Title of Course

December, 2021

Table of Contents

# About this Course

# 1 Getting started

# 2 Getting Started

* 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.

### 2.0.1 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).

## 2.1 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)

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

## Could not find git credentials, please set by running usethis::create\_github\_token(),  
## or directly providing a personal access token using the git\_pat argument

## https://raw.githubusercontent.com//jhudsl/OTTR\_Template/main/docs/02-chapter\_of\_course.md

##   
##   
## processing file: ./resources/other\_chapters/02-chapter\_of\_course.md

|  
| | 0% |  
|…………………………………………………………….| 100% ordinary text without R code

\*If you haven’t yet read the getting started Wiki pages; [start there](https://github.com/jhudsl/OTTR_Template/wiki/Getting-started)

Every chapter needs to start out with this chunk of code:

## 2.3 Learning Objectives

\*Every chapter also needs Learning objectives that will look like this:

This chapter will cover:

* {You can use <https://tips.uark.edu/using-blooms-taxonomy/> to define some learning objectives here}
* {Another learning objective}

## 2.4 Libraries

For this chapter, we’ll need the following packages attached:

\*Remember to add [any additional packages you need to your course’s own docker image](https://github.com/jhudsl/OTTR_Template/wiki/Using-Docker#starting-a-new-docker-image).

library(magrittr)

# 3 Topic of Section

You can write all your text in sections like this!

## 3.1 Subtopic

Here’s a subheading and some text in this subsection!

### 3.1.1 Code examples

You can demonstrate code like this:

output\_dir <- file.path("resources", "code\_output")  
if (!dir.exists(output\_dir)) {  
 dir.create(output\_dir)  
}

And make plots too:

hist\_plot <- hist(iris$Sepal.Length)

You can also save these plots to file:

png(file.path(output\_dir, "test\_plot.png"))  
hist\_plot

## $breaks  
## [1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0  
##   
## $counts  
## [1] 5 27 27 30 31 18 6 6  
##   
## $density  
## [1] 0.06666667 0.36000000 0.36000000 0.40000000 0.41333333 0.24000000 0.08000000  
## [8] 0.08000000  
##   
## $mids  
## [1] 4.25 4.75 5.25 5.75 6.25 6.75 7.25 7.75  
##   
## $xname  
## [1] "iris$Sepal.Length"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

dev.off()

## png   
## 2

### 3.1.2 Image example

How to include a Google slide. It’s simplest to use the leanbuild package:

But if you have the slide or some other image locally downloaded you can also use html like this:

### 3.1.3 Video examples

You can use knitr::include\_url() like this:

knitr::include\_url("https://www.youtube.com/embed/VOCYL-FNbr0")

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

OR this works:

### 3.1.4 Links to files

This works:

Or this:

[This works](https://www.messiah.edu/download/downloads/id/921/Microaggressions_in_the_Classroom.pdf).

Or this:

### 3.1.5 Links to websites

Examples of including a website link.

This works:

knitr::include\_url("https://yihui.org")

OR this:

![Another link](data:text/html; charset=utf-8;base64,)

Figure : Another link

OR this:

### 3.1.6 Citation examples

We can put citations at the end of a sentence like this ([Allaire et al. 2021](#ref-rmarkdown2021)). Or multiple citations Xie, Allaire, and Grolemund ([2018](#ref-Xie2018)).

but they need a ; separator ([Allaire et al. 2021](#ref-rmarkdown2021); [Xie, Allaire, and Grolemund 2018](#ref-Xie2018)).

In text, we can put citations like this Allaire et al. ([2021](#ref-rmarkdown2021)).

## 3.2 Print out session info

sessionInfo()

## R version 4.0.2 (2020-06-22)  
## Platform: x86\_64-pc-linux-gnu (64-bit)  
## Running under: Ubuntu 20.04.3 LTS  
##   
## Matrix products: default  
## BLAS/LAPACK: /usr/lib/x86\_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.8.so  
##   
## locale:  
## [1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C   
## [3] LC\_TIME=en\_US.UTF-8 LC\_COLLATE=en\_US.UTF-8   
## [5] LC\_MONETARY=en\_US.UTF-8 LC\_MESSAGES=C   
## [7] LC\_PAPER=en\_US.UTF-8 LC\_NAME=C   
## [9] LC\_ADDRESS=C LC\_TELEPHONE=C   
## [11] LC\_MEASUREMENT=en\_US.UTF-8 LC\_IDENTIFICATION=C   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] magrittr\_1.5  
##   
## loaded via a namespace (and not attached):  
## [1] knitr\_1.33 hms\_0.5.3 R6\_2.4.1 rlang\_0.4.10   
## [5] stringr\_1.4.0 highr\_0.8 httr\_1.4.2 tools\_4.0.2   
## [9] webshot\_0.5.2 xfun\_0.26 htmltools\_0.5.0 ellipsis\_0.3.1   
## [13] yaml\_2.2.1 leanbuild\_0.1.2 digest\_0.6.25 tibble\_3.0.3   
## [17] lifecycle\_1.0.0 crayon\_1.3.4 bookdown\_0.24 readr\_1.4.0   
## [21] vctrs\_0.3.4 fs\_1.5.0 curl\_4.3 evaluate\_0.14   
## [25] rmarkdown\_2.10 stringi\_1.5.3 compiler\_4.0.2 pillar\_1.4.6   
## [29] pkgconfig\_2.0.3

# Content creation tips  
  
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.  
  
### 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.  
  
### 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.  
  
  
### 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:  
  
![Levels of learning](https://i2.wp.com/wlclassroom.com/wp-content/uploads/2013/03/fx\_bloom\_new.jpg)  
  
[[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.  
  
### 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.  
  
### 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:  
![Emphasizing graphic](https://jhudatascience.org/Informatics\_Research\_Leadership/resources/images/02-team\_guidelines\_files/figure-html/1OU5qeRgN\_fojGbcyu2qEdwlcKpDO6BveWtYW\_u1Hqd4\_gcd0b5dd2d8\_1\_34.png)  
  
Here is an example of a graphic to summarise a section:  
![Summarising graphic](https://jhudatascience.org/Informatics\_Research\_Leadership/resources/images/02-team\_guidelines\_files/figure-html/1OU5qeRgN\_fojGbcyu2qEdwlcKpDO6BveWtYW\_u1Hqd4\_gcd0b5dd2d8\_1\_6.png)  
  
  
### 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:  
  
![Quiz question example](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/question\_example.png)  
  
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.  
  
### 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.  
  
  
### 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.  
  
### 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://jhudatascience.org/Documentation\_and\_Usability/)  
2) [Leadership for Cancer Informatics Research](https://jhudatascience.org/Informatics\_Research\_Leadership/)  
  
  
# Start a new course  
  
1. In the upper right of the landing page for this repository, click the green `Use this template` button and follow the steps to set up your course's GitHub repository.  
  
![Where is the `Use this template` button?](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/template\_button.png)  
  
2. Name your repository and fill in a short description.  
  
![Creating new course repository](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/creating\_new\_course\_repo.png)  
  
  
3. Clone the repository  
  
We then recommend that you set up a project with RStudio and connect this with your repository on GitHub.  
  
If you are unfamiliar with GitHub, we recommend you go through these two chapters in our Reproducibility courses:  
- [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)  
  
<details> <summary> Click here to see more using RStudio for use with Git version control. </summary>  
  
First, to create a new project in RStudio, select File > New Project.  
![Make a new project](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/new\_project.png)  
  
Select a directory for your course to live. We suggest a new directory.  
  
![Select a directory for your new project](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/select\_directory.png)  
  
Select `New Project` for project type.  
  
![Select project type](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/project\_type.png)  
  
If creating a new directory for your project, name the new directory and decide where it should be.  
  
![New Project Directory](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/project\_directory.png)  
  
  
Go back to your new GitHub repository for the course you are creating and click on the green button on the right corner of the landing page to clone the repository.  
  
This will pop up a window that looks like this:  
  
![Clone the GitHub repo to work on the files on your local machine](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/clone.png)  
  
Click the button that looks like a clip board to copy the location of your repository on GitHub.  
  
In the terminal pane of RStudio, make sure you are in the newly created directory for your project.  
  
Then type: `git init` and press enter.  
  
Then type:`git clone` and paste the location of the repository that you just copied from GitHub and press enter.  
  
Finally type: `git pull origin main` and you should see the files on GitHub populate your directory.  
  
This will give you a message like this:  
  
![cloning messages](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/messages.png)  
  
Next make a new git branch in the terminal in RStudio by typing the following and replacing `branch\_name` with something that makes sense for you like `update\_about`: `git checkout -b branch\_name`  
  
Now you are ready to start making changes to files for the next section!  
  
</details>  
  
  
<details> <summary>Also see here for more information about how to make changes to your files with pull requests using RStudio and Git.</summary>  
  
To make changes to files, you can open them in Rstudio by clicking on the file name from the file pane (lower right corner).  
  
![Open File in RStudio](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/file\_pane.png)  
  
After editing a file, make sure you save it.  
  
Then in the terminal you can use Git for version control. Again, if you are new to Git please see this [tutorial](https://happygitwithr.com/).  
  
However, as a brief introduction:  
  
After making changes to the `about.Rmd` file. For example, say you add a new line.  
  
You can type: `git add` and the file name to stage it for committing to GitHub. For example `git add about.Rmd`.  
  
Then type a message about what changes you are committing to the copy on GitHub:  
  
For example:  
  
`git commit -m "updating the about file."`  
  
Then type `git push`.  
  
You will get a message about your branch the first time - you can type what git suggests and press enter. Then on GitHub you should see that new Pull Request is possible.  
  
Click on the green button that says `Compare & pull request`.  
  
![New pull request possible](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/pull\_request.png)  
  
Then fill out the prompt with the information about your pull request. Don't worry if some of the later sections are unclear, we will get to more of that in a bit.  
  
![Submitting pull request](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/pull\_request\_2.png)  
  
When you are done click the `Create pull request` button!  
  
The first time you shouldn't have any conflicts with the main branch, thus you should see something like this:  
  
![Pull request output](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/pull\_request\_3.png)  
  
If everything looks good, you can press the `Merge pull request` button.  
</details>  
  
Now you're ready to continue to ["Set up your repository files"](https://github.com/jhudsl/OTTR\_Template/wiki/Setting-up-your-repository-files)  
  
  
  
# Setting up your repository files  
  
Now if you go to the `Issues` on GitHub for your course you will see issues filed that you can follow to set up the new course! Note that this may require you to wait a minute and refresh the page.  
  
![Issues to Guide you through creating the course](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/guides\_issues.png)  
  
## Setting up the repository  
  
1. Start with the issue called `New Course - Templates to Edit`.  
  
### 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.  
  
#### Using a style set  
  
[Read more about the style sets here](https://github.com/jhudsl/OTTR\_Template/wiki/Change-Title-&-Customize-style#using-a-style-set).  
  
- 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.  
  
  
If you are new to git recall that now you need to push your changes like so [you may benefit from following the steps described here](https://github.com/jhudsl/OTTR\_Template/wiki/Adding-content-to-a-course) and [here](https://github.com/jhudsl/OTTR\_Template/wiki/About-pull-request-review) :  
  
`git commit -m "first changes"`  
  
`git push`  
  
This will likely tell you a message if you are doing this in a branch other than main (which we highly suggest you do so that you can make a pull request also called PR). Type what Git tells you to do and hit enter!  
  
Delete the `code\_output` folder if you do not expect to have any code output (or rename it if you wish to call the folder something else).  
  
Now, you can go through each of the other files listed in the checklist and make sure that the existing Rmd template files are changed to fit your course. Don't worry about adding the bulk of your content yet, you can do that later after you learn more about setting up your course, but this checklist will help you get started.  
  
There are `{}` in these files to get you started filling out information and should be deleted after you've filled them out.  
  
\*\*Remember make your changes in a new branch to enable you to make pull requests with Git.\*\* This will later be very important for the automated tasks that are part of the template. If you are unfamiliar with how to do a pull request, please see the [previous section](https://github.com/jhudsl/OTTR\_Template/wiki/Start-a-new-course) about how to do this.  
  
  
It is likely that you might see something like this with your pull requests which will happen if you have too many new words not in the dictionary:  
  
![Spelling check fails](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/spell\_check\_fails.png)  
  
We will discuss how to get the spelling results and fix your file in a bit, but for now you can press the merge pull request button.  
  
  
Once this is done, you can start on the next issue guide checklist called `New Course - Set Repository Settings`. We will explain how with the [next section](https://github.com/jhudsl/OTTR\_Template/wiki/Setting-up-your-repository-settings).  
  
  
  
# Setting up your repository settings  
  
### Recommended repository settings:  
  
With your course repository set up, there are some settings recommended for development.  
  
Using the issue guide checklist called `New Course - Set Repository Settings`, you can walk through the steps for setting up your repository as we would recommend.  
  
![Next issues checklist for repository settings](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/guides\_issues\_2.png)  
  
### Set up GitHub pages  
  
Go to `Settings` > `Pages`.  
  
![Find pages settings](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/pages\_settings.png)  
  
![Change pages settings](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/change\_pages\_settings.png)  
  
- 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.   
  
![Enforce HTTPS](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/change\_https.png)  
  
### Set up branches  
  
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.  
  
![Add branch rule](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/branches.png)  
  
For `Branch name pattern`, put `main`.  
  
![Put main for branch name pattern](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/main.png)  
  
\_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`.  
  
![Branch settings](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/branch\_settings.png)  
  
After setting up these new branch items, click `Create` and `Save changes`.  
  
After you submit a pull request if you haven't already done so, you will see several automatic checks happening. Don't be alarmed if this image looks slightly different as we make updates to the template:  
  
![Automatic updates](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/automatic\_checks.png)  
  
  
These checks will do important things like:  
\* check that the all the urls actually take learners somewhere  
\* check that the code is styled using the `stylr` package  
\* check that the spelling is correct using the `spelling` package  
\* create previews of the rendered versions of the course  
  
  
See the [Github Actions page](https://github.com/jhudsl/OTTR\_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots) for more details on these.  
  
  
If you are a part of JHU DaSL then you can skip the next section and the issue about GitHub secrets and start adding more content!  
  
Otherwise, you will need to do the [next section](https://github.com/jhudsl/OTTR\_Template/wiki/Setting-up-GitHub-secrets) to set up Git Secrets.  
  
There is one more issue that you can do to get updates for template files.  
![one more issue](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/guides\_issues\_4.png)  
  
We recommend that you do this but it is optional.  
  
![one more issue](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/issue\_enroll.png)  
  
 See [this section](https://github.com/jhudsl/OTTR\_Template/wiki/Receiving-template-updates) to learn more about this or continue following along with the guide which will tell you how to set this up in a bit.  
  
Note that you can comment and close the issues you finish if you click on the issue and scroll down to the comment field to add comments and click the `close issue` button if you are finished.  
  
  
  
# Setting up GitHub secrets  
  
The GitHub actions that this repository uses needs three GitHub secrets set up if you are not a part of the `jhudsl` organization.  
  
It's important that these are set up and named exactly what they are below in order for GitHub actions to work correctly.  
  
See [GitHub Actions section](https://github.com/jhudsl/OTTR\_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots) for how you can customize Github actions depending on the needs of the course you are working on.  
  
![GitHub secrets](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/git-secrets.png)  
  
To set up these repository secrets, on your repository's main GitHub page, go to `Settings` and scroll down to see `Secrets` on the left side menu bar.  
  
For each new secret, click the `New repository secret` button and set each as follows, clicking `Add secret` as you fill each in appropriately:   
  
\_Name: `GH\_PAT`\_:   
\*If you are a part of `jhudsl` organization you do not need to set these.  
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.  
  
