)

Particularly for courses that involve running example code, it’s highly recommended that you use a Docker image for development to maintain software version consistency across course developers.

## 11.1 Using the jhudsl/course\_template Docker image

If your course doesn’t require any additional packages to run, then you do not need to set up Docker locally but [this Docker image: jhudsl/course\_template](https://hub.docker.com/repository/docker/jhudsl/course_template) will run and re-render all of your changes as you add them.

We advise you use the jhudsl/course\_template:main tagged image as opposed to the latest. The main is what is ready for use, while the latest may be under development. All GitHub actions by default use the jhudsl/course\_template:main.

If you are new to Docker, you may find it helpful to read this [introduction to Docker](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/launching-a-docker-image.html). - You will need to [sign up with a Docker account](https://hub.docker.com/) if you don’t have one.  
- If you haven’t installed Docker desktop (or need to update it), you can do [so here](https://docs.docker.com/get-docker/).

If your Docker desktop is running, you should see a Docker whale in your tool bar. On Macs, this will be on the bar on the top of your screen; in Windows, on the bottom right.

A Docker image is similar to a virtual machine - it contains preinstalled software in a preconfigured environment. Docker images can be downloaded from DockerHub, or you can create your own.

We have created the course\_template:main image as a starting point; you can download it from jhudsl/course\_template:main on DockerHub using the docker pull command we have below. To pull the docker image associated with this template, you can run this command below in your command line. This may take a while.

docker pull jhudsl/course\_template:main

This pulls the course\_template:main image from Docker Hub and copies it to your computer. It will be placed in your local collection of Docker images, managed by Docker (not in your pwd). If you get an error, it may be because you forgot to have your Docker desktop running… see above.

To use the Docker image associated with the course template, first navigate to the the top of this GitHub repository. Now you can start up the Docker container using the command below.

This runs your local copy of the course\_template:main image (which you downloaded from DockerHub). The option -v $PWD:/home/rstudio mounts pwd (this repo) and makes it available at /home/rstudio within the container. Replace all of <CHOOSE\_PASSWORD> (including the < and >) with a password of your choosing.

*On a Mac*:

docker run -it -v $PWD:/home/rstudio -e PASSWORD=<CHOOSE\_PASSWORD> -p 8787:8787 jhudsl/course\_template:main

*On a Windows*:

docker run -it -v %CD%:/home/rstudio -e PASSWORD=<CHOOSE\_PASSWORD> -p 8787:8787 jhudsl/course\_template:main

Do not close this window, but you can minimize it. Open up a new command line window and run the command: docker ps, you should see a container is up and running!

*Couple other handy Docker commands:*

* To stop your Docker container, run docker ps to obtain the docker container ID. Then you can use that ID to run docker stop <CONTAINER\_ID>.
* To remove a docker image (which you may need to do from time to time to clear out space), you can run docker image ls to see all your current images. Then you can run either docker image rm <IMAGE\_ID>.
* If you really need to clear out space, you can follow this [StackOverflow post](https://stackoverflow.com/questions/44785585/docker-how-to-delete-all-local-docker-images) on how to remove all images and all containers.

For more info on how to use Docker, they have very [extensive documentation here](https://docs.docker.com/).

#### 11.1.0.1 Optionally run RStudio from the Docker container

In a web browser navigate to the following to get to an RStudio that is run from your Docker container.

localhost:8787

To log in, you’ll need to use rstudio as the username and whatever password you put for <CHOOSE\_PASSWORD> in the above command.

## 11.2 Starting a new Docker image

Should you find that your course needs additional packages beyond what’s included in the template, you should probably start a new Docker image and you’ll need to do these steps to get this going:

1. Create a [Dockerhub account](https://hub.docker.com/signup) if you don’t have one.
2. Create a new Docker image on your Dockerhub account. [Follow these instructions if you don’t know how to do that](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/launching-a-docker-image.html).
3. [Set up Dockerhub secrets in your repository or organization](#set-dockerhub-related-secrets).
4. [Update the Github actions workflows accordingly](#updating-workflows-for-new-docker-image).
5. [Edit the Dockerfile in your repository](#modifying-the-dockerfile-for-a-new-image).

### 11.2.1 Set Dockerhub related secrets

You only need to do this once as an organization (if this course is under jhudsl you don’t need to do this step).

To give them permission for all these Docker actions, you need to set a GitHub secret. Go to Settings > Secrets and scroll down to organization secrets or you can set this for each repository by creating a repository secret.

* Click New repository secret button for each of these secrets
* The login information in these GitHub secrets must be from an account that has write permissions to the Docker image you wish to manage).

*Name: DOCKERHUB\_USERNAME*:  
For value: put your login username for <https://hub.docker.com/>

*Name: DOCKERHUB\_TOKEN*:  
For value: put an access token for Dockerhub. You can create this by following [these instructions](https://docs.docker.com/docker-hub/access-tokens/#create-an-access-token).

### 11.2.2 Updating workflows for new Docker image

To make sure that your new Docker image is being used for rendering in the GitHub actions, you need to change the rendering-docker-image: from the default of jhudsl/course\_template:main and replace it with your docker image tag.

Then start a new branch so that you can submit a new pull request with your changes.

Now, when you file a pull request, the Dockerfile build for your docker image will be tested automatically by the [GitHub actions](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots) if docker-test is set to yes. The default is docker-test: no

### 11.2.3 Modifying the Dockerfile for a new image

[Read this chapter for instructions on how to modify Docker images](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/modifying-a-docker-image.html)

You will probably want to create your Docker image by using the jhudsl/course\_template:main as your base – this means that all the packages that are in our jhudsl/course\_template:main image will be included in the Docker image you build.

FROM jhudsl/course\_template:main

You can take a look at this [Dockerfile template we’ve set up here](https://github.com/jhudsl/OTTR_Template/blob/main/resources/TEMPLATE_Dockerfile) (note that the commands would need to be uncommented and real package names put in place of package\_name’s).

#### 11.2.3.1 Template commands for adding packages to the Dockerfile

This section gives you the basics on what it looks like to add new packages to your new Docker image.

##### 11.2.3.1.1 CRAN Packages:

For R packages installed from CRAN, you can add to the running vector list of R packages.

##### 11.2.3.1.2 Bioconductor Packages:

To add an R package from Bioconductor, you can follow this kind of format:

RUN Rscript -e "options(warn = 2); BiocManager::install( \  
 c('limma', \  
 'newpackagename')

##### 11.2.3.1.3 Python Packages:

To add a Python package, you will need to add pip3 to install Python packages using this format:

RUN pip3 install \  
 "somepackage==0.1.0"

### 11.2.4 Testing a modified Docker image

[Read this chapter for instructions on how to modify Docker images](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/modifying-a-docker-image.html)

When you’ve added a package to the Dockerfile, you may want to check that it builds successfully before it’s added to your repository. You can include changes to your Dockerfile in a pull request which will trigger an automatic testing of building it.

[Read this chapter for more tips on how to modify Docker images](https://jhudatascience.org/Adv_Reproducibility_in_Cancer_Informatics/modifying-a-docker-image.html)

OR

If you prefer to test it locally, then you can follow these steps.

First create a GITHUB token file by making a token and copying a pasting it into a plain text file named docker/git\_token.txt. (***Make sure you do not push this to github and possibly delete it after testing your docker image build!***)

Then you’ll need to rebuild the Docker image using this command after you move into the docker directory) (But replace the <TAG\_FOR\_COURSE> with the tag for your new image including dropping the < and >:

docker build -f Dockerfile . -t <TAG\_FOR\_COURSE>

If it fails, often the issue is a missing dependency. Take a look at the error messages and see if you can determine the issue with some Googling. Also be sure that all your directories and files are named correctly.

Once it builds successfully, run the above command with the new name for your docker image make sure that the tag does not have upper case characters):

docker build -f Dockerfile . -t jhudsl/<TAG\_FOR\_COURSE>

#### 11.2.4.1 Pushing the Docker image

For any changes you make to your Docker image to take effect in your repository’s github actions and workflows, you must push your updated docker image to Dockerhub. There are two different ways you can push your Docker image.

Locally, you can push your updated image to Dockerhub using (make sure that the tag does not have upper case characters):

docker push jhudsl/<TAG\_FOR\_COURSE>

OR

You can use GitHub actions to do this by going to your course’s GitHub repository, go to Actions and then to Test build of Dockerfile.

Click on run workflow type in true underneath Push to Dockerhub?.