### Dockerhub related secrets  
  
Note these are not required if [Docker update GitHub actions are not turned on](#about-customizing-render-bookdownyml-also-called-build-all).  
\*If you are a part of `jhudsl` organization you do not need to set these.  
  
\_Name: `DOCKERHUB\_USERNAME`\_:   
For `value`: put your login username for https://hub.docker.com/  
  
\_Name: `DOCKERHUB\_TOKEN`\_:   
For `value`: put a access token for Dockerhub.  
You can create this by following [these instructions](https://docs.docker.com/docker-hub/access-tokens/#create-an-access-token).  
  
  
  
# Adding content to a course  
  
Here's a standard workflow for adding new content to your course. If you are new to git, it is recommended you go through [Happy Git and GitHub for the useR](https://happygitwithr.com/). This is not meant to be a complete guide to git but just a summary of the steps you would need to take to add changes to a course.  
  
Also if you do not do so already, it's recommended you use a git client to help you manage your branches more easily.  
[GitKraken](https://www.gitkraken.com/) is one such git client that is handy for this.  
You may also find this [git cheatsheet](https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet) helpful.  
\_Note\_: The commands in these steps all for running in Terminal/Command Prompt.  
  
## Getting going with a new course (you only have to do this once)  
1. [git clone](https://git-scm.com/docs/git-clone) the repo you are working on if you don't have it locally yet.  
  
## Starting up a new set of changes  
2. If you are looking at a particular existing issue you want to address, [assign yourself to it](https://docs.github.com/en/issues/tracking-your-work-with-issues/assigning-issues-and-pull-requests-to-other-github-users) so that others know not to be working on it. If there is no existing issue for the changes that need to be made you may want to consider making one so others know not to address that themselves if they have the same idea.  
3. Change your directory to be that git repository directory on your computer  
[`git checkout -b “name-of-new-branch”`](https://git-scm.com/docs/git-checkout).  
4. Start up the Docker image:  
[`docker run -it -v $PWD:/home/rstudio -e PASSWORD=CHOOSE\_PASSWORD -p 8787:8787 jhudsl/course\_template`](https://github.com/jhudsl/OTTR\_Template/wiki/Using-Docker) But replace `CHOOSE\_PASSWORD` with a password of your choosing.  
5. Navigate to `localhost:8787` on your browser (your username is Rstudio and password is whatever you set it to be in the previous step).  
  
## Editing content  
6. In the RStudio window you just opened, copy the [`02-chapter\_of\_course.Rmd`](https://github.com/jhudsl/OTTR\_Template/blob/main/02-chapter\_of\_course.Rmd) course and use that as a guide to [add a new chapter](https://github.com/jhudsl/OTTR\_Template/wiki/Publishing-with-Bookdown).  
7. Make your edits. If your edits involve images, graphics, or videos, make sure you follow the set up in [adding images and graphics in text](https://github.com/jhudsl/OTTR\_Template/wiki/Setting-up-images-and-graphics#adding-images-and-graphics-in-text)  
8. To preview how your [edits look in Bookdown](https://github.com/jhudsl/OTTR\_Template/wiki/Publishing-with-Bookdown), you can run bookdown::serve\_book().  
8. As you’ve made enough edits and want to save them run [`git add filepath`](https://git-scm.com/docs/git-add).  
9. Now [`git commit -m “Add a pertinent message here”`](https://git-scm.com/docs/git-commit) to add your edits to your branch.  
10. When you are ready to push changes to be online, run [`git push`](https://git-scm.com/docs/git-push).  
11. Repeat 6-10 as needed until you are happy with your changes and think they are ready for review. Keep in mind incremental changes can sometimes be a better strategy, as it is easier to fix things if you realize that something isn't rendering as expected. It also helps to avoid losing work in case something happens to your computer or work before you are able to push your changes.  
  
## Changing title  
  
The title is specified on the index.Rmd page in the yml header. Follow the directions above and modify the yaml header to change the title for your course.

## Filing a PR  
12. Now go to your GitHub repository’s website and click on [Pull Requests](https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests).  
13. If you pushed your changes recently you will see a prompt to create a pull request as a yellow banner. Click on `Create Pull Request`.  
14. The pull request will be auto-populated with the template -- fill out the pull request prompts and check off the checklist marks by placing an `x` between the brackets of the items you have completed.  
15. Look at the [`files changed`](https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-comparing-branches-in-pull-requests) tab. Does it show the changes you expected? If not, then continue to make edits following steps 6 - 10 as needed.  
16. The Github actions will automatically generate a preview of your course and post it to a comment on your pull request like this:  
![](https://raw.githubusercontent.com/jhudsl/OTTR\_Template/main/resources/screenshots/gha-preview-comment.png)  
17. When you think it is ready for review, [request a reviewer](https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/requesting-a-pull-request-review) by clicking on `Reviewer` on the right side and choose an appropriate person to review your changes.  
  
Have you been requested as a reviewer or just received a review? [Read this page](https://github.com/jhudsl/OTTR\_Template/wiki/Pull-Request-review-tips) for advice on reviewing/receiving reviews.  
  
## Merging your PR  
18. When you receive feedback, repeat steps 6 - 10 to make edits and incorporate the feedback, all the while communicating with your reviewer through comments on the pull request.  
19. If any of the checks have failed, click on `Details` and try to investigate why.  
 - For failed spell checks follow the [instructions here](https://github.com/jhudsl/OTTR\_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots#style-guide).  
 - For failed URL checks follow the [instructions here](https://github.com/jhudsl/OTTR\_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots#url-checking).  
20. If the checks have passed, and the reviewer thinks it's ready, use the down arrow next to the `merge` button to choose `squash and merge`.  
21. [Close the issues](https://github.blog/2013-05-14-closing-issues-via-pull-requests/) that your changes were addressing. File any new issues during any step to keep track of what you need to do next! Start with step 2 all over again.  
  
Congrats! You did it!  
  
  
  
# About pull request review  
  
Everybody benefits from code and content review: The reviewer, the reviewee and whoever comes to look at the reviewed code and content later!  
  
To quote [Parker 2017](https://peerj.com/preprints/3210v1/):  
  
> Code review will not guarantee an accurate analysis, but it’s one of the most reliable ways of establishing one that is more accurate than before.  
  
### The mechanics of review:  
  
1. A problem is identified  
2. An [issue](https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues) is filed that describes that problem  
3. A writer files a [pull request](https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests) that contains changes that addresses the problem.  
4. A reviewer [reviews the changes](https://docs.github.com/en/github/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/about-pull-request-reviews) on that pull request.  
5. The reviewer [reviews and recommends edits](https://docs.github.com/en/github/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/reviewing-proposed-changes-in-a-pull-request) to those changes or accepts them as is.  
6. The writer [incorporates those changes](https://docs.github.com/en/github/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/incorporating-feedback-in-your-pull-request) as best they can.  
7. (Steps 4-6) with comments between the are repeated until the reviewer and writer accept the changes.  
8. Changes are added to the code base by [merging the pull request](https://docs.github.com/en/github/collaborating-with-pull-requests/incorporating-changes-from-a-pull-request/merging-a-pull-request).  
9. A new problem is identified (and the whole cycle starts again).  
  
That's the workflow, but how do you get the most out of code review, both as the reviewer and writer?  
Here's some tips about getting the most out of code review:  
  
### Tips for the writer  
- Put context around the changes you are proposing, linking any relevant issues, prior pull requests, or relevant code/media.  
- Review your own pull request before requesting the review of someone else; check the `Files Changed` tab.  
- If a pull request is still in progress change it to a draft so no one reviews it before you are ready.  
- Try to make sure your pull requests aren't too long! Code reviewing fatigue is very real.  
- Highlight for the reviewer what kind of feedback you are looking for.  
- Try to explain or ask for help on any peculiar parts of your PR or if there's any parts you are stuck on.  
- Accept reviews with a helpful frame of mind -- it's perfectly okay to be wrong!  
  
### Tips for the reviewer  
- Check that the pull request meets the goals that the writer is intending.  
- If you don't understand, ask more questions!  
- Take careful review of areas that are more complex or seem odd.  
- Give reviews loaded with [empathy and positivity](https://slack.engineering/on-empathy-pull-requests/) !  
- Utilize suggestions where possible.  
- If you are completely lost, don't be afraid to ask the writer to reconfigure or expound on what they mean.  
  
### Recommended reading:   
- [Effective code reviews](https://medium.com/@omarmeky/effective-code-reviews-4bca0da61dcc)  
- [9 Tips for Opening a Better Pull Request](https://www.mutuallyhuman.com/blog/9-tips-for-opening-a-better-pull-request/)  
- [Writing proper GitHub issues](https://medium.com/nyc-planning-digital/writing-a-proper-github-issue-97427d62a20f)  
- [On Empathy and Pull Requests](https://slack.engineering/on-empathy-pull-requests/)  
- [Best practices for Code Review](https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/)  
- [Writing Great Scientific Code](http://alexanderganderson.github.io/code/2016/10/12/coding\_tips.html)  
- [Six Commandments for Writing Good Code](https://www.toptal.com/software/six-commandments-of-good-code)  
  
  
# Setting up images and graphics  
  
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:  
  