Then click Run. If your Dockerfile builds an image successfully it will automatically be pushed to Dockerhub.

# 12 Spell Check

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Running locally](#running-locally)
* [Ignoring URLs](#ignoring-urls)

Github actions will automatically run a spell check on all Rmds whenever a pull request to the main branch is filed. [See this section about how to customize what checks are run](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots). Depending on your preference, you may find it easier to spell check manually on your local computer before pushing to Github.

It will fail if there are more than 3 spelling errors and you’ll need to resolve those before being able to merge your pull request.

Click on the GitHub comment that has the download link to spelling errors that need to be resolved.

Some of these errors may be things that the spell check doesn’t recognize for example: ITCR or DaSL. If it’s a ‘word’ the spell check should recognize, you’ll need to add this to the dictionary.

Go to the resources/dictionary.txt file. Open the file and add the new ‘word’ to its appropriate place (the words are in alphabetical order). Then commit the changes to resources/dictionary.txt to your branch and this should make the spell check status check pass.

### 12.0.1 Running locally

You can run spell check locally by calling the script from the top of the repository (your main git repository directory) in bash:

Rscript scripts/spell-check.R

### 12.0.2 Ignoring URLs

If you want to write out a URL without having to add it to the spell check dictionary, you can style the URL as inline code by adding single back ticks, and it will be ignored by spell check:

Example:

Visit [`example.com`](https://www.example.com) now.

Visit [example.com](https://www.example.com) now.

# 13 Borrowing chapters between courses

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Borrowing from a local file](#borrowing-from-a-local-file)
* [Borrowing from a private repository](#borrowing-from-a-private-repository)
* [Removing an h1 header](#removing-an-h1-header)
* [Linking between chapters](#linking-between-chapters)

If you have two courses that the content and topics overlap, you may want to share written material between the two.

But, if you copy and paste to share material this would create a maintenance problem because if you update one you will need to remember to copy over the other! 😱

In OTTR, we try to minimize maintenance pains so to get around this, we use cow::borrow\_chapter() from the [jhudsl/cow](https://jhudatascience.org/cow/index.html) package. The cow package is already on the jhudsl/course\_template docker image so you do not need to install it if you are using the docker image or if you are have GitHub actions do all the rendering for you.

To borrow a chapter from another course, create an .Rmd as you normally would, with a ottrpal::set\_knitr\_image\_path() in a chunk at the beginning of the file and a [H1 title](https://www.markdownguide.org/basic-syntax/).

Then, wherever you would like the borrowed chapter to appear, put an R chunk with this:

The magic of this function is that whenever the course is re-rendered it will knit the latest version of the chapter you are borrowing.  
Note that this chunk cannot be run interactively, just include it in your Rmd and render your course as usual.  
  
### Borrowing from a local file  
  
If for some reason you would like a local file incorporated, just leave off the repo\_name argument and `cow::borrow\_chapter()` will look for the chapter locally:

### 13.0.1 Borrowing from a private repository

If you are borrowing from a course in a private repository, you will need to supply [GitHub personal access token](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token) using a git\_pat argument like this:

### Removing an h1 header  
  
If you want to change the title you can use an option `remove\_h1` to remove the title from the incoming borrowed chapter.  
  
  
  
  
Some new words after the borrowed chapter content.

### 13.0.2 Linking between chapters

If you don’t want the material from another chapter completely copied over, you might instead just want to put a link to the Bookdown chapter. You can just use the full url. A link would look something like this:

![](https://jhudatascience.org/OTTR\_Template/a-new-chapter.html)

# 14 Using Google Docs

# 15 Automation info and set up

## Warning in readLines(dest\_file): incomplete final line found on 'resources/  
## other\_chapters/How-to-set-up-and-customize-GitHub-actions-robots.md'

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Customizing Automation](#customizing-automation)
  + [Pull request checks:](#pull-request-checks)
    - [Check quiz formatting](#check-quiz-formatting)
    - [Check for broken URLs](#check-for-broken-urls)
    - [Preview rendering](#preview-rendering)
    - [Spell checking](#spell-checking)
    - [Style code](#style-code)
    - [Docker testing](#docker-testing)
  + [Rendering actions](#rendering-actions)
  + [Manually re-running rendering or checks](#manually-running-rendering-or-checks)
  + [Fixing broken GitHub actions](#fixing-broken-github-actions)

# 16 Customizing Automation

By default, all automation steps and checks will run. But depending on the needs of your course, you can turn these on and off by going to the config\_automation.yml file and switching options to yes or no.

The config\_automation.yml file looks like this:

# Formatting Checks (run at pull request)  
check-quizzes: yes  
url-checker: yes  
render-preview: yes  
style-and-sp-check: yes  
...

There are two main sets of automation steps and checks run: - .github/workflows/pull-request.yml : run upon opening a pull request - .github/workflows/render-all.yml: run upon changes being merged to the main branch

## 16.1 Pull request checks:

These actions are triggered upon a pull request being opened. It’s set up is in the file: .github/workflows/pull-request.yml/.

### 16.1.1 Check quiz formatting

In the config\_automation.yml file it is set by:

check-quizzes: no

By default, it is set to no. But if you wish to [create quizzes on Leanpub](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Leanpub), you should set this to yes. This is not necessary if you only want quizzes for Coursera. Leanpub needs a particular format for it to upload correctly. This action will look for quizzes in a quizzes directory and check for these items. The outcome of these quiz checks will be printed to a GitHub comment on your pull request.

### 16.1.2 Check for broken URLs

In the config\_automation.yml file it is set by:

url-checker: yes

GitHub actions runs a check on all the URLs upon creating a pull request to the main branch. If it fails, you’ll need to go the Actions tab of this repository, then find the GitHub check\_urls job for the last commit you just pushed. Click on check\_urls and the specific step of Check URLs to see a print out of the URLs tested.

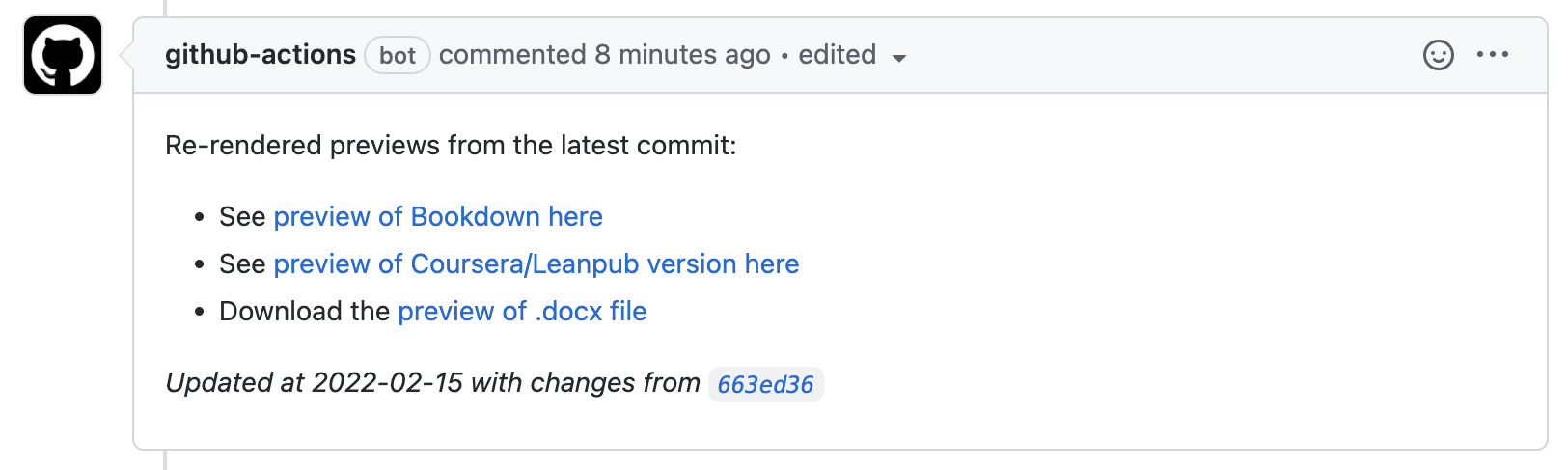
If the URL checker is failing on something that isn’t really a URL or doesn’t need to be checked, open up the pull-request.yml, scroll down to the url-check step and add the imposter URL on to the exclude-urls: argument with a comma in between.

### 16.1.3 Preview rendering

In the config\_automation.yml file it is set by:

render-preview: yes

After you open a pull request, a preview of the renders will be linked in a automatic comment on the pull request. Upon each commit these previews will re-render and edit the comment with the latest render.