  
```markdown  
`r ''````{r, include=FALSE}  
`r ''`leanbuild::set\_knitr\_image\_path()  
`r ''````

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.

### 3.2.1 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.



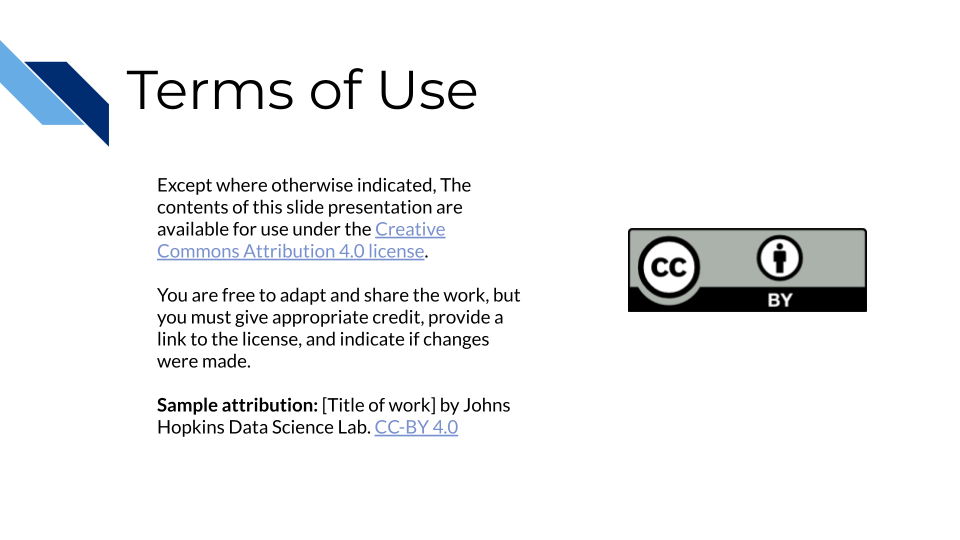
* If you are working on a project (more likely) with the [Johns Hopkins Data Science Lab](http://jhudatascience.org/) than import the theme from this [template](https://docs.google.com/presentation/d/1qa3v_zlNedCKHQ6IYF5MM1ZLZLgHUs0U4usgDRRe3W8/edit?usp=sharing).

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:

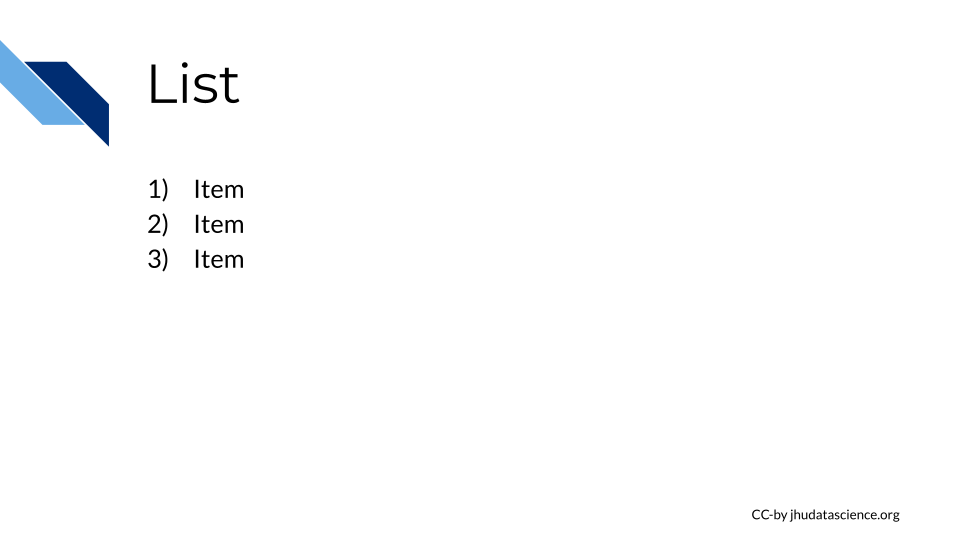


Figure : list\_layout

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



Figure : source\_layout

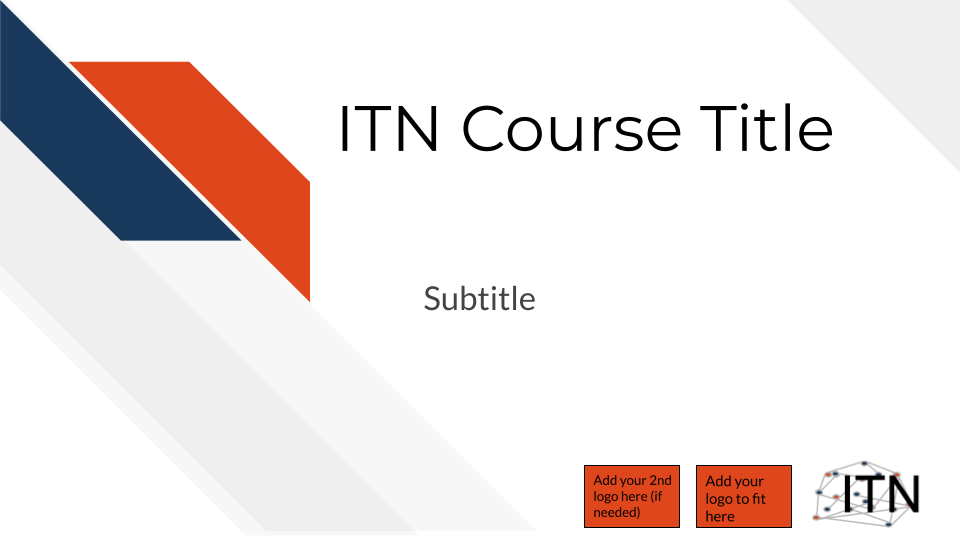
Please update the text at the bottom to describe the source. If it is not from JHU, remove the JHU information.

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.

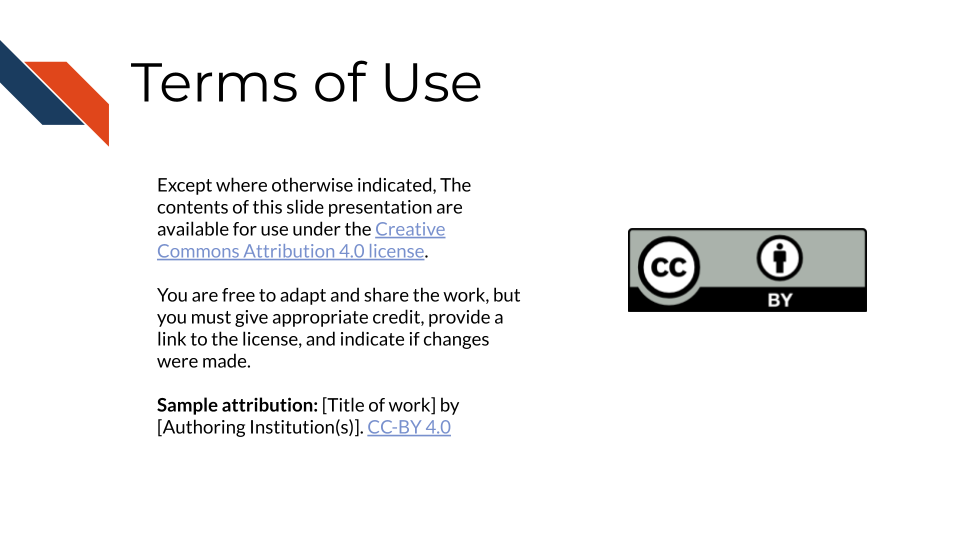
Once complete, you can incorporate slides into your course using the instructions in the [Adding images and graphics in text](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-images-and-graphics#adding-images-and-graphics-in-text) below.

### 3.2.2 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.

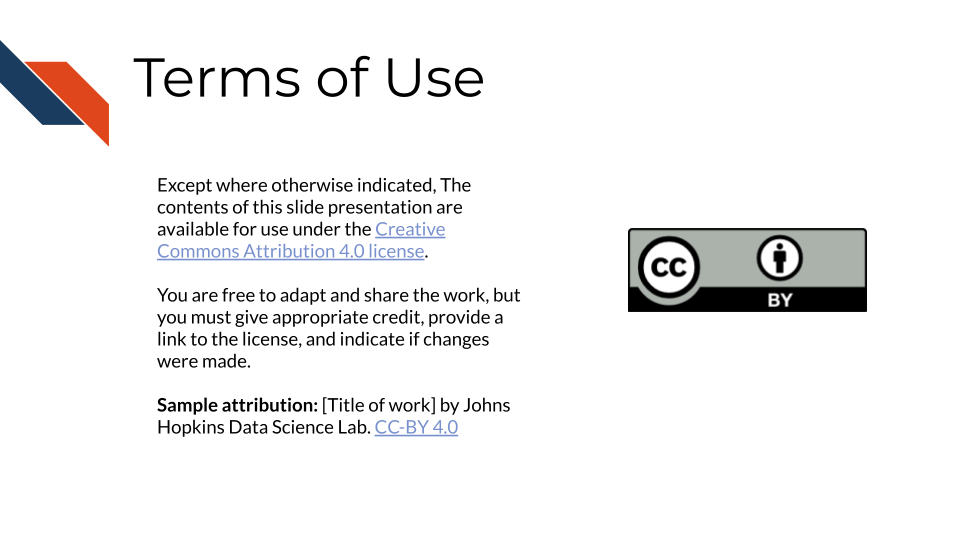


Figure : Terms of Use

For any major point, please select this layout:



Figure : Major\_point\_layout

For itemized lists, please select this layout:

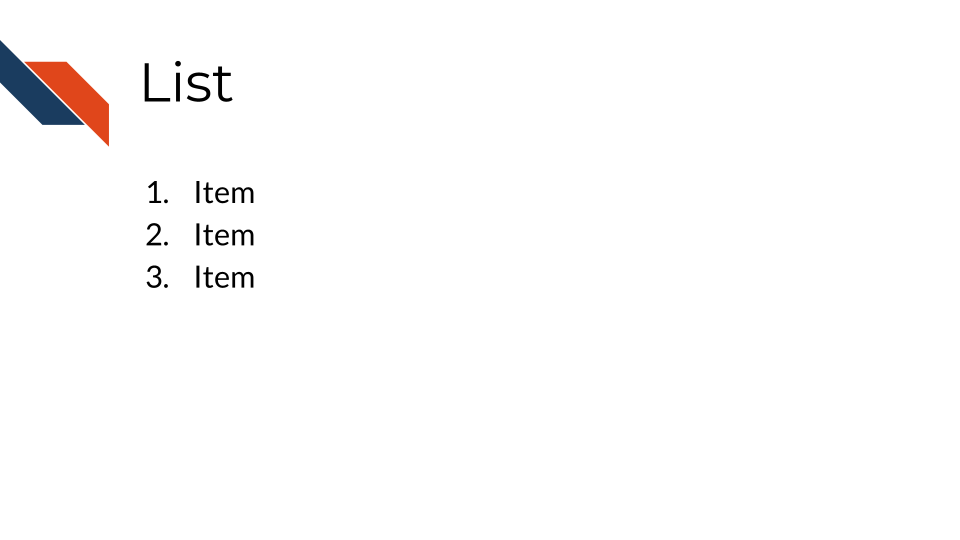


Figure : list\_layout

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.

### 3.2.3 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/).

### 3.2.4 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%"}  
leanbuild::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.

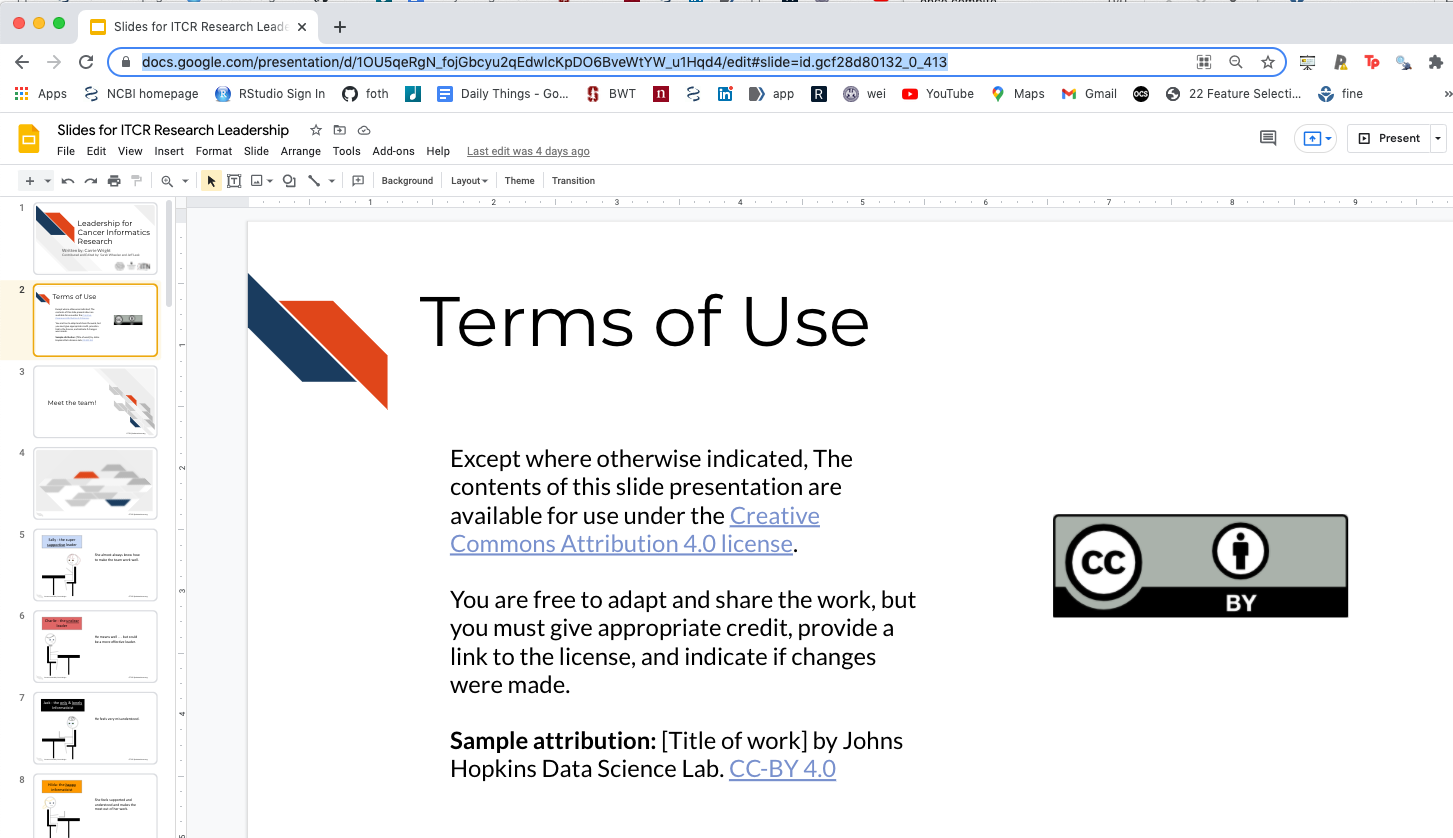


Figure : Google slide url

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.

### 3.2.5 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.

### 3.2.6 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.

### 3.2.7 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).

# 4 OTTR Tools

# 5 The tools in this template

## 5.1 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.

# 6 Using Docker

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.

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.

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 image as a starting point; you can download it from jhudsl/course\_template 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

This pulls the course\_template 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).

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 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

*On a Windows*:

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

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!

In a web browser navigate to the following to get to an RStudio that is ran 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.

*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/).

### 6.0.1 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 two things to get this going:

1. You need to uncomment the Docker image update [following these instructions](#X64360bdde474af7ac5cb1f5d501615222749632).
2. You’ll need to set up a Dockerhub account and set up Dockerhub secrets [following these instructions](#dockerhub-related-secrets).