These Github Actions are located in render-preview section of the pull-request.yml. These previews do NOT incorporate any changes influenced by any changes to the Docker image if the Dockerfile is also updated in the same pull request.

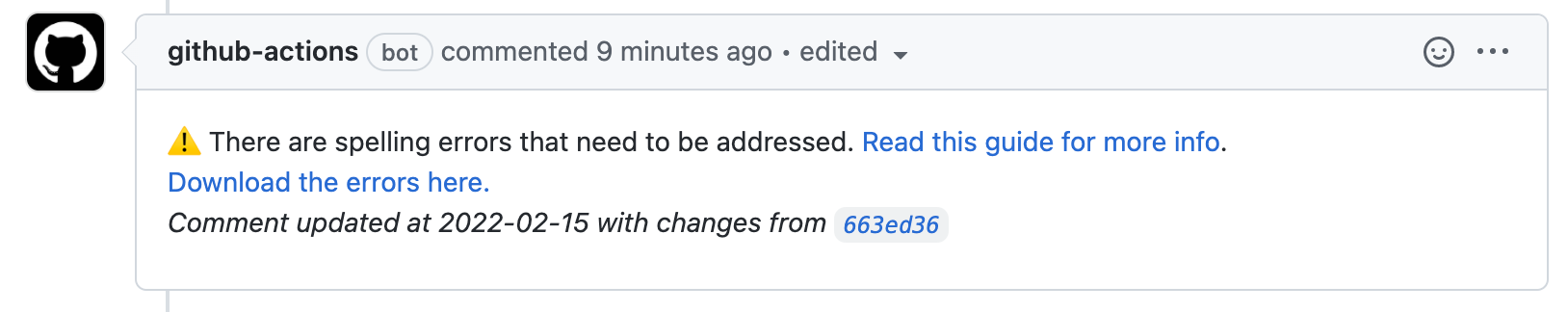
### 16.1.4 Spell checking

In the config\_automation.yml file it is set by:

spell-check: yes

Github actions will automatically [run a spell check on all Rmd’s and md’s](https://github.com/jhudsl/OTTR_Template/blob/main/.github/workflows/style-and-sp-check.yml) whenever a pull request to the main branch is filed.

It will fail if there are more than 2 spelling errors and you’ll need to resolve those before being able to merge your pull request. Errors will be printed out on a GitHub comment on your pull request.



To resolve those spelling errors, click on the link with the errors on the autogenerated comment. Some of these errors may be things that the spell check doesn’t recognize for example: ITCR or DaSL. If it’s a ‘word’ the spell check should recognize, you’ll need to add this to the dictionary.

Go to the resources/dictionary.txt file. Open the file and add the new ‘word’ to its appropriate place (the words are in alphabetical order). Then commit the changes to resources/dictionary.txt to your branch and this should make the spell check status check pass.

### 16.1.5 Style code

In the config\_automation.yml file it is set by:

style-code: yes

The styler package to all style R in all Rmds. Style changes will automatically be committed back to your branch.

### 16.1.6 Docker testing

In the config\_automation.yml file it looks like:

docker-test: no

By default it is set to no which means it won’t run automatically unless you change this to yes. This is only relevant if you have [your own Docker image you are managing for your course](https://github.com/jhudsl/OTTR_Template/wiki/Using-Docker#starting-a-new-docker-image). If changes are made to Docker-relevant files: Dockerfile, install\_github.R, or github\_package\_list.tsv, this will test re-build the Docker image. If it is successfully built, then it makes sense to merge it to main but the docker image will not be pushed to Dockerhub automatically. [Follow these instructions to push your Docker image to Dockerhub](https://github.com/jhudsl/OTTR_Template/wiki/Using-Docker#pushing-the-docker-image).

## 16.2 Rendering actions

Upon merging changes to any Rmd or assets/ folder to main, the course material will be automatically re-rendered. By default, all rendering steps will be run. But depending on the needs of your course, you can turn these on and off by going to the config\_automation.yml file and switching options to yes or no.

render-bookdown: yes  
render-leanpub: yes  
render-coursera: yes

For publishing to Leanpub, make sure that the render Leanpub option is set to yes: render-leanpub: yes. [See more details about publishing to Leanpub, here](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Leanpub).

For publishing to Coursera, make sure render Coursera option is set to yes: render-coursera: yes. [See more details about publishing to Coursera, here](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Coursera).

## 16.3 Manually running rendering or checks

From time to time, it may be useful to manually re-trigger a particular GitHub action. Most of the GitHub Actions, particularly the rendering ones, can be re-run manually. See this article about [how to manually re-run a GitHub Action](https://docs.github.com/en/actions/managing-workflow-runs/manually-running-a-workflow#running-a-workflow).

## 16.4 Fixing broken GitHub Actions

GitHub action rendering or other GitHub actions may fail sometimes if the input is unexpected or for a number of other reasons. To investigate why a GitHub action has failed, go to Actions and click on the failed action. [See this article for how to find this information](https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions#viewing-the-workflows-activity).

If you are unsure what the error message means and have trouble addressing it, please [file an issue on the OTTR\_Template repository to get help](https://github.com/jhudsl/OTTR_Template/issues/new?assignees=cansavvy&labels=bug&template=course-template-problem-report.md).

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

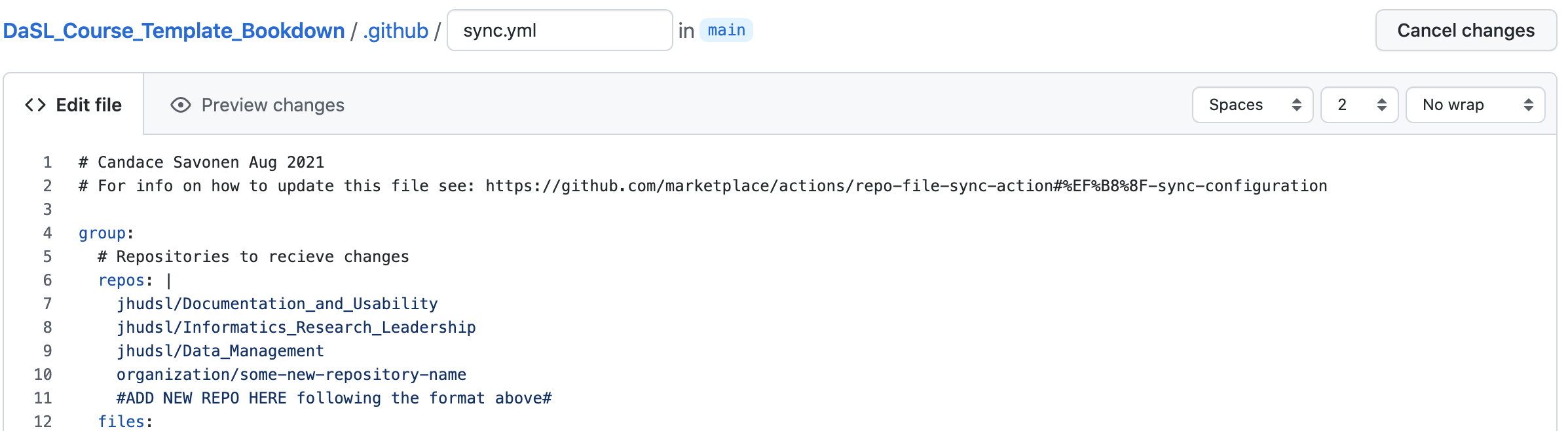
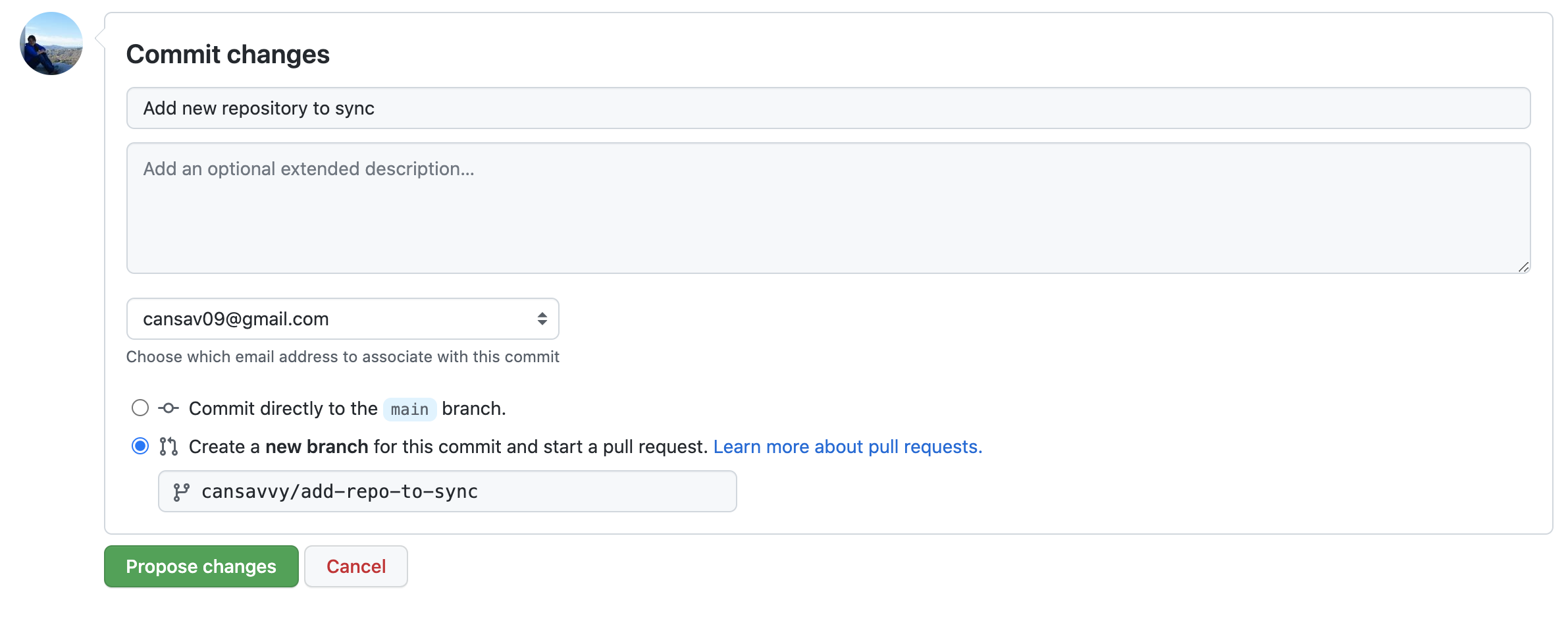
* [How to enroll a repository to receive template updates](#Xb71da84cae5fe3c7933d4e7c42ae6869fbabf8a)
* [How to tailor incoming sync changes:](#how-to-tailor-incoming-sync-changes)

This original course template: <https://github.com/jhudsl/OTTR_Template> is always a work in progress. We are working on adding more features and smoothing out bugs as we go (this is also why [your feedback is greatly appreciated](https://github.com/jhudsl/OTTR_Template/issues/new/choose)).

When updates are made to files that aren’t specific to the course content but instead run checks and other processes in the original repository, pull requests are filed automatically to any downstream repositories made from this template.

### 16.4.1 How to enroll a repository to receive template updates

**To enroll in these automatic update PRs, the new course’s repository name will need to be added to** [**this file in the original template**](https://github.com/jhudsl/OTTR_Template/edit/main/.github/sync.yml)

* Use [this link to edit the file](https://github.com/jhudsl/OTTR_Template/edit/main/.github/sync.yml).
* Add your repository’s name where it says #NEW REPO HERE#, being careful to indent the same amount as the other repositories listed. 
* Choose Create a new branch for this commit and start a pull request. Name the branch what you like. 
* Name the commit Add new repository to sync
* Click Propose changes.
* Write a short pull request description describing that you are adding a new repository to the sync.yml file.
* Click Create Pull Request
* Request @cansavvy as a reviewer.

If the your new course doesn’t need some of the functionality of these files or you find the automatic bothersome, you can feel free to use [this guide](https://github.com/marketplace/actions/repo-file-sync-action#sync-the-same-files-to-multiple-repositories) to tailor which files you want updates for. *If you have any questions about the implications of any of these updates or files, please tag @cansavvy.*

### 16.4.2 How to tailor incoming sync changes:

Upon a new release, a pull request is automatically filed to any repositories that are added to the [sync.yml file](https://github.com/jhudsl/OTTR_Template/blob/main/.github/sync.yml). **Carefully review the latest** [**release notes**](https://github.com/jhudsl/OTTR_Template/wiki/Release-Notes) **and the Files changed tab of the sync PR.** And again, tag @cansavvy with any questions.

It might be that you want some of the changes from the PR but not all of them. Each file that is updated in a sync PR is updated in its own commit.

* If you don’t want the changes from a particular file, you can always [revert that particular commit](https://git-scm.com/docs/git-revert) before merging the sync PR.  
  If you will not want any updates on this file in the future, you may want to remove a file from being synced in your repo [by reconfiguring the sync file](https://github.com/jhudsl/OTTR_Template/blob/main/.github/sync.yml).
* If you want only some changes, but they are not on a whole file basis, you could check out the branch and make manual edits. To checkout the branch, navigate to your own repository you should be able to run:

git checkout repo-sync/OTTR\_Template/default

* If you don’t want any of the changes you can close the PR entirely. You may want to unenroll your repository from the [sync github actions by deleting your repo name from this file](https://github.com/jhudsl/OTTR_Template/blob/main/.github/sync.yml) if this will continue to be the case.

# 17 Finishing touches

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Set up feedback method](#set-up-feedback-method)
  + [Step 1) Decide on a method of feedback for your users to use.](#X1a0087ae8bb0367103743de58ee49c1e8da7320)
  + [Step 2) Add feedback button](#step-2-add-feedback-button)
  + [Step 3) Add your updated \_output.yml to a pull request.](#Xad2b4e2f97ba4031792feb86459ed5b6789cf76)

## 17.1 Set up feedback method

Before you release your course to the public, you will want to set up a method to allow users to give you feedback about the course. This is useful so users can allow you to know if anything is broken or if they have an idea for improvements!

We will explain one way of adding a feedback method to your course, but depending on your course’s needs and set up you may want to adjust to it or add to it.

### 17.1.1 Step 1) Decide on a method of feedback for your users to use.

Here’s some ideas: - Link the to the GitHub issues (you may want to use an issue template to help users structure their ideas). - Link to a Google form that you’ll monitor. Here’s a [template Google form](https://docs.google.com/forms/d/13jFBLjZaYmIZM02OfuF5_yQ-ccHdNZg5XE518LeSpfg/edit) you can make a copy of to get started.

### 17.1.2 Step 2) Add feedback button

Add the line below to the last part of your \_output.yml file but fill in the method of feedback URL from Step 1 where it says{FEEDBACK-URL-HERE} including removing the {}:

<p style="padding-left: 40px;"><div class="trapezoid" style = "padding-left: 40px;"><span> <a href="{FEEDBACK-URL-HERE}"> Click here to provide feedback</a> <img src="assets/itcr\_arrow.png" style=" width: 10%" ></span></div></p>

### 17.1.3 Step 3) Add your updated \_output.yml to a pull request.

Check the preview link and make sure it’s what you want. Add to the main branch when you and your team have deemed it is ready.

\*For more ideas and discussion on obtaining user feedback see [this course chapter](https://jhudatascience.org/Documentation_and_Usability/obtaining-user-feedback.html).

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Changing title](#changing-title)
* [Customizing the Style](#customizing-the-style)
  + [Using a style set](#using-a-style-set)
* [Creating your own style](#creating-your-own-style)
  + [Changing the favicon](#changing-the-favicon)
  + [Adding logos](#adding-logos)
* [Adding sections that aren’t numbered](#adding-sections-that-arent-numbered)
* [Changing course text colors](#changing-course-text-colors)
* [Modifying the image at the top of the course](#Xe6a05fb8184fd1e74bf541be112ba96ebcd8a49)

### 17.1.4 Changing title

The title is specified on the index.Rmd page in the yml header. Modify the yaml header to change the title for your course.

---  
title: "Title of Course"  
---

Note that if one of the chapter Rmd files has a title in the yml that comes first alphabetically, it will be shown as the title of the course. Hence avoid having yml header titles for the chapter files.

## 17.2 Customizing the Style

There are styles/brandings that are available in our library of style sets. However, there are also instructions to customize your own course style following these instructions in the next section.

### 17.2.1 Using a style set

By default this course template will use the jhudsl data science lab style. However, you can switch this to another style set by moving some files. Take a look at the style-sets for the other styles available.