To start up a new Docker image for your new course, you can start with the Dockerfile in this repository and add the additional packages you need using the tips in the next section, but you’ll need to change the tag.

If you wish to keep the Docker image underneath the Dockerhub jhudsl organization account, you’ll need to be granted access to that organization – contact one of the jhudsl team to have them add you.

### 6.0.2 Adding packages to the Dockerfile

If you find you need a new package to run the code you are adding, you’ll need to add the package to the docker/Dockerfile.

Try to keep things in alphabetical order where possible.

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

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

To add an R package from GitHub you can add a line that follows this general format (need a comma and space):

RUN Rscript --vanilla install\_github.R \  
 --packages "jhudsl/didactr, jhudsl/leanbuild, <NEW\_PACKAGE>" \

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"

#### 6.0.2.2 Rebuilding the Docker image

When you’ve added a package to the Dockerfile, you’ll need to check that it builds successfully before including it in a pull request.

First create a GITHUB token file by making a token and copying a pasting it into a plain text file named docker/github\_token.txt.

Then you’ll need to rebuild the docker image using this command:

docker build -f docker/Dockerfile . -t jhudsl/course\_template

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.

Once it builds successfully, you should file a PR with your Dockerfile changes. Once the changes are accepted and merged, we’ll need to push the updated image to Dockerhub using:

docker push jhudsl/<TAG\_FOR\_COURSE>

When you file a pull request, the Dockerfile build will be tested automatically by the [GitHub actions](https://github.com/jhudsl/OTTR_Template/wiki/How-to-set-up-and-customize-GitHub-actions-robots).

The Github actions that this repository uses needs four Github secrets set up if you are not a part of the jhudsl organization.

It’s important that these are set up and named exactly what they are below in order for Github actions to work correctly.

See [Github Actions section](#github-actions) for how you can customize Github actions which can change the necessity of these secrets.

![Github secrets](data:text/plain; charset=utf-8;base64,NDA0OiBOb3QgRm91bmQ=)

Figure : Github secrets

To set up these repository secrets, on your repository’s main Github page, go to Settings and scroll down to see Secrets on the left side menu bar.

For each new secret, click the New repository secret button and set each as follows, clicking Add secret as you fill each in appropriately:

*Name: GH\_PAT*:  
\*If you are a part of jhudsl organization you do not need to set these. 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.

## 6.1 For a course that will need changes to Docker image

If you know for sure that the course you are working on will never require Docker updates – this may be the case if the course doesn’t any interactive code as a part of the material, you can delete the Docker updating chunk or keep it commented out.

However, if you will need to make any change to the Docker image specific to the course you are working on, in the file .github/render-bookdown.yml you should uncomment the ###### START OF DOCKER UPDATE CHUNK up to the part that says ###### END OF DOCKER UPDATE CHUNK.