For example, if you are creating an ITCR course, you will need the files in style-sets/itcr or if you are making a DataTrail course, the files in style-sets/data-trail. For these instructions,let’s refer to data-trail or itcr as the <set-name>.

1. On a new branch, copy the style-sets/<set-name>/index.Rmd and style-sets/<set-name>/\_output.yml to the top of the repository to overwrite the default index.Rmd and \_output.yml.
2. Copy over all the files in the style-sets/<set-name>/copy-to-assets to the assets folder in the top of the repository.
3. [Create a pull request](https://github.com/jhudsl/OTTR_Template/wiki/About-pull-request-review) with these changes, and double check the rendered preview to make sure that the style is what you are looking for.

## 17.3 Creating your own style

Here are the instructions to change the aesthetic aspects about your course if you wish to create a new style for your course.

### 17.3.1 Changing the favicon

Favicons are small icons that appear on your browser tab. To change the favicon, first take the image you would like to use to this [website](https://favicon.io/favicon-converter/) to convert it into a favicon. Then save this file in the assets/ directory. On the index.Rmd file, make sure that the correct favicon is referenced to in the yaml header, so that the correct favicon will be used.

Here you can see that by default the Data Science Lab (dasl) favicon will be used.

---  
title: "Course Name "  
date: "March, 2022"  
site: bookdown::bookdown\_site  
documentclass: book  
bibliography: [book.bib, packages.bib]  
biblio-style: apalike  
link-citations: yes  
description: "Description about Course/Book."  
favicon: assets/dasl\_favicon.ico  
---

If you are making an [ITN](https://www.itcrtraining.org/) course, then the favicon is already set up n the index-itcr.Rmd file. Just delete the existing index.Rmd file and rename the index-itcr.Rmd file to be index.Rmd. This is already part of the set up instructions.

### 17.3.2 Adding logos

Logos for the table of contents are added with the \_output.yml file. This adds an image above the table of contents when the content is rendered with bookdown.

If you are creating a general DaSL course: - Please replace the URL in the line 13 of code for the \_output.yml file with the URL for the GitHub repo for your course. This will allow people to more easily find how out how you created your course. Otherwise, they will be directed to this template.

If you are creating a DaSL course for a project other than [ITN](https://www.itcrtraining.org/): - Delete the \_output.yml file and rename the \_output-itcr.yml to be \_output.yml.  
- Please modify the lines that link to the <http://jhudatascience.org/> with your own website and logo if you are not part of the [jhuDaSL](http://jhudatascience.org/) . - Please replace the URL in the line 13 of code with the URL for the GitHub repo for your course. This will allow people to more easily find how out how you created your course. Otherwise, they will be directed to this template. - If you wish to create a different color scheme, the font colors can also be modified along with other aspects in the assets/style.css file. Take a look at the assets/style\_ITN.css file to see what was changed for that color scheme from the assets/style.css file. - You can replace the logo with the appropriate project logo by replacing https://www.itcrtraining.org/ with the project website link and "https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/images/logo.png" for the project logo image link in line 11.

## 17.4 Adding sections that aren’t numbered

You may notice that currently the References page and about pages are not numbered like the other chapters. If you want additional sections like this add an Rmd file and type the name of the page after a single hashtag # followed by this: {-}. This will exclude this page from being numbered.

Thus as example the reference page looks like this:

# References {-}

## 17.5 Changing course text colors

To modify the colors used for the text, take a look at the assets/style.css code.

If you would like to change the current dark blue color to be a different color, search and replace for #012d72. You can find hex color codes at this [website](https://htmlcolorcodes.com/color-picker/) to use as a replacement.

If you would like to change the current light blue color to be a different color, do the same but search for #68ace5.

## 17.6 Modifying the image at the top of the course

If you would like to change the image at the top of the Bookdown version of the course, you need to do the following steps: \* Add a new image file to the assets directory \* Modify the assets/big-image.html file on line 11. Change out src = "assets/dasl\_thin\_main\_image.png" so that dasl\_thin\_main\_image.png is replaced with the name of your image file.

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Adding Credits](#adding-credits)
  + [Adding the Credits table to Coursera](#adding-the-credits-table-to-coursera)
  + [Adding the Credits table to Leanpub](#adding-the-credits-table-to-leanpub)

## 17.7 Adding Credits

Since so many individuals contribute to our courses in so many ways. Thus, we decided to make a table of credits to make it clear who did what.

Here is an example:

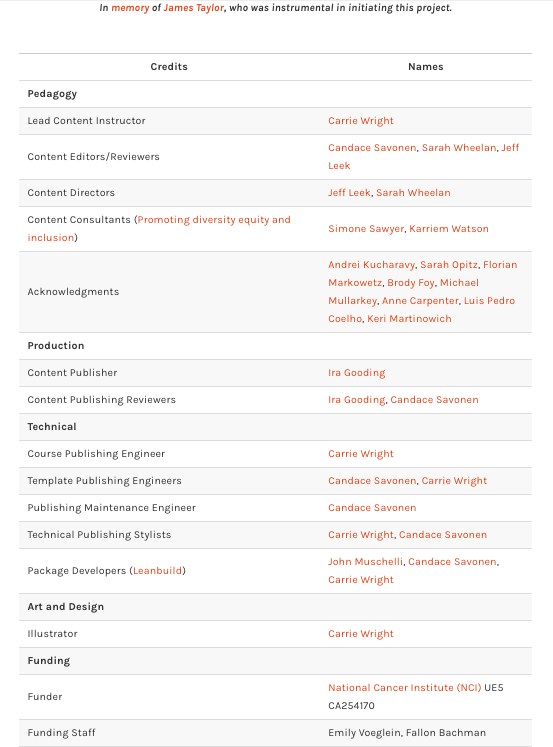


Figure : Credit section from a course that used this template

Please see the [about section](https://jhudatascience.org/OTTR_Template/about-the-authors.html) of the template for descriptions of how credits should be attributed for the course.

This is generated from the About.Rmd file. An About.Rmd file will already be in your course from using our template. You will need to do the following steps to update it for your course:

1. Fill out author names for the various roles where applicable.
2. Delete lines for roles that are not applicable.
3. Make sure roles grammatically match names. If there are multiple people include the “s” in the role name where applicable and remove the parentheses. If only one person is appropriate for a given role remove “(s)” from the role name.
4. Ensure that each row has “|” around each cell value.

*Example Table*:

The first row and additional rows based on [this table](https://bit.ly/course-credits-table) should be added and filled in using [markdown table format](https://www.markdownguide.org/extended-syntax/).

|Credits|Names|  
|-------|-----|  
|Lead Content Instructor|FirstName LastName|

Links to personal websites can be added below the table to make it easier to read.

|Credits|Names|  
|-------|-----|  
|Lead Content Instructor|[FirstName LastName]|  
  
<!-- Author information -->  
  
[FirstName LastName]: link to personal website

These rows should be included for *all courses*:

|Template Publishing Engineers|[Candace Savonen], [Carrie Wright]|  
|Publishing Maintenance Engineer|[Candace Savonen]|  
|Technical Publishing Stylists|[Carrie Wright], [Candace Savonen]|  
|Package Developers[ottrpal]|[John Muschelli], [Candace Savonen], [Carrie Wright]|  
  
<!-- Author information -->  
  
[John Muschelli]: https://johnmuschelli.com/  
[Candace Savonen]: https://www.cansavvy.com/  
[Carrie Wright]: https://carriewright11.github.io/  
  
<!-- Links -->  
  
[ottrpal]: https://github.com/jhudsl/ottrpal

These rows should be added to all *Johns Hopkins* courses:

|Content Publisher|[Ira Gooding]|  
|Content Publishing Reviewer|Ira Gooding]|  
  
<!-- Author information -->  
  
[Ira Gooding]: https://publichealth.jhu.edu/faculty/4130/ira-gooding

Additionally, this row should be added for *all ITCR Courses*:

|Content Directors|[Jeff Leek], [Sarah Wheelan]|  
  
<!-- Author information -->  
  
[Jeff Leek]: https://jtleek.com/  
[Sarah Wheelan]: https://www.hopkinsmedicine.org/profiles/details/sarah-wheelan

### 17.7.1 Adding the Credits table to Coursera

In Coursera, you can add the credits table URL as an ungraded plugin ([the same as described here for adding chapter content](https://github.com/jhudsl/OTTR_Template/wiki/Publishing-on-Coursera#navigating-to-your-course-on-coursera)). This should be added at the beginning of your course, right after the introduction.