This will require you to set up the [Github secrets for Docker](https://github.com/jhudsl/OTTR_Template/wiki/Setting-up-GitHub-secrets#dockerhub-related-secrets) (see next section).

## 6.2 Dockerhub related secrets

Note these are not required if [Docker update Github actions are not turned on](#X84eb6565c5fec599949f779da656c6c42efb577). \*If you are a part of jhudsl organization you do not need to set these.

*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).

# 7 Spell check

Github actions will automatically [run a spell check on all Rmds](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. 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 2 spelling errors and you’ll need to resolve those before being able to merge your pull request.

To resolve those spelling errors, go to this repository’s Actions tab. Then, click on the GitHub action from the PR you just submitted. Scroll all the way down to Artifacts and click spell-check-results. This will download a zip file with a TSV that lists all the spelling errors.

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.

### 7.0.1 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:

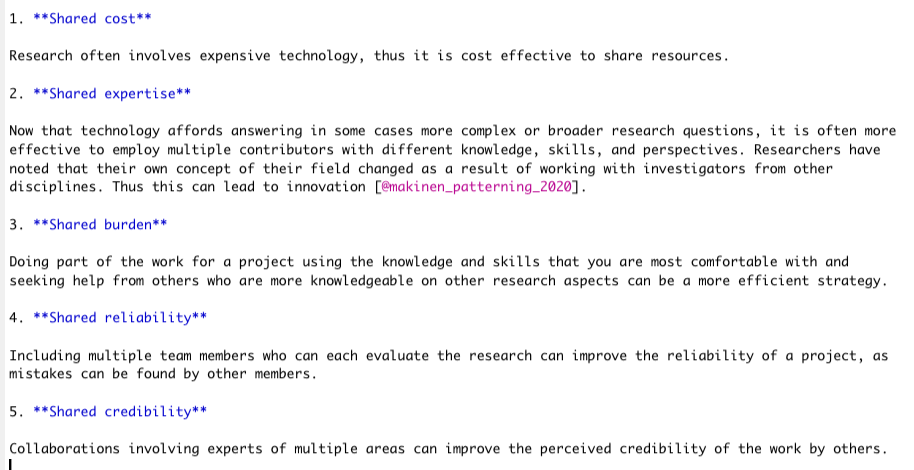
Rscript scripts/spell-check.R

# 8 Leanpub conversion tips

To allow for easy conversion to Leanpub, we suggest the following:

1. When making a list of points, if you make each major heading a separate line, this can avoid numbering issues. This also works better typically then using a bulleted list unless the list items are short (less than one line).

Here you can see how this list in bookdown made a nice list in Leanpub.

Here is the Bookdown version: 

Here is the Leanpub version: ![Example in Leanpub](data:text/plain; charset=utf-8;base64,NDA0OiBOb3QgRm91bmQ=)

1. Avoid images between numeric list items.
2. For making empty lines use &nbsp; as it is compatible with Leanpub.

# 9 Setting up docx rendering

You might want your course available for download as a docx. For example, you might be running a “train-the-trainer” workshop where trainees don’t feel comfortable using Github to edit the lessons for their own use.

The following yml in index.Rmd allows you to render the docx with a table of contents:

output:  
 bookdown::word\_document2:  
 toc: true

You can also incorporate a template docx if you have headers and logos you want to use. To incorporate a template, make sure you add the reference\_docx argument:

output:  
 bookdown::word\_document2:  
 reference\_docx: <path/to/template>.docx  
 toc: true

Learn more about templates [here](https://bookdown.org/yihui/rmarkdown-cookbook/word-template.html).

## 9.1 Using Google Docs to solicit feedback

Google Docs can be a great way to get feedback from collaborators who aren’t comfortable developing in the GitHub environment. docx versions of the course are automatically rendered in pull requests and can be downloaded from the autogenerated comment.

* Once you’ve downloaded the docx, this can be uploaded to Google Drive and shared with the individuals you would like to solicit feedback or new material from. In Google Drive, click + New in the left corner and then File Upload. Choose the docx file you downloaded.
* You may also want to include a link to the commit ID the Google Doc came from so you are able to keep track of what changes have occurred since the Google doc was set up initially.