### 17.7.2 Adding the Credits table to Leanpub

In Leanpub, make sure that your About.md file in your manuscript folder is listed in your Book.txt file and this Credits table will automatically be incorporated into your Leanpub course.

If you would like to add Google Analytics to track traffic to your course, you can do the following:

1. Get a **Google Analytics account** (if you do not already have one): <https://analytics.google.com/analytics>

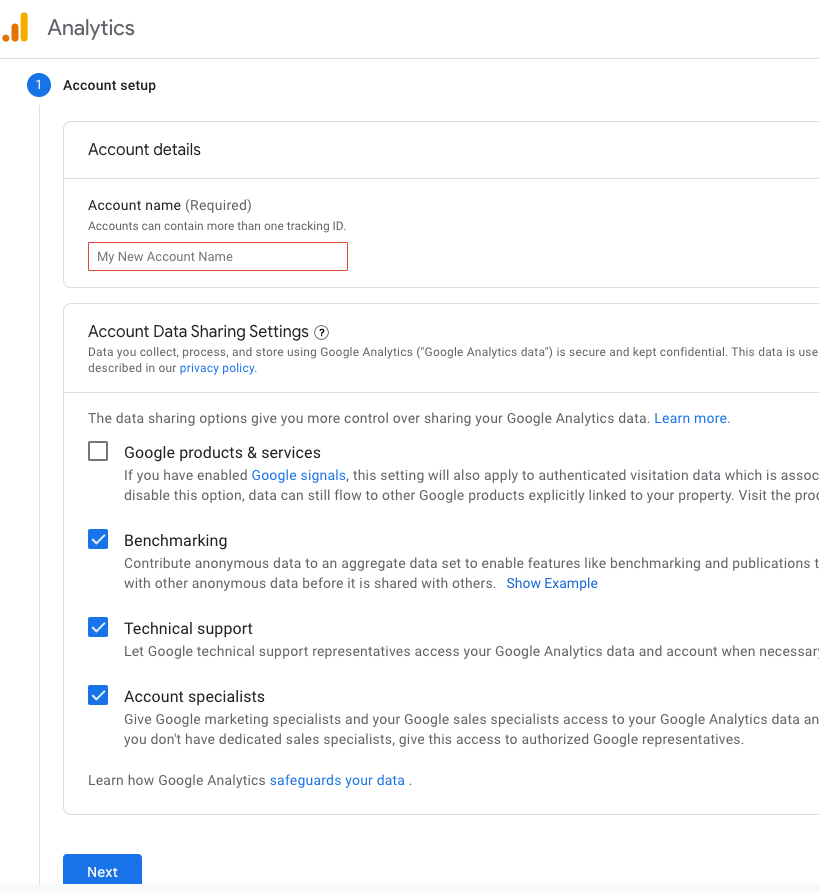


Figure : Google Analytics account setup

Note that in creating an account you will need to agree to some terms. Currently it is free to get Google analytics data as long as your course does not exceed a very high user rate. Check to make sure that the terms work for you

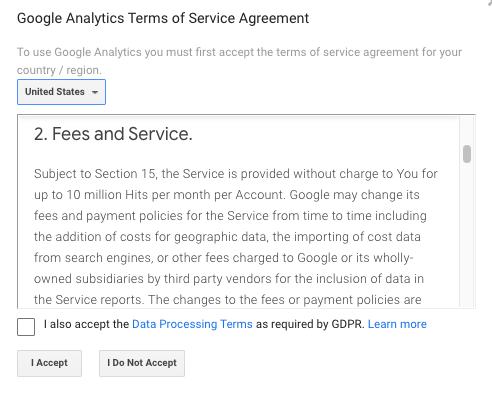


Figure : Google Analytics terms

1. If you already have an account or navigated away from where you started - Go to the **Admin tab** (lower left button that looks like a gear)
2. Set up a new **property** (fill out name and details, select options about tracking traffic for how you intend to use Google Analytics)

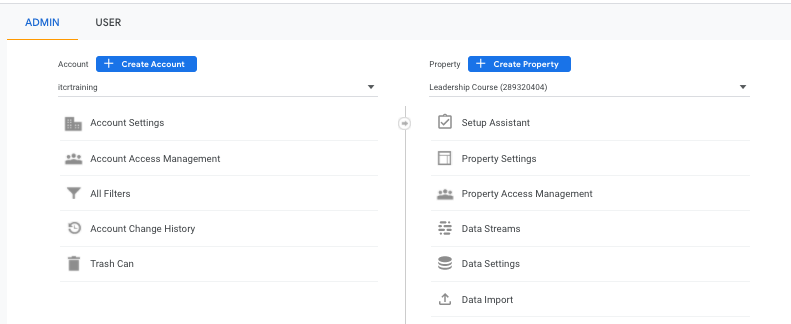


Figure : Google Analytics Property creation

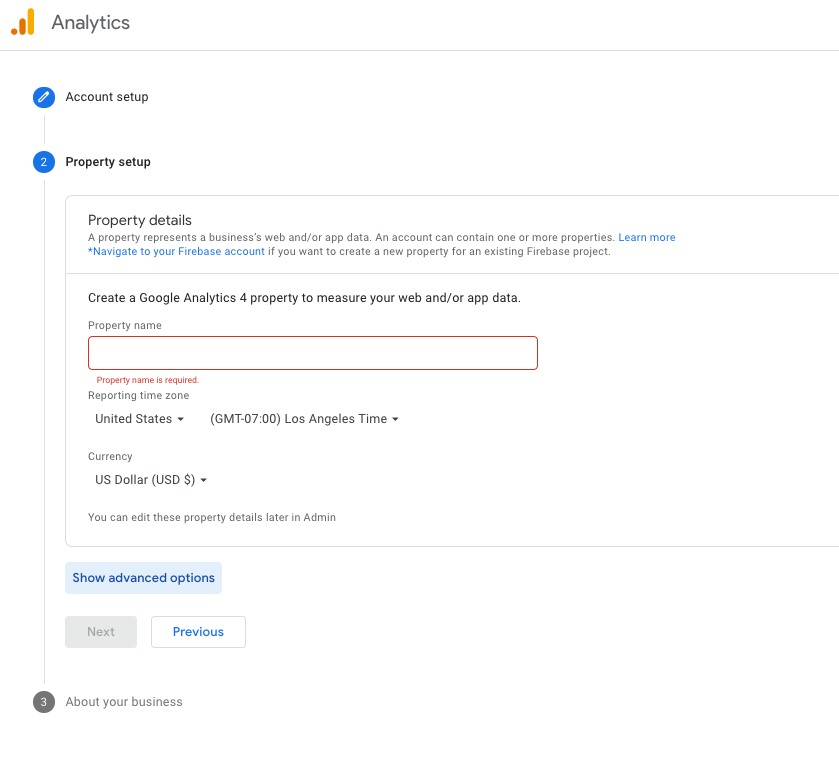


Figure : Google Analytics Property

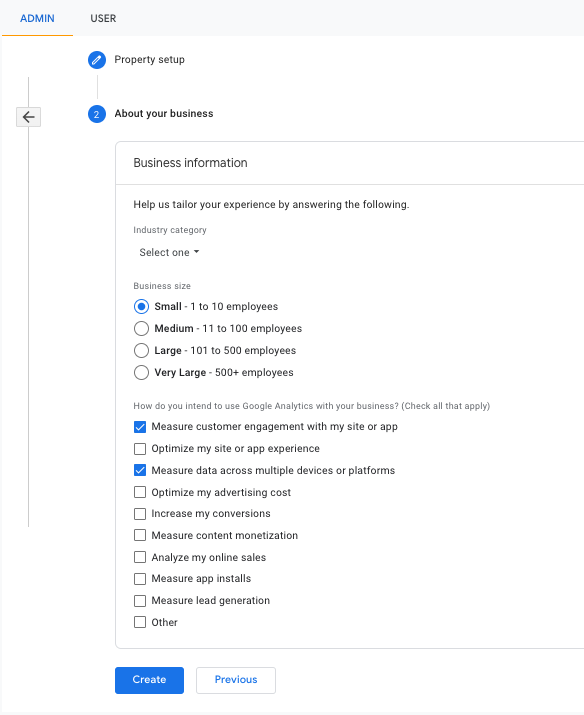


Figure : Google Analytics Property choices

1. Add a **stream** to your property, choose the **Web** option.

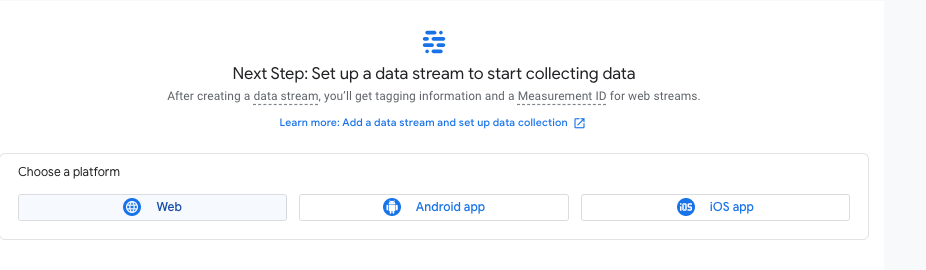


Figure : Google Analytics stream options

1. Fill in your stream information with the **link** for your course (note you may need to remove https as this is part of a pull down menu to the left of where you paste your link)