* So you are able to keep track of the changes to the docx version of the course, its advisable to set to “suggestions” when you share the document.
* Then as comments and suggestions trickle in, a lead author who is comfortable with GitHub can incorporate those comments into an existing or new pull request which can checked for its rendering and eventually added to the main content branch.

# 10 Platform specific set up

# 11 Platform specific set up

## 11.1 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:

## 11.2 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)

# 12 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.

# 13 Publishing on Leanpub

The [\_Quizzes template repository](https://github.com/jhudsl/OTTR_Quizzes) includes all of the files that you need to convert your Bookdown course that was set up from a OTTR\_Template to a [Leanpub](https://leanpub.com/) course with quizzes.

These repositories are separate so that the quizzes and answers can be kept private. If you haven’t created a OTTR\_Template course repository for your course [template repository’s getting started section Wiki](https://github.com/jhudsl/OTTR_Template/wiki/Getting-started) and start there.

If you encounter any problems or have ideas for improvements to this template repository or this getting started Wiki, please [file an issue here](https://github.com/jhudsl/OTTR_Template/issues/new/choose)! Your feedback is very much appreciated.

## 13.1 How to use these repositories:

If you don’t wish to have quizzes with your material or publish on Leanpub then you do not need the \_Quizzes repository template, and the [OTTR\_Template course repository](https://github.com/jhudsl/OTTR_Template/) should be sufficient for your needs.

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.

## 13.2 Setting up your Leanpub Github repository

In the upper right of this screen, click Use this template and follow the steps to set up your course’s GitHub repository.

Name your repository fill in a short description.

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

##### 13.2.0.0.1 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.

## 13.3 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 or Cool\_CourseOTTR\_Template course; it’s Leanpub 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)

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 course will not be updated.*

## 13.4 Setting up quizzes

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; this ensures its incorporated by Leanpub in the correct order.

You need to modify the Book.txt file in the manuscript directory to include the .md files that you wish in the order that you would like. We have also included a quiz example. 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.

## 13.5 Leanpub rendering

For convenience purposes the leanbuild 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 leanbuild::bookdown\_to\_Quizzes() function at the top of the repository. You can also run this command manually if you wish.

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

## 13.6 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.

1. Once you are ready and you like your course, you can click the “Publish New Version” instead of “Preview New Version”.

## 13.7 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

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

## 13.8 Setting up the \_Quizzes repository checklist:

* ☐ 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 course’s Leanpub has been set up](#hosting-your-course-on-leanpub)
* ☐ This repo has been enrolled in [automatic updates from the original template](#X93252b3fcf4d5772e88d152ba50f2b8f15550e8).

## 13.9 Linking the OTTR\_Template course repository to the \_Quizzes repository

transfer-rendered-files.yml is a Github action that will copy over the output docs/ files rendered by Bookdown to a parallel Leanpub repository and it lives in the OTTR\_Template course repository.

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.

# 14 Publishing on Coursera

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.

## 14.1 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.

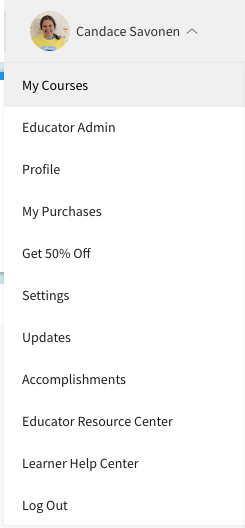
## 14.2 Converting your files for upload to Coursera

The Github actions set up in the [render-bookdown.yml](https://github.com/jhudsl/OTTR_Template/blob/main/.github/workflows/render-bookdown.yml) also 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/coursera/ folder. Click on your chapter htmls to see what it will look like.

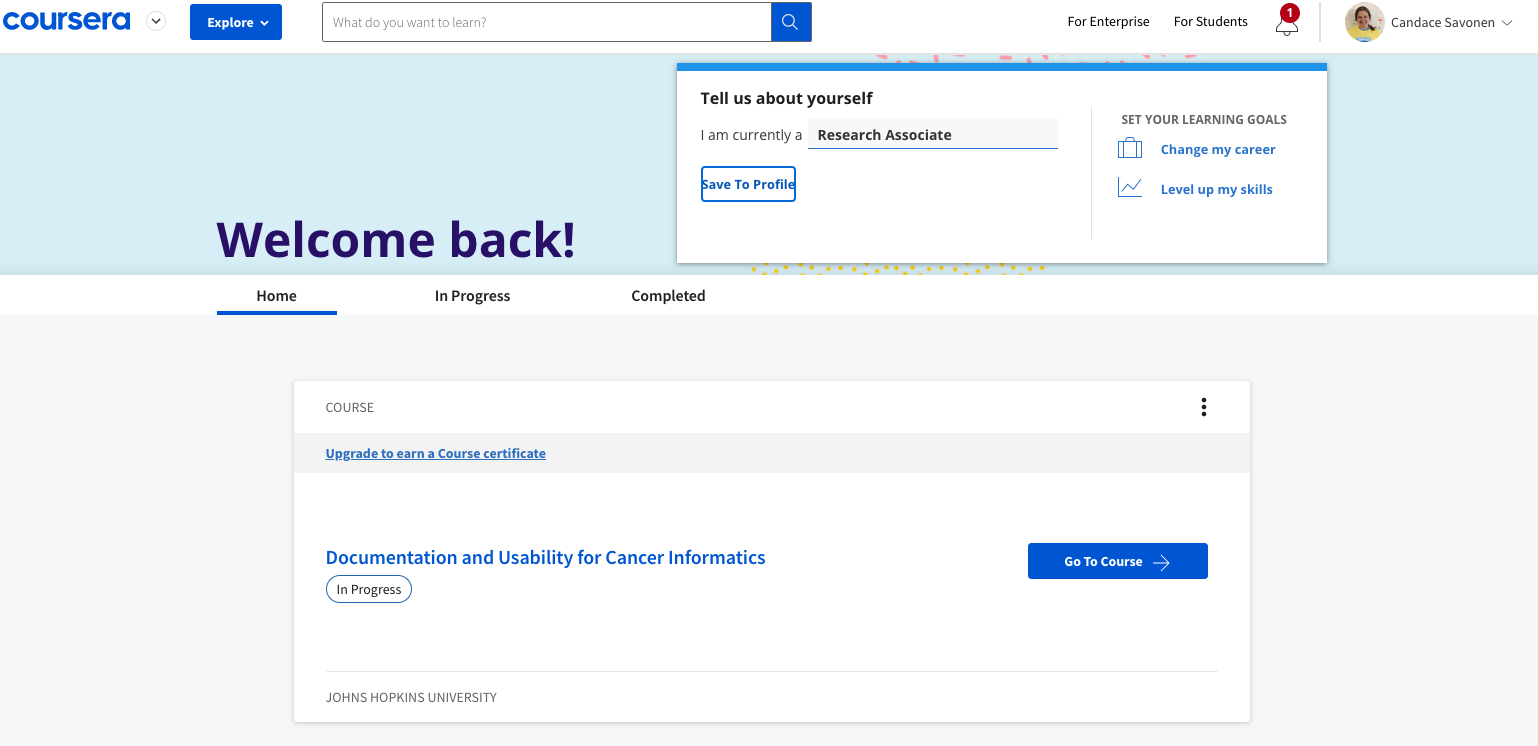
In the Leanpub repo, Leanpub-formatted quizzes can be converted to a Coursera yaml format. Or alternatively if you do not wish to have a Leanpub version of your course, you can either manually enter your Coursera quizzes or manually write a Coursera yaml file following the format of our Coursera template quiz.

## 14.3 Navigating to your course on Coursera

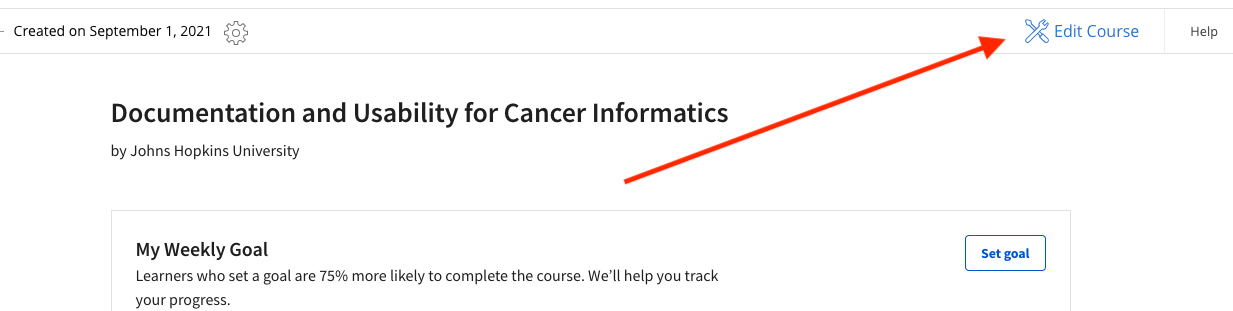
After your [course shell has been added to your account](#setting-up-your-coursera-course) you can navigate to it by going to My Courses:



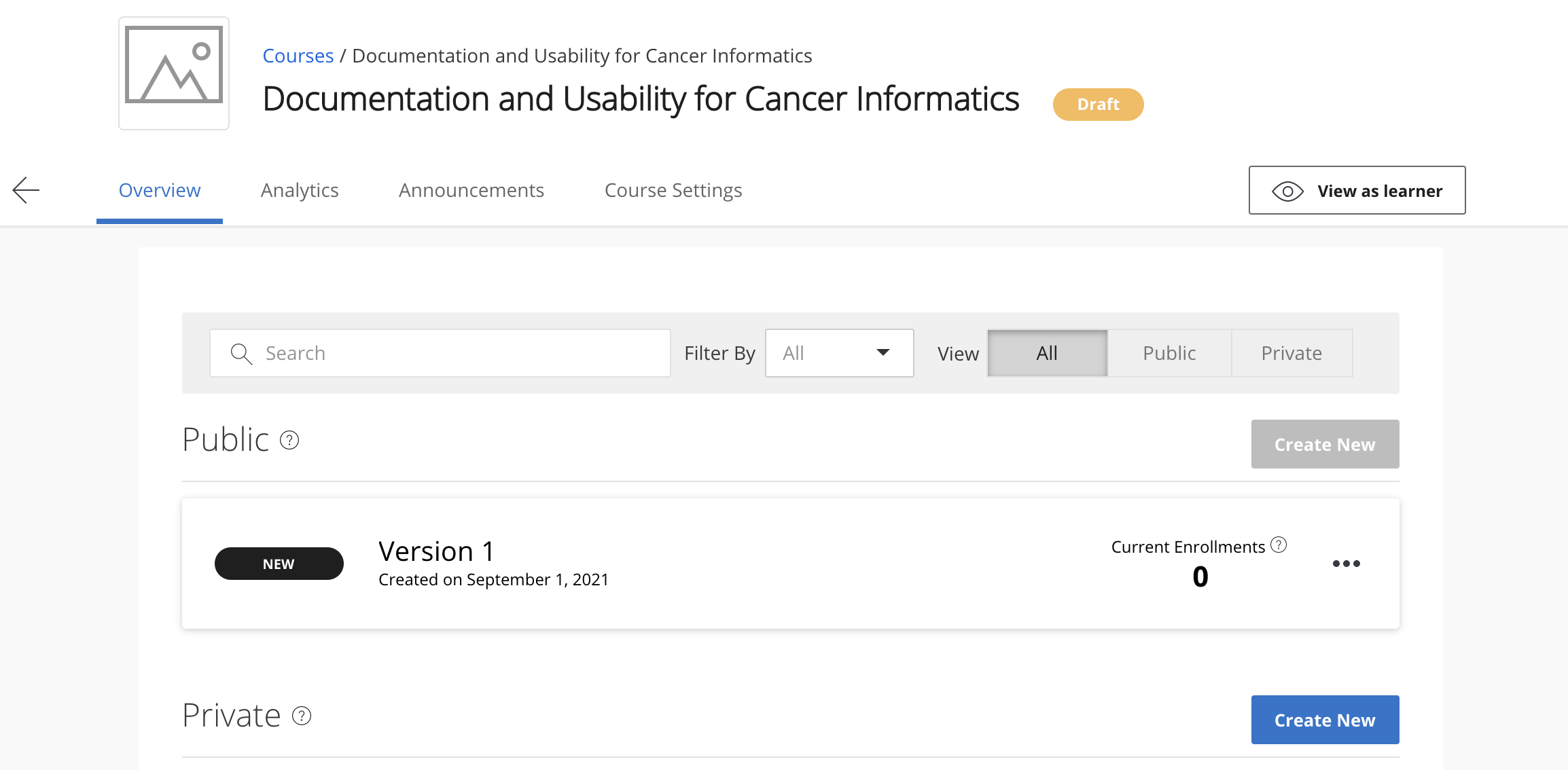
And you should see it listed:



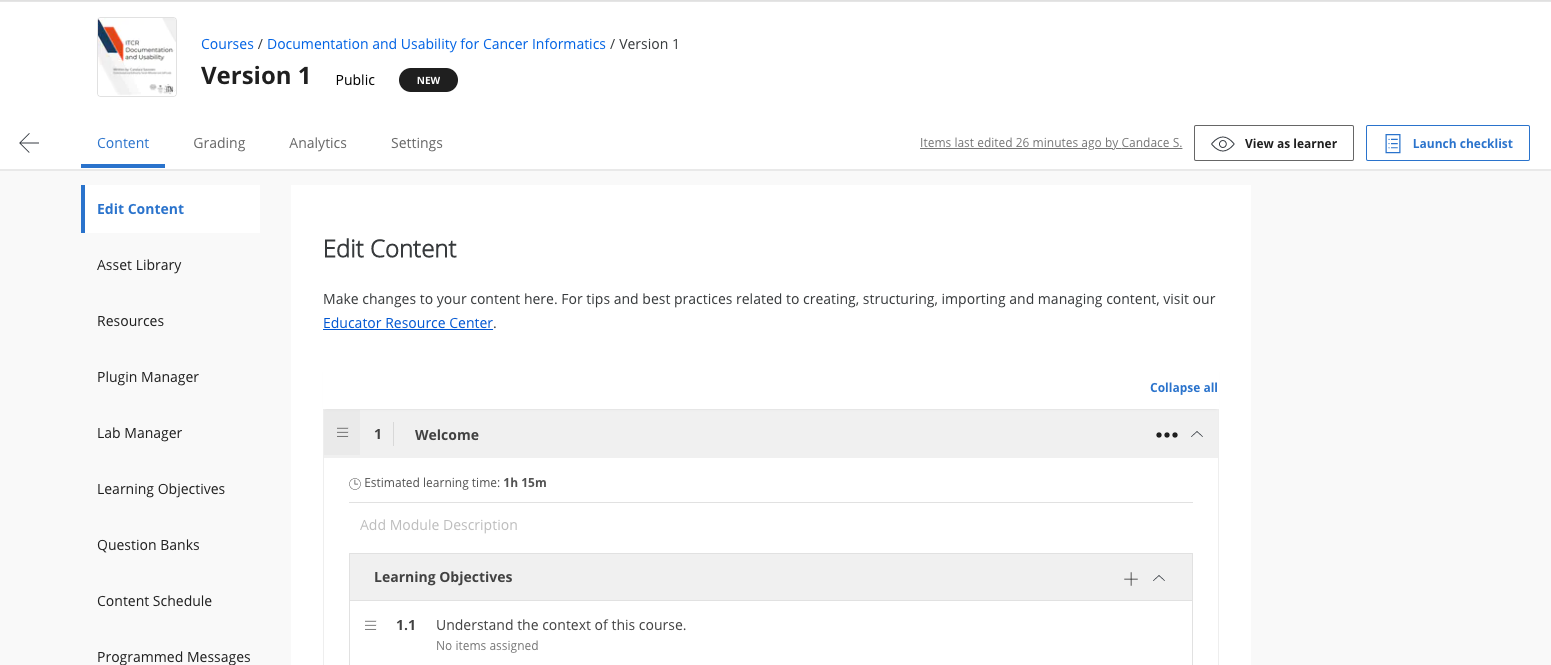
You can click on Go to Course. And on your main course page click Edit Course in the right corner.



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



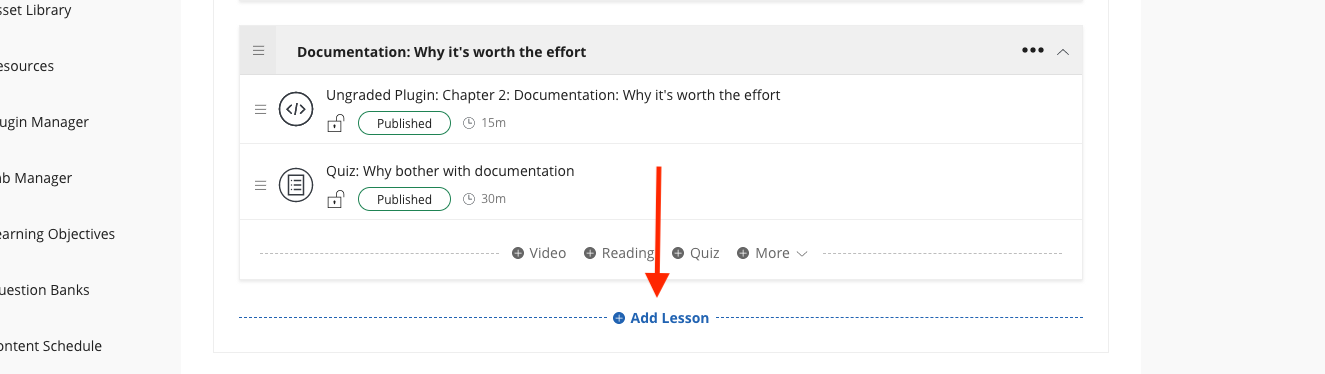
Now you are on the Edit content page and ready to add lessons, modules and quizzes!



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!

### 14.3.1 Adding new chapters to Coursera

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

You can see a preview of these chapters you will link to in Coursera by going to your [Github pages](https://guides.github.com/features/pages/), but adding /coursera 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:

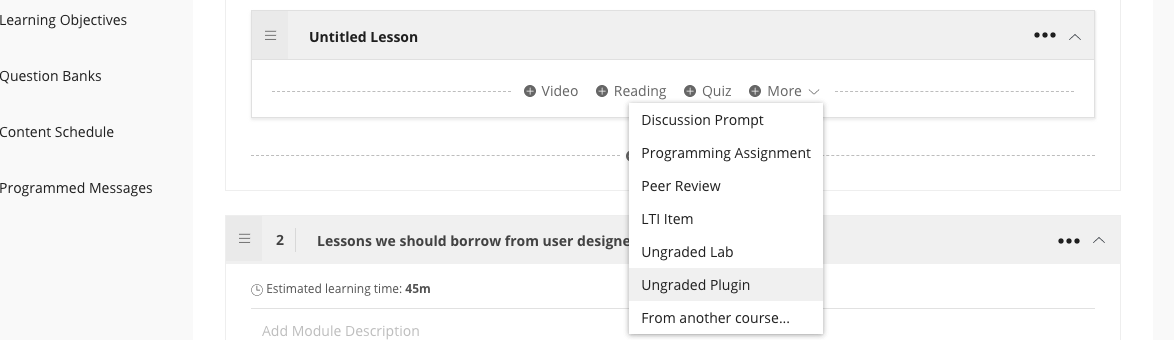
<main-github-pages-url>/coursera/<html\_filename>

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:

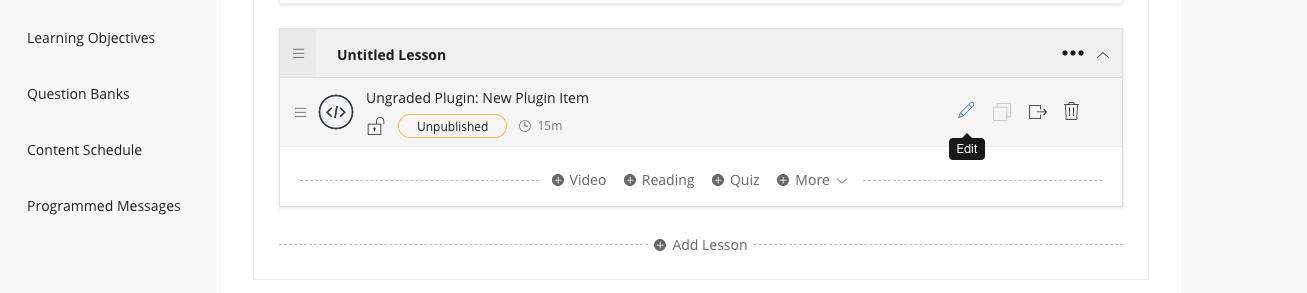
https://jhudatascience.org/OTTR\_Template/coursera/introduction.html

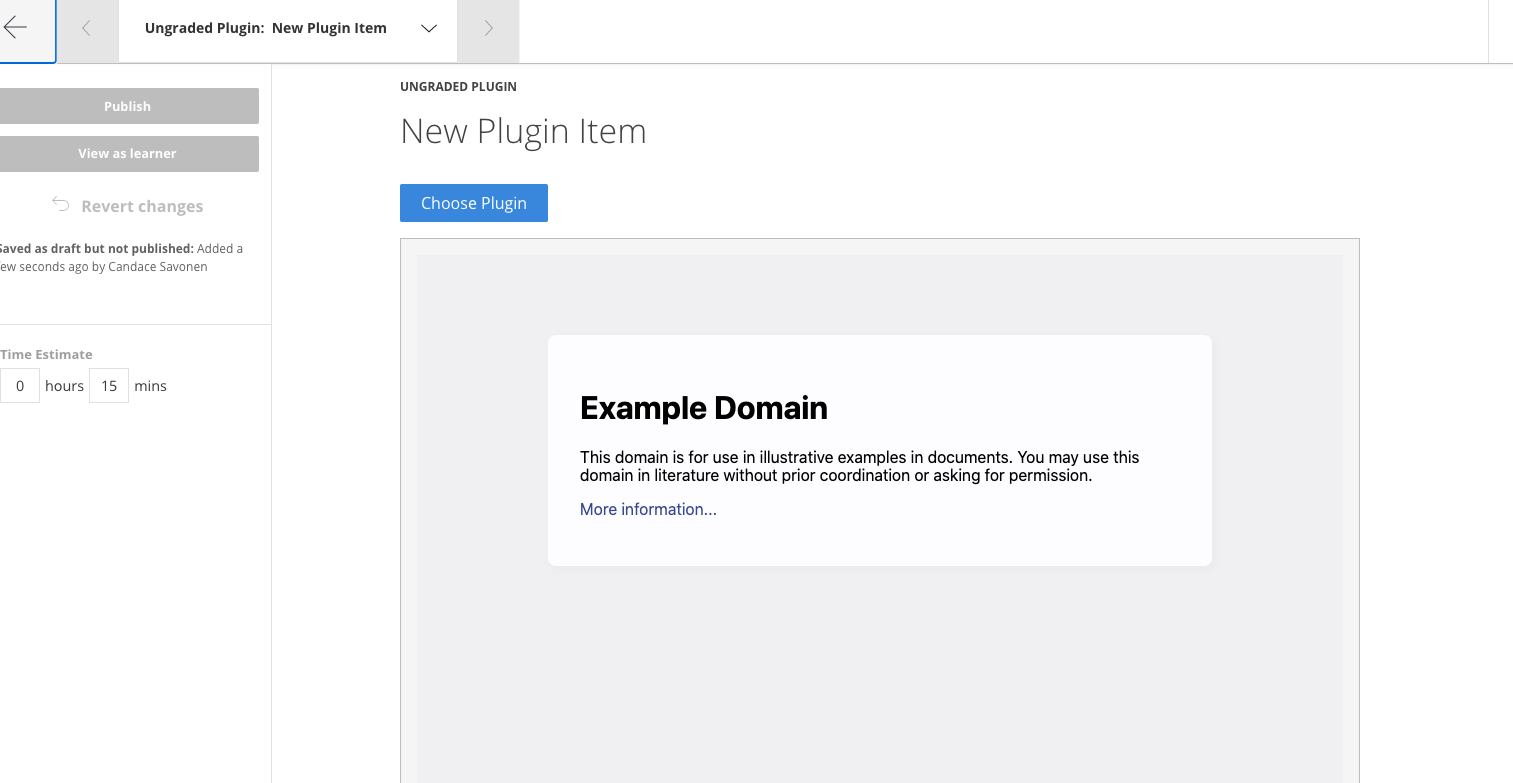
To add this URL to Coursera, add a new lesson, click + More and choose Ungraded Plugin.



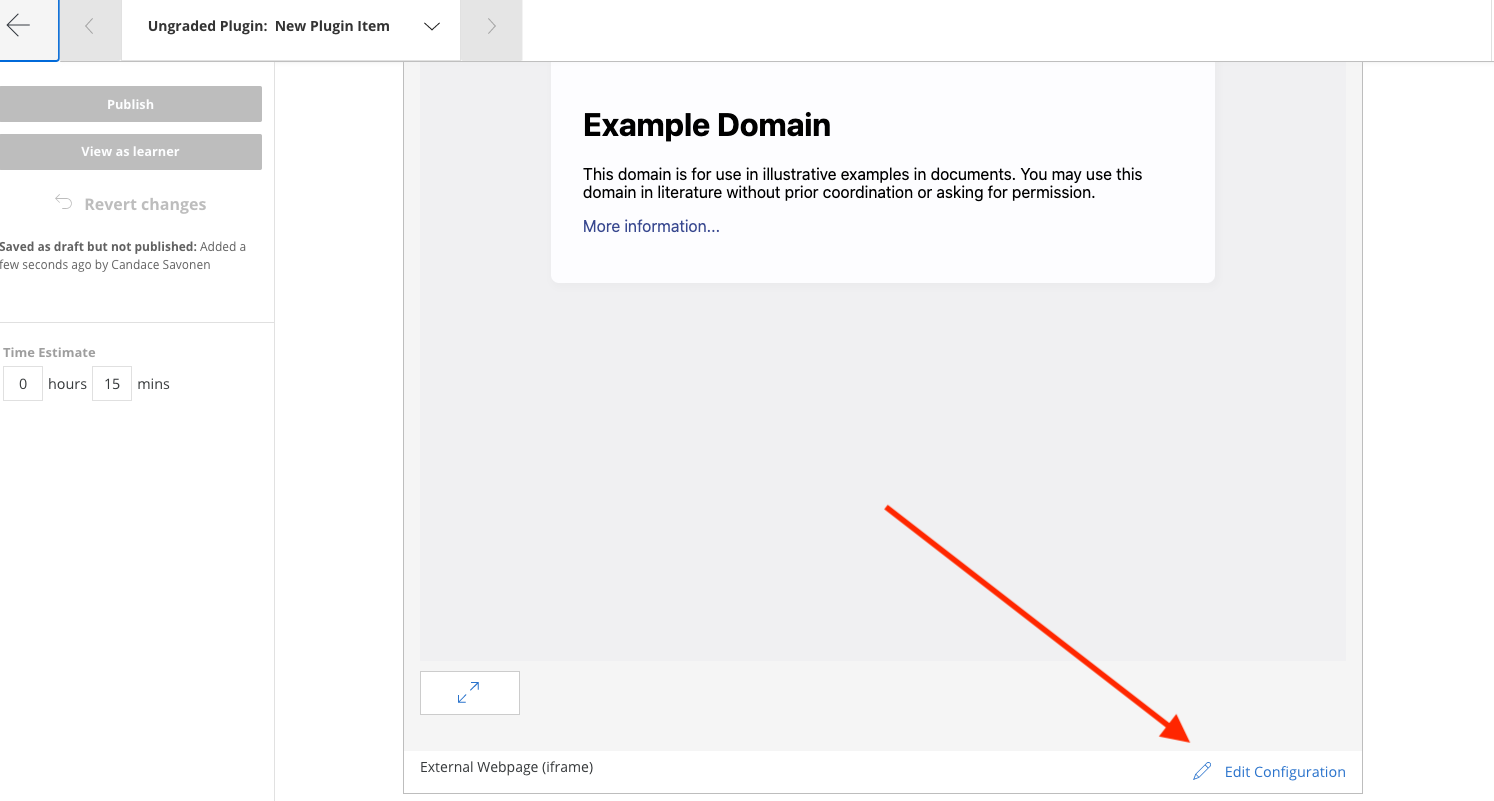
This will add an unpublished ungraded plugin to your lesson: 

Now click the edit button on the new Ungraded Plugin.

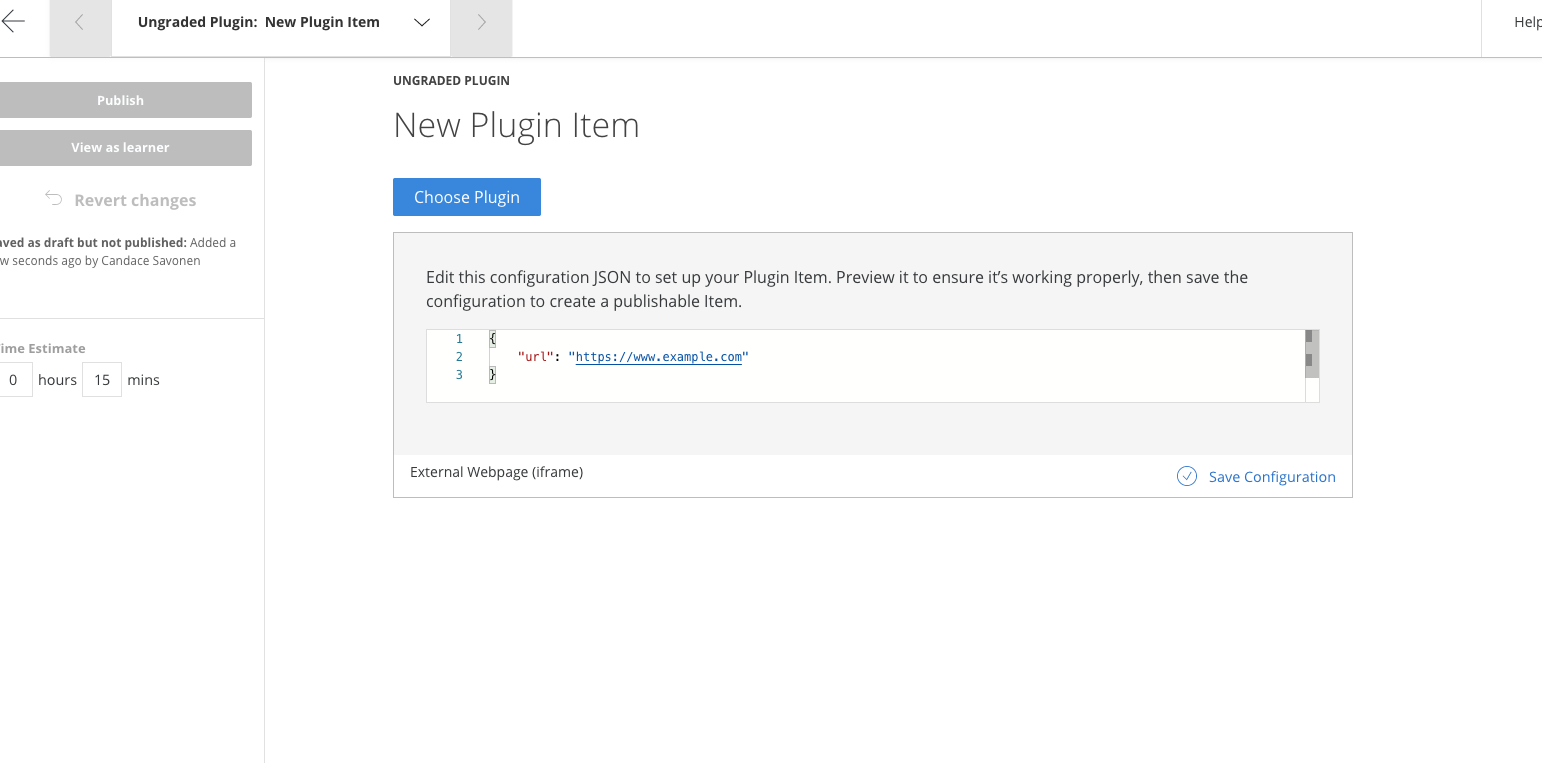


This will bring you to the standard example plug in page. 

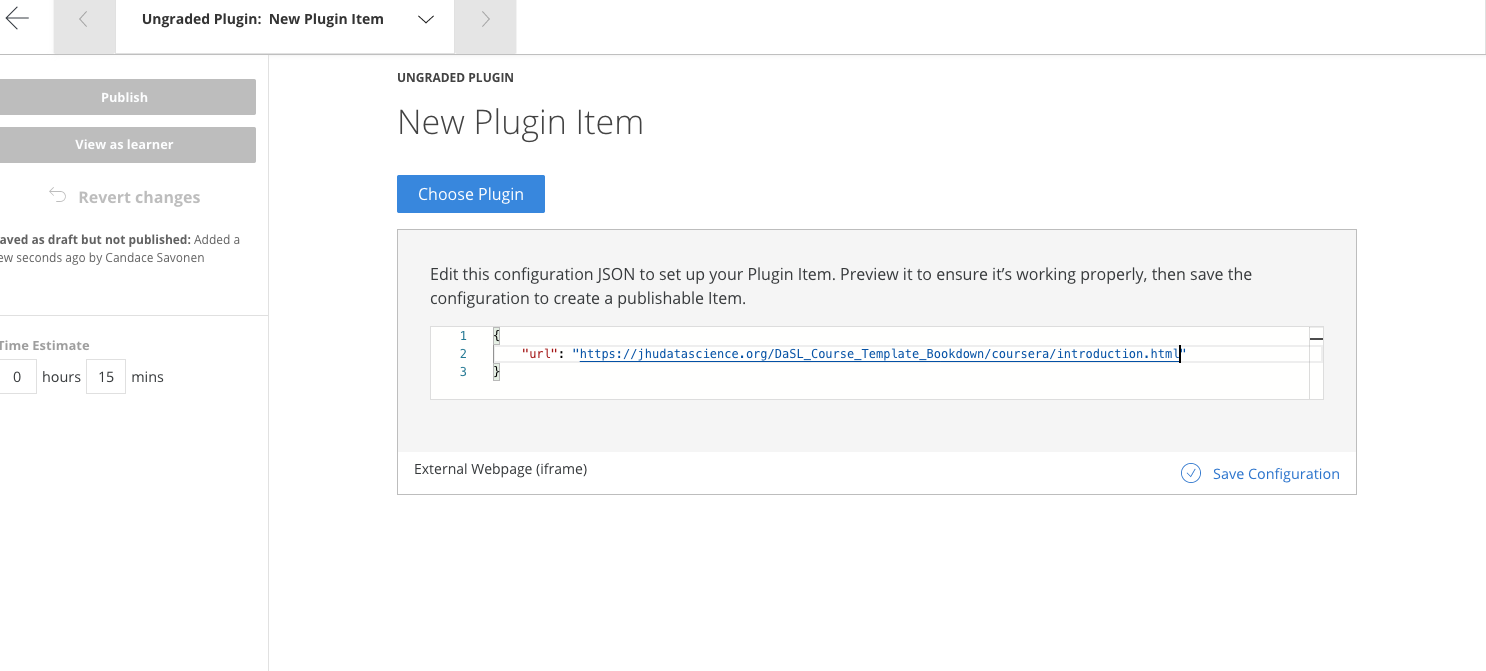
Scroll down and click Edit Configuration:



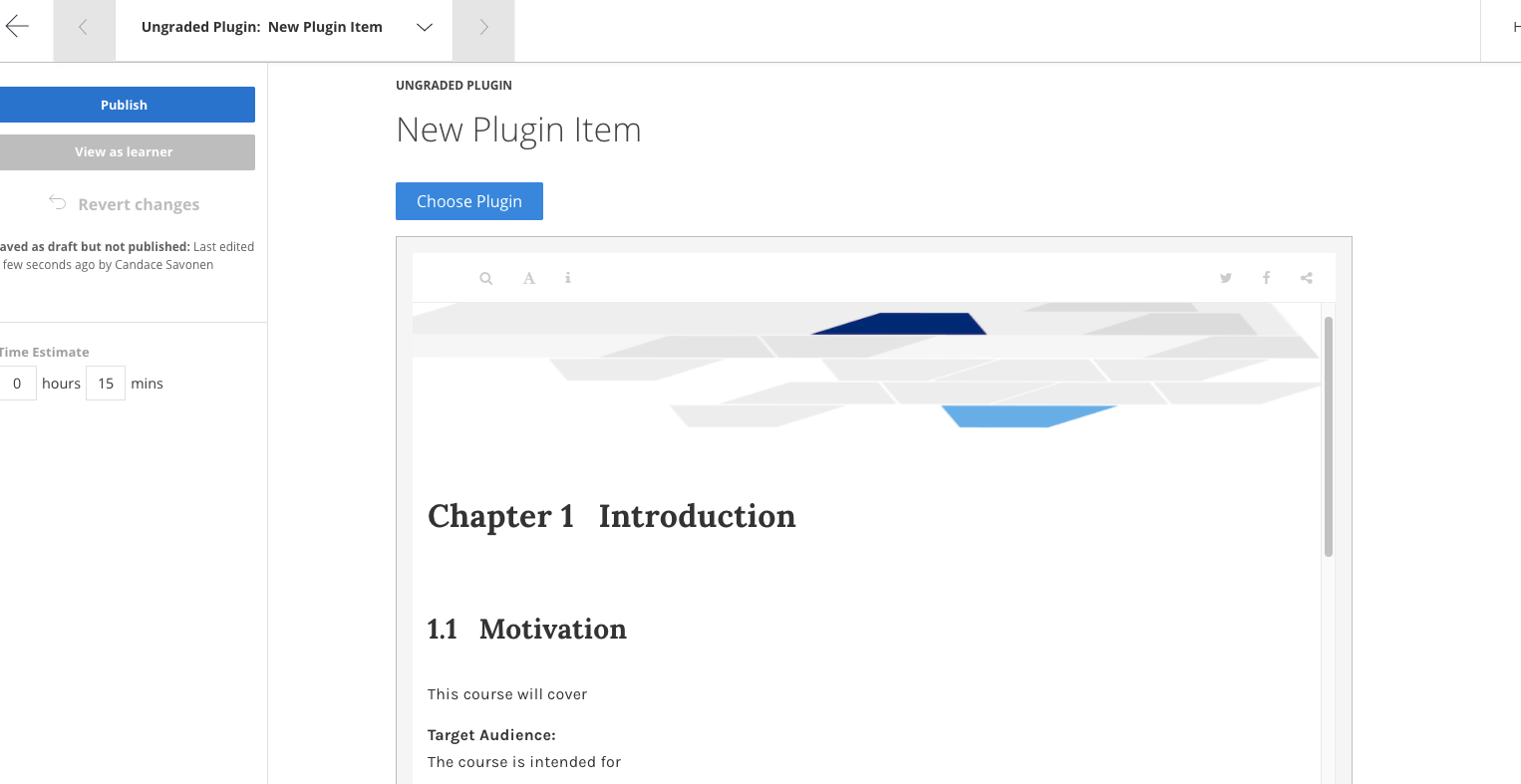
Replace the example url with the url of your individual chapter we discussed above.