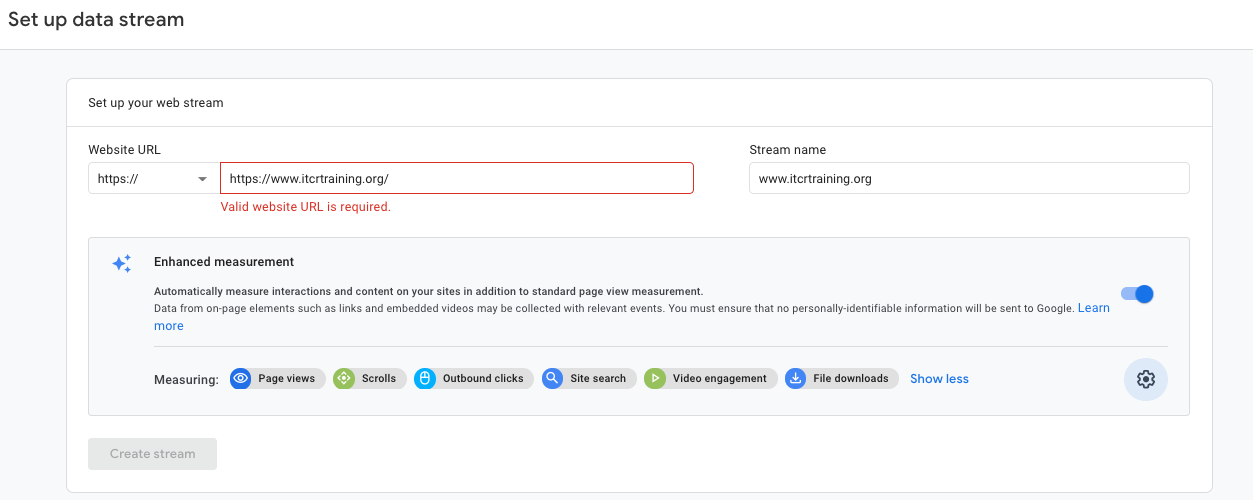


Figure : Google Analytics stream

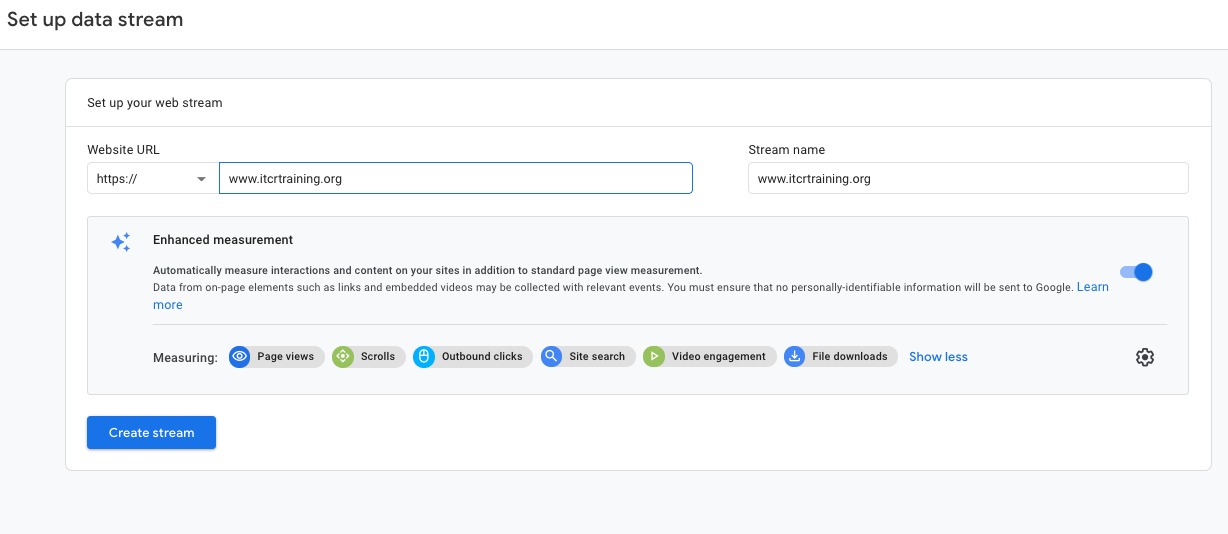
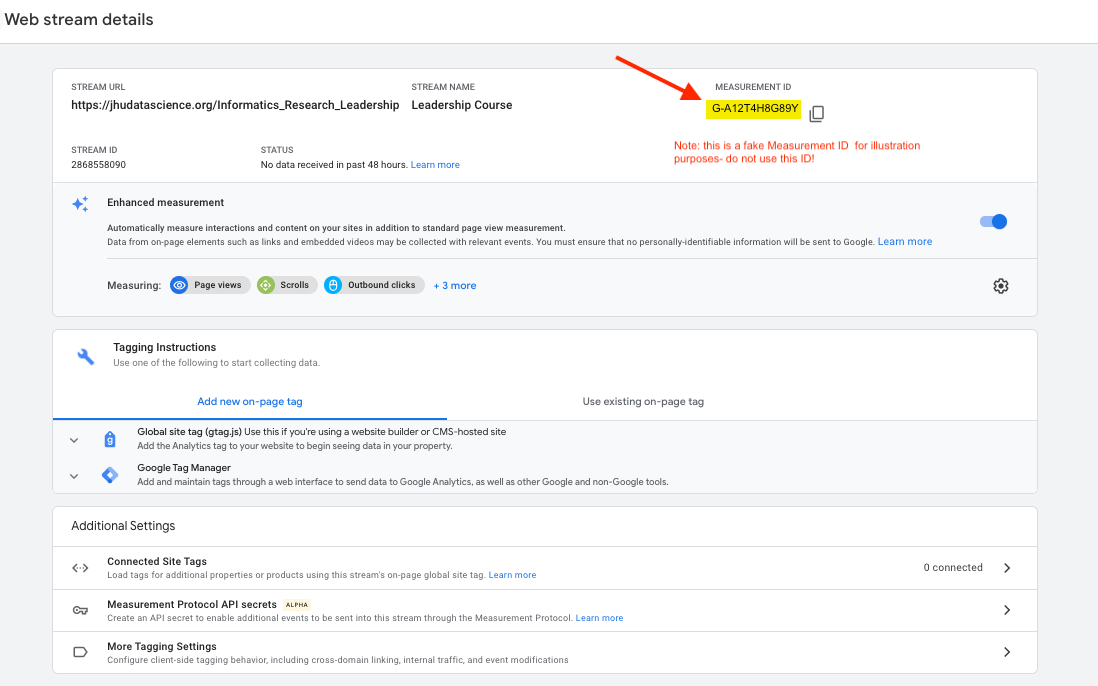
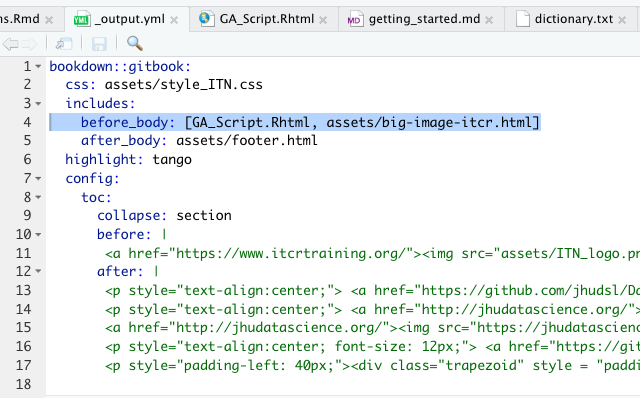


Figure : Google Analytics stream

1. On the resulting page you will see a **Measurement ID**. Copy this ID and paste it in the GA\_Script.Rhtml file in the template replacing the fields that say {MeasurementID} including the curly brackets.



1. Modify the \_output.yml file so that the line that starts with before body: is changed to include brackets around the GA\_script.Rhtml file before a comma and the name of the hero image html file (which should already be listed). For example, like this: before\_body: [GA\_Script.Rhtml, assets/big-image-itcr.html] or this: before\_body: [GA\_Script.Rhtml, assets/big-image.html]



1. **Rerender** your course by making a change to one of your chapter RMD files in a pull request and pushing and merging the pull request. This will cause new html files to be made for each chapter in the docs folder. The Google Analytics code should now be in each of the html files - you can check by searching for Google Analytics.
2. Go back to [Google Analytics](https://analytics.google.com/analytics) and log in if you need to.
3. Check on the **Reports** button on the top of the far left icon menu. The icon looks like a bar chart. If you open up a browser window to your hosted course online, then you should see yourself as 1 user in the last 30 minutes.

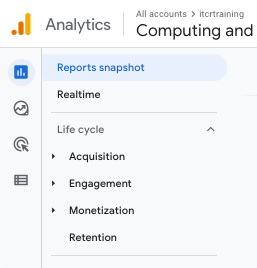


Figure : Google Analytics Reports

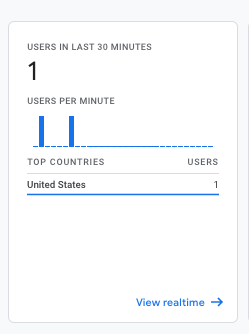


Figure : Check Google Analytics

1. Enjoy the data about how people are using your course!

**Table of Contents** *generated with* [*DocToc*](https://github.com/thlorenz/doctoc)

* [Release coding](#release-coding)

Congrats, you are most of the way toward publishing your course!

At this point, it is a good idea to [create a release](https://docs.github.com/en/repositories/releasing-projects-on-github/managing-releases-in-a-repository)! (It’s a good idea to [set up Google Analytics](https://github.com/jhudsl/OTTR_Template/wiki/Google-Analytics) before completing this step)

For jhudsl courses your release tag signifies your course’s readiness for public viewing (and whether or not it will be added to the public version of the [JHU course library](https://docs.google.com/spreadsheets/d/14KYZA2K3J78mHVCiWV6-vkY6it37Ndxnow1Uu7nMa80/edit#gid=0).

### 17.7.3 Release coding

* Releases that are 1.0 or greater are considered ready for public viewing.
* Releases less than 1.0 will be considered “on deck”. They will be advertised to learners as nearly ready, but will have a warning that lets them know the course is still actively under development.
* No release tags means the course is not ready for public listing or viewing at this time.

# About the Authors

These credits are based on our [course contributors table guidelines](https://github.com/jhudsl/OTTR_Template/wiki/How-to-give-credits).

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Lead Content Instructor(s) | [FirstName LastName](link%20to%20personal%20website) |
| Lecturer(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | Delivered the course in some way - video or audio |
| Content Author(s) (include chapter name/link in parentheses if only for specific chapters) - make new line if more than one chapter involved | If any other authors besides lead instructor |
| Content Contributor(s) (include section name/link in parentheses) - make new line if more than one section involved | Wrote less than a chapter |
| Content Editor(s)/Reviewer(s) | Checked your content |
| Content Director(s) | Helped guide the content direction |
| Content Consultants (include chapter name/link in parentheses or word “General”) - make new line if more than one chapter involved | Gave high level advice on content |
| Acknowledgments | Gave small assistance to content but not to the level of consulting |
| **Production** |  |
| Content Publisher(s) | Helped with publishing platform |
| Content Publishing Reviewer(s) | Reviewed overall content and aesthetics on publishing platform |
| **Technical** |  |
| Course Publishing Engineer(s) | Helped with the code for the technical aspects related to the specific course generation |
| Template Publishing Engineers | [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/) |
| Publishing Maintenance Engineer | [Candace Savonen](https://www.cansavvy.com/) |
| Technical Publishing Stylists | [Carrie Wright](https://carriewright11.github.io/), [Candace Savonen](https://www.cansavvy.com/) |
| Package Developers ([ottrpal](https://github.com/jhudsl/ottrpal)) | [John Muschelli](https://johnmuschelli.com/), [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/) |
| **Art and Design** |  |
| Illustrator(s) | Created graphics for the course |
| Figure Artist(s) | Created figures/plots for course |
| Videographer(s) | Filmed videos |
| Videography Editor(s) | Edited film |
| Audiographer(s) | Recorded audio |
| Audiography Editor(s) | Edited audio recordings |
| **Funding** |  |
| Funder(s) | Institution/individual who funded course including grant number |
| Funding Staff | Staff members who help with funding |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value   
## version R version 4.0.2 (2020-06-22)  
## os Ubuntu 20.04.3 LTS   
## system x86\_64, linux-gnu   
## ui X11   
## language (EN)   
## collate en\_US.UTF-8   
## ctype en\_US.UTF-8   
## tz Etc/UTC   
## date 2022-03-03   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] RSPM (R 4.0.3)   
## bookdown 0.24 2022-02-15 [1] Github (rstudio/bookdown@88bc4ea)   
## callr 3.4.4 2020-09-07 [1] RSPM (R 4.0.2)   
## cli 2.0.2 2020-02-28 [1] RSPM (R 4.0.0)   
## crayon 1.3.4 2017-09-16 [1] RSPM (R 4.0.0)   
## desc 1.2.0 2018-05-01 [1] RSPM (R 4.0.3)   
## devtools 2.3.2 2020-09-18 [1] RSPM (R 4.0.3)   
## digest 0.6.25 2020-02-23 [1] RSPM (R 4.0.0)   
## ellipsis 0.3.1 2020-05-15 [1] RSPM (R 4.0.3)   
## evaluate 0.14 2019-05-28 [1] RSPM (R 4.0.3)   
## fansi 0.4.1 2020-01-08 [1] RSPM (R 4.0.0)   
## fs 1.5.0 2020-07-31 [1] RSPM (R 4.0.3)   
## glue 1.6.1 2022-01-22 [1] CRAN (R 4.0.2)   
## htmltools 0.5.0 2020-06-16 [1] RSPM (R 4.0.1)   
## knitr 1.33 2022-02-15 [1] Github (yihui/knitr@a1052d1)   
## lifecycle 1.0.0 2021-02-15 [1] CRAN (R 4.0.2)   
## magrittr 2.0.2 2022-01-26 [1] CRAN (R 4.0.2)   
## memoise 1.1.0 2017-04-21 [1] RSPM (R 4.0.0)   
## pkgbuild 1.1.0 2020-07-13 [1] RSPM (R 4.0.2)   
## pkgload 1.1.0 2020-05-29 [1] RSPM (R 4.0.3)   
## prettyunits 1.1.1 2020-01-24 [1] RSPM (R 4.0.3)   
## processx 3.4.4 2020-09-03 [1] RSPM (R 4.0.2)   
## ps 1.3.4 2020-08-11 [1] RSPM (R 4.0.2)   
## purrr 0.3.4 2020-04-17 [1] RSPM (R 4.0.3)   
## R6 2.4.1 2019-11-12 [1] RSPM (R 4.0.0)   
## remotes 2.2.0 2020-07-21 [1] RSPM (R 4.0.3)   
## rlang 0.4.10 2022-02-15 [1] Github (r-lib/rlang@f0c9be5)   
## rmarkdown 2.10 2022-02-15 [1] Github (rstudio/rmarkdown@02d3c25)  
## rprojroot 2.0.2 2020-11-15 [1] CRAN (R 4.0.2)   
## sessioninfo 1.1.1 2018-11-05 [1] RSPM (R 4.0.3)   
## stringi 1.5.3 2020-09-09 [1] RSPM (R 4.0.3)   
## stringr 1.4.0 2019-02-10 [1] RSPM (R 4.0.3)   
## testthat 3.0.1 2022-02-15 [1] Github (R-lib/testthat@e99155a)   
## usethis 2.1.5.9000 2022-02-15 [1] Github (r-lib/usethis@57b109a)   
## withr 2.3.0 2020-09-22 [1] RSPM (R 4.0.2)   
## xfun 0.26 2022-02-15 [1] Github (yihui/xfun@74c2a66)   
## yaml 2.2.1 2020-02-01 [1] RSPM (R 4.0.3)   
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library

# References