Then click on Save Configuration:

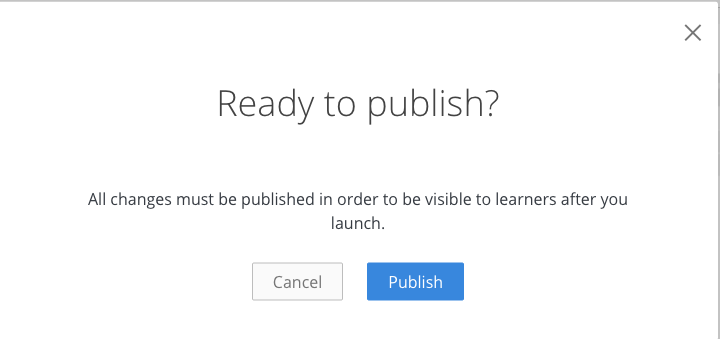


If the url you provided was correct, you should see a preview of your chapter:



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

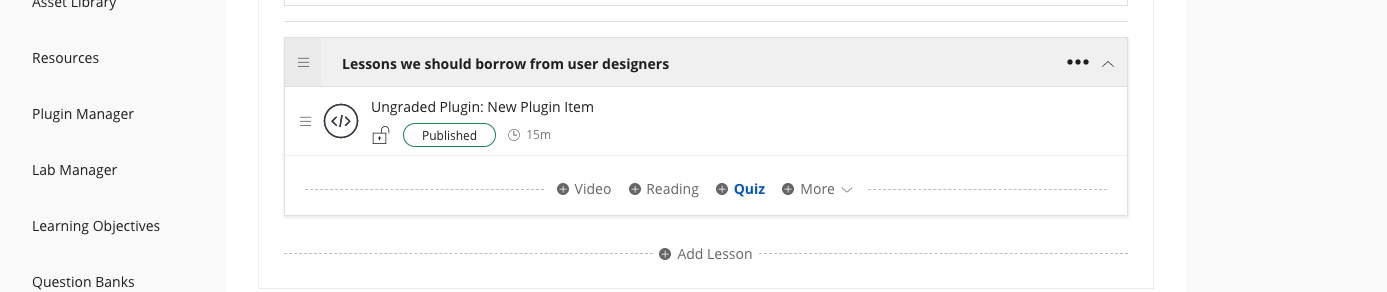
Now click Publish in the left upper corner (don’t worry it’s not actually publishing publicly, its just saving your changes). And it will ask you again to really scare you, but yes, click Publish.



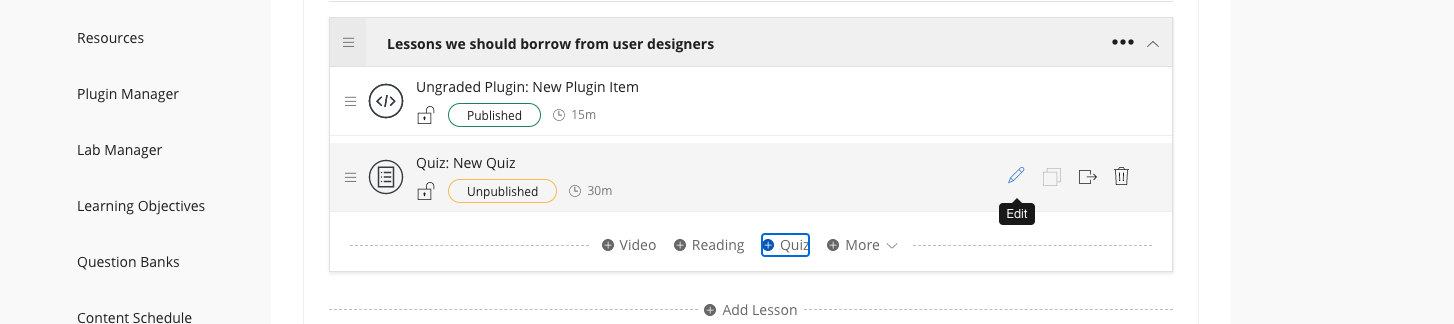
Congrats you added a chapter to your course! Now do the same thing for all your chapters!

### 14.3.2 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.

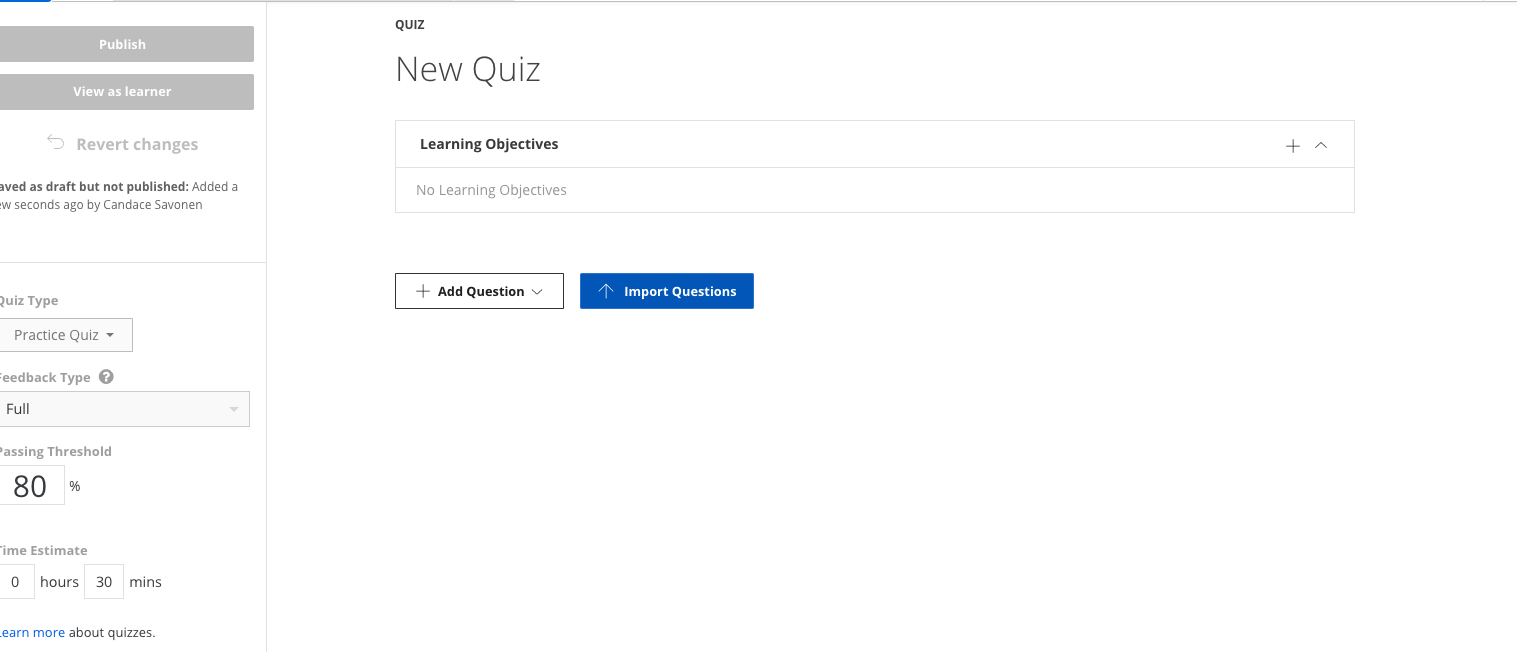


Now go to Edit on the new quiz.

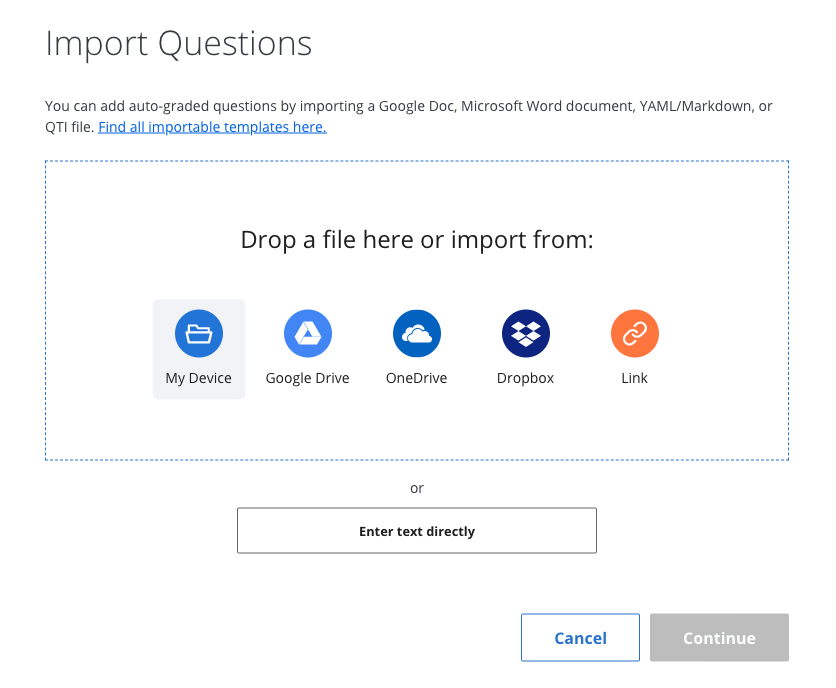


If you are choosing to manually enter your quizzes, you can continue to add each question from this page.

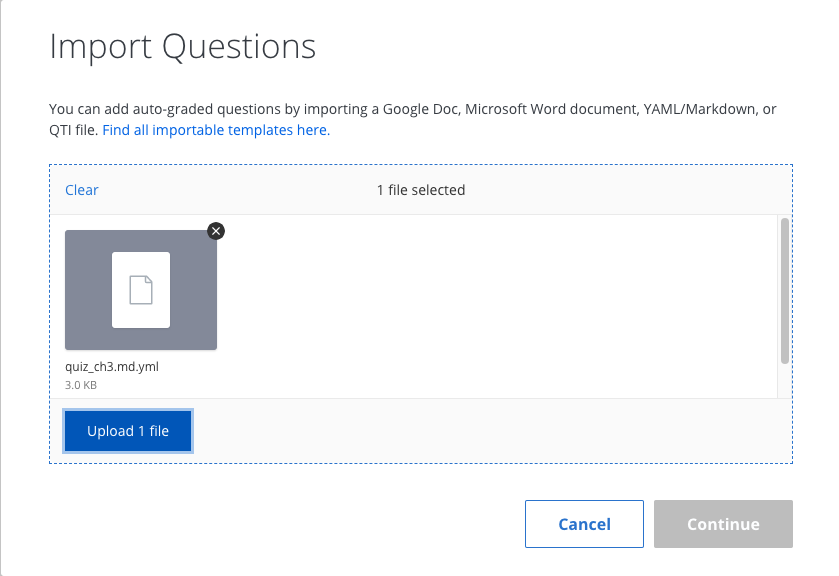
If you are importing a Leanpub converted quiz click the Import Questions button and choose your converted yml files. (If you’ve not yet converted your leanpub quizzes, then [refer to that section](#converting-quizzes-from-leanpub-format).)



Upload your Coursera quiz yml file from your device:



After you’ve selected the file click Upload:

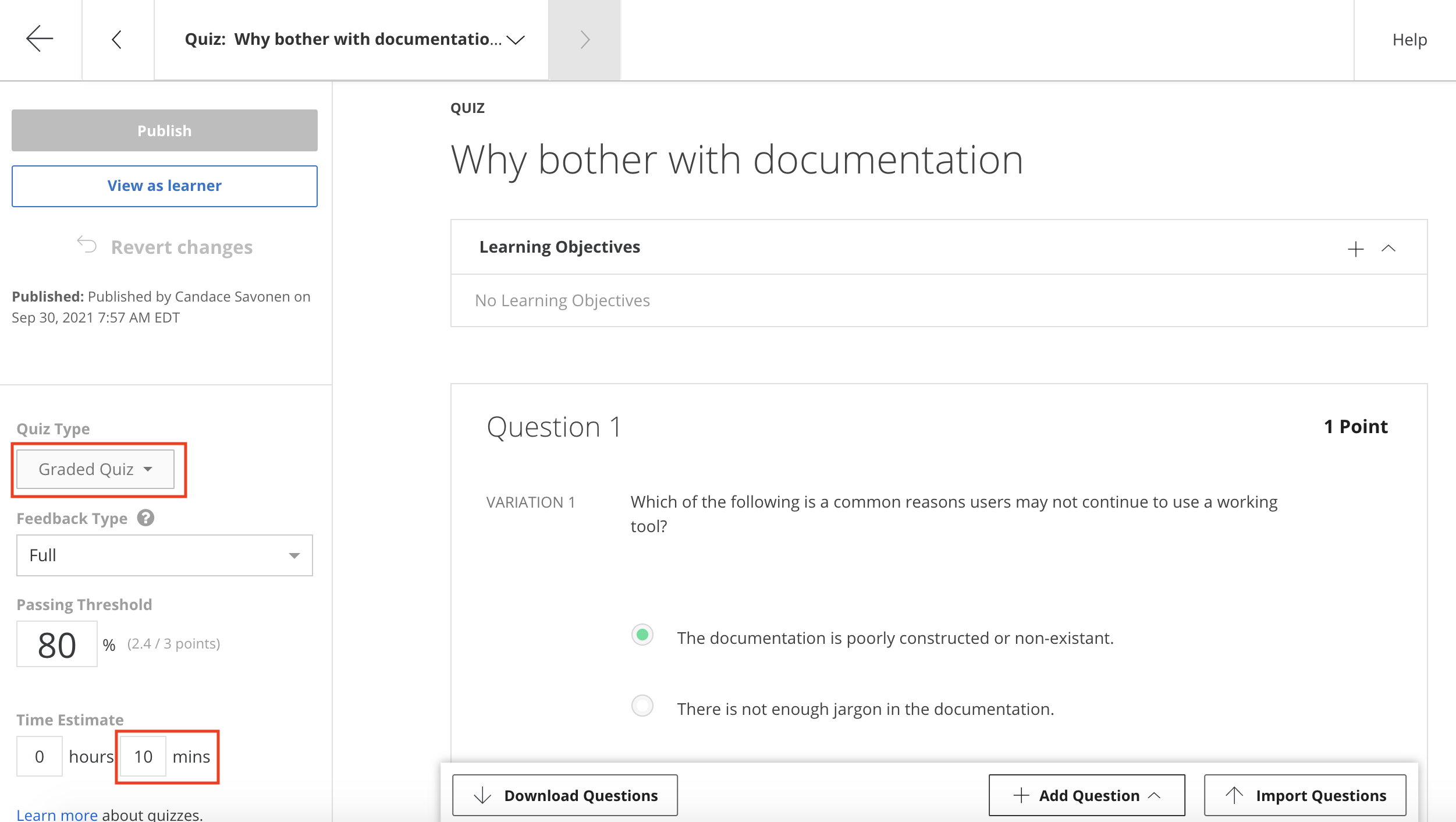


It will let you know if there are any problems uploading the quiz. If the coursera\_quiz\_conversion.R script failed to render something properly, please [file an issue here](https://github.com/jhudsl/OTTR_Quizzes/issues) describing how it failed and including a file that shows the instance it failed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

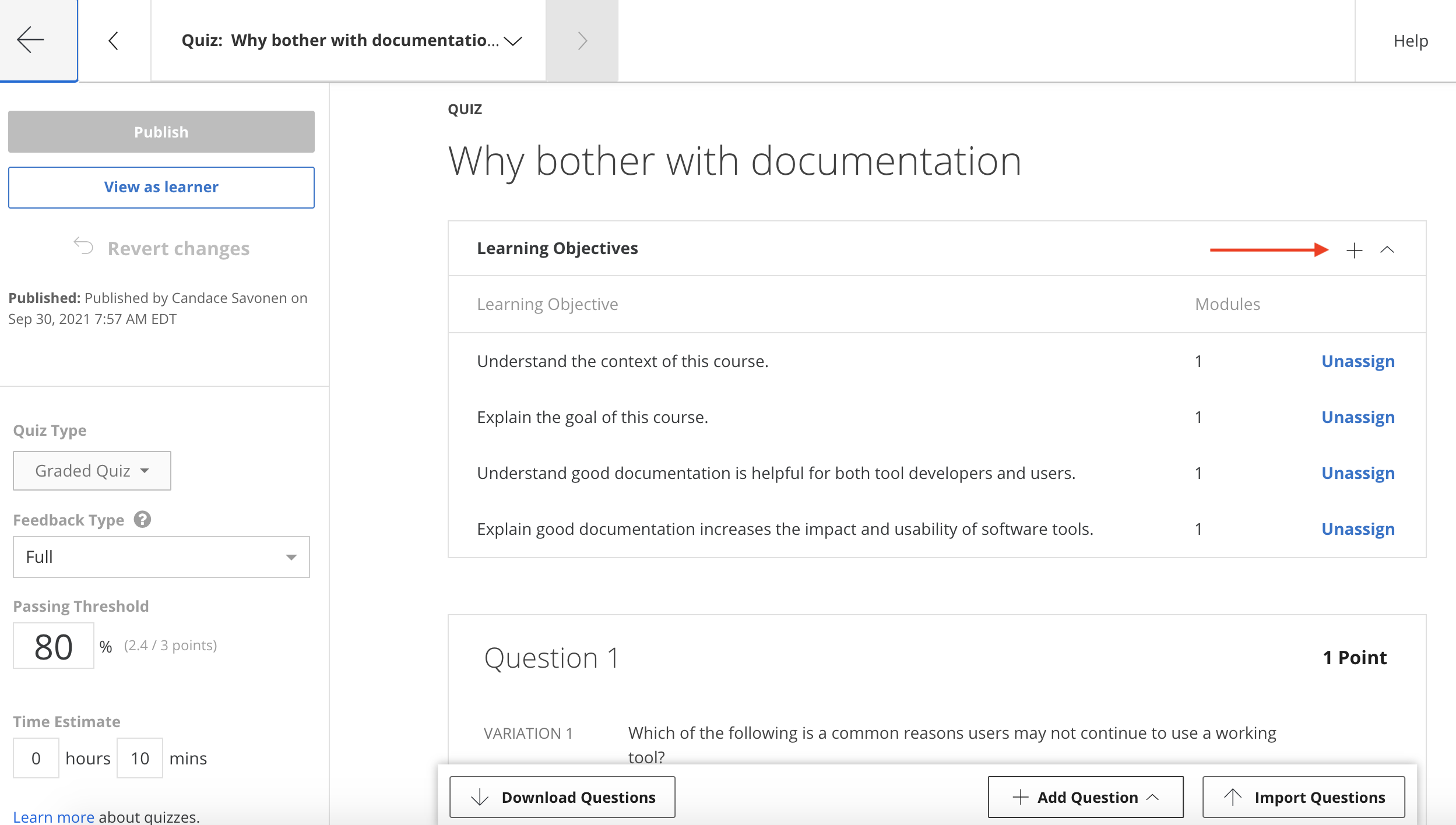
Note that images and links are not currently supported in that conversion script. You also cannot have : in your prompts or answers.

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

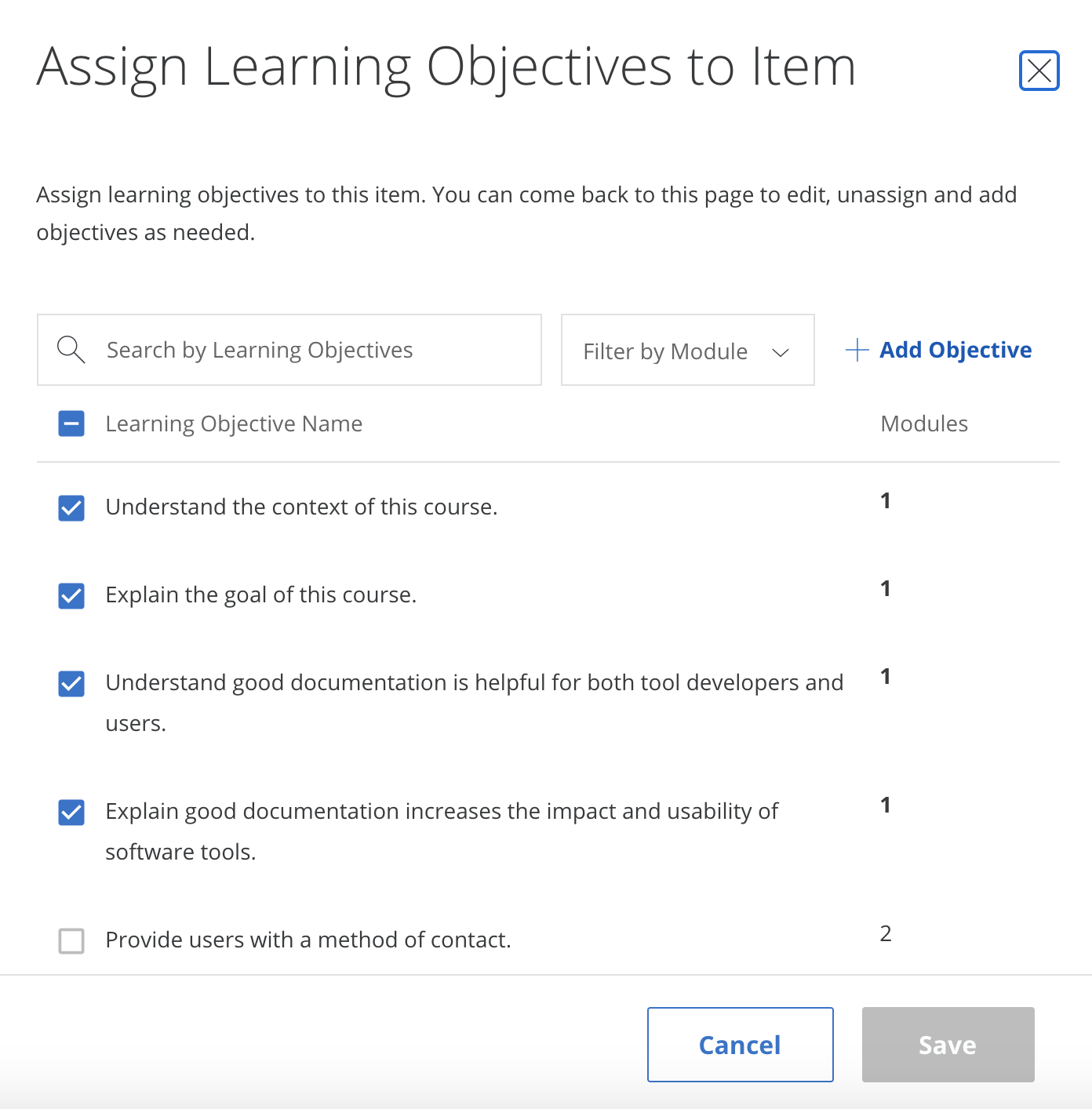
By default quizzes are set to Practice Quiz but you will need at least some of your quizzes to be used for Grading. On the left side of quiz editing page, there’s a drop down menu to select Graded Quiz. Also in this location you can edit the estimated time it should take to complete the quiz (by default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



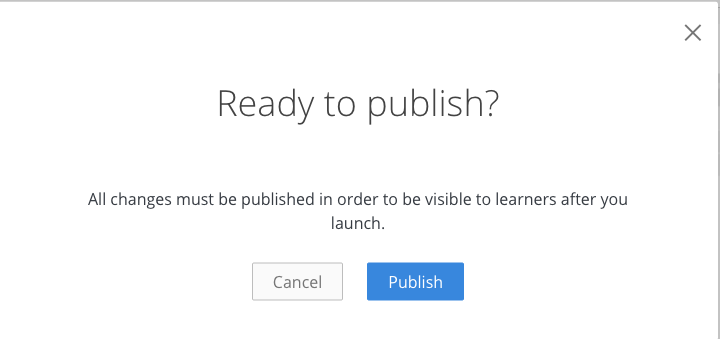
You also will want to assign the learning objectives that apply to this quiz. You can do this by clicking on the plus sign underneath the title.



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



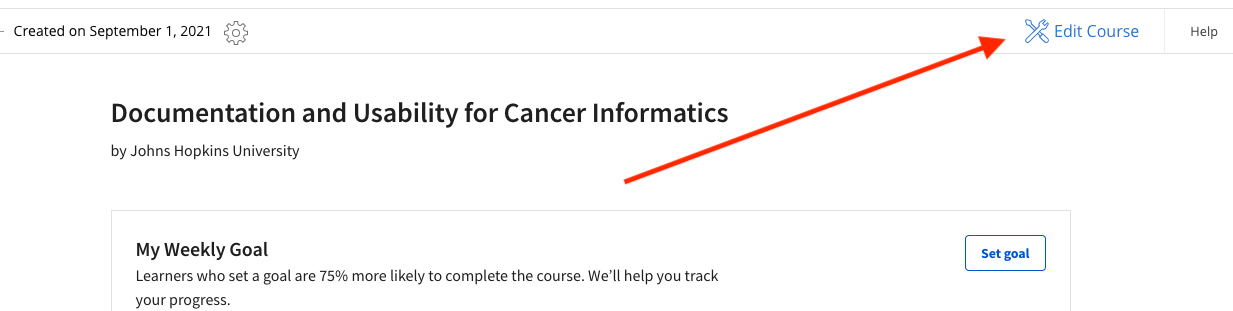
You can edit the title of your quiz and then click Publish (don’t worry it’s not actually publishing publicly, its just saving your changes). And it will ask you again to really scare you, but yes, click Publish.



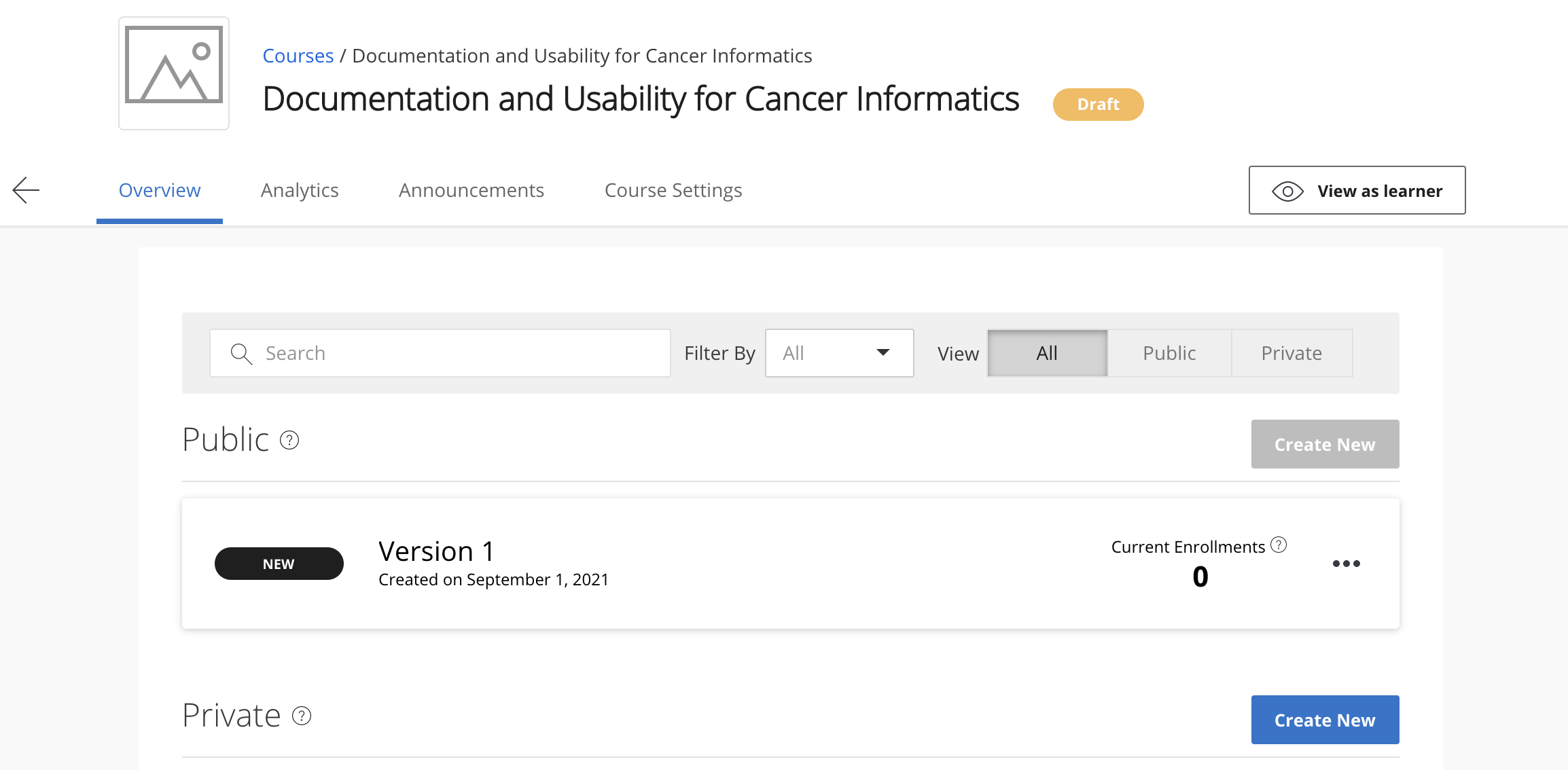
Congrats you added a quiz to your course!

## 14.4 Add programmed messages

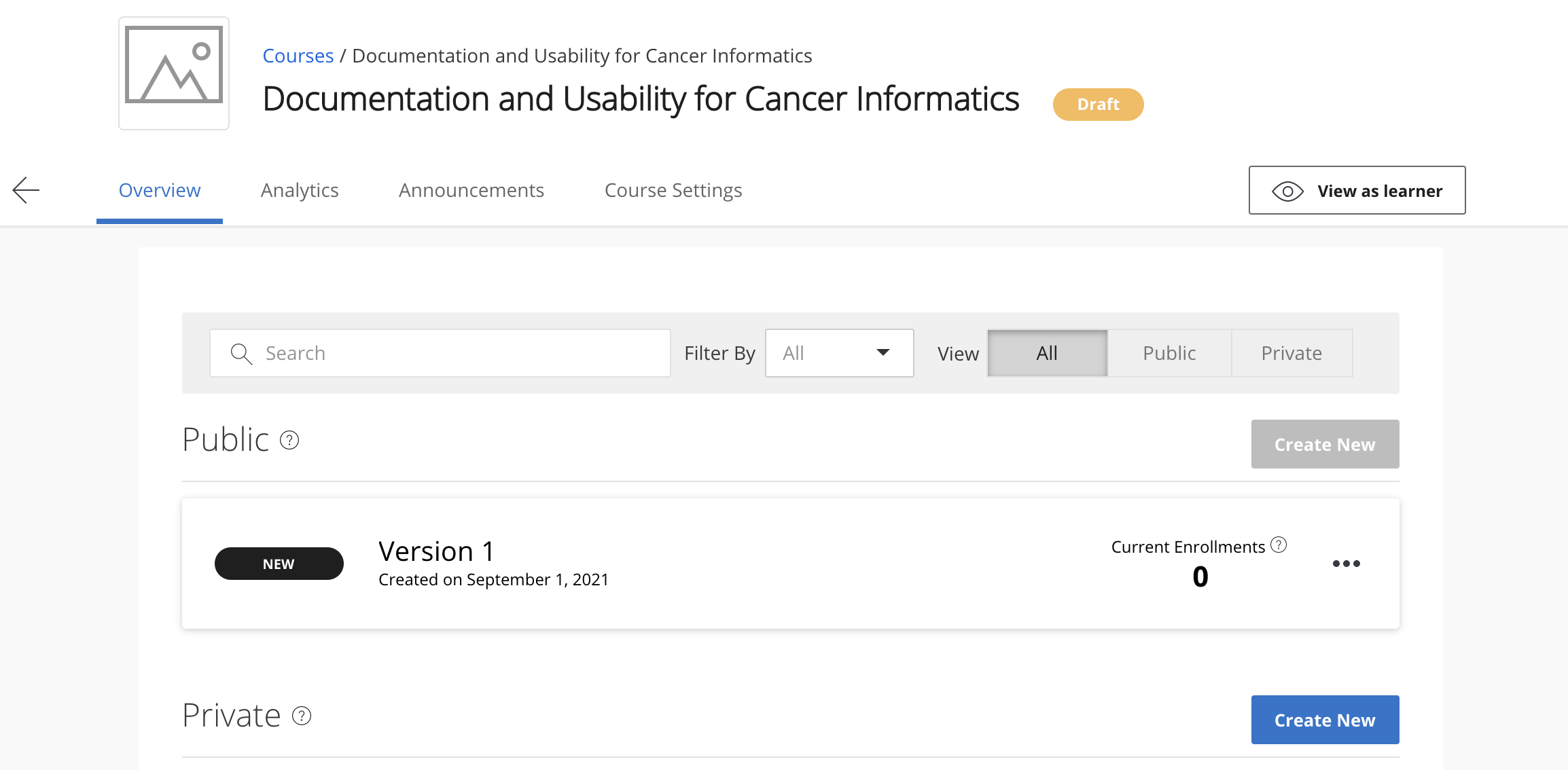
You can click on Go to Course. And on your main course page click Edit Course in the right corner.



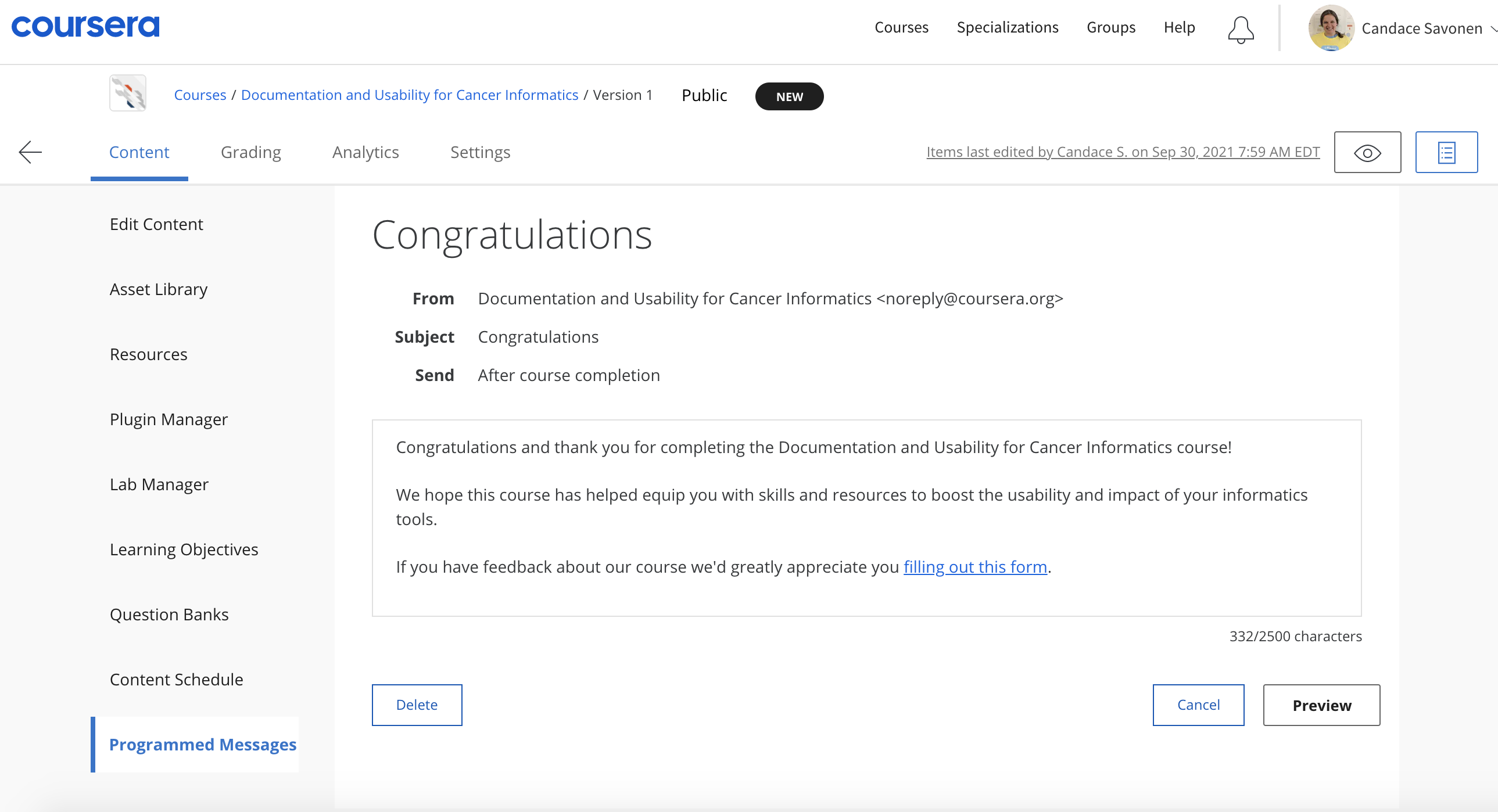
This will bring you to your version page.



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:

#### 14.4.0.1 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.

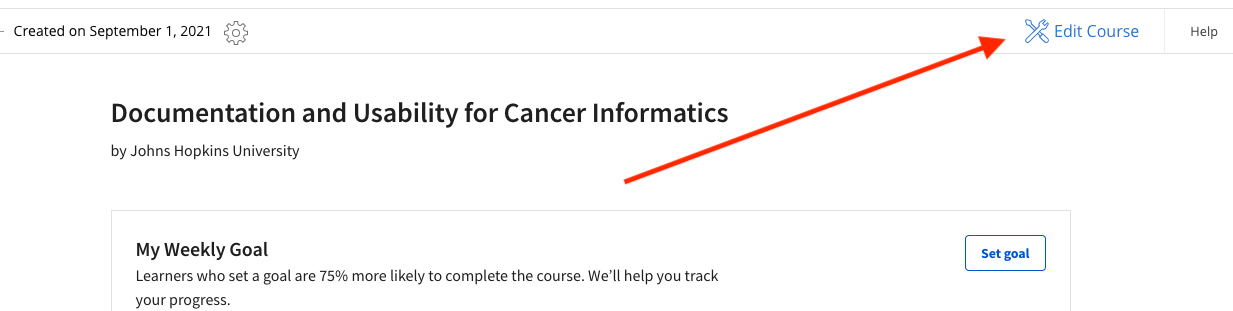
#### 14.4.0.2 Completion template message:

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.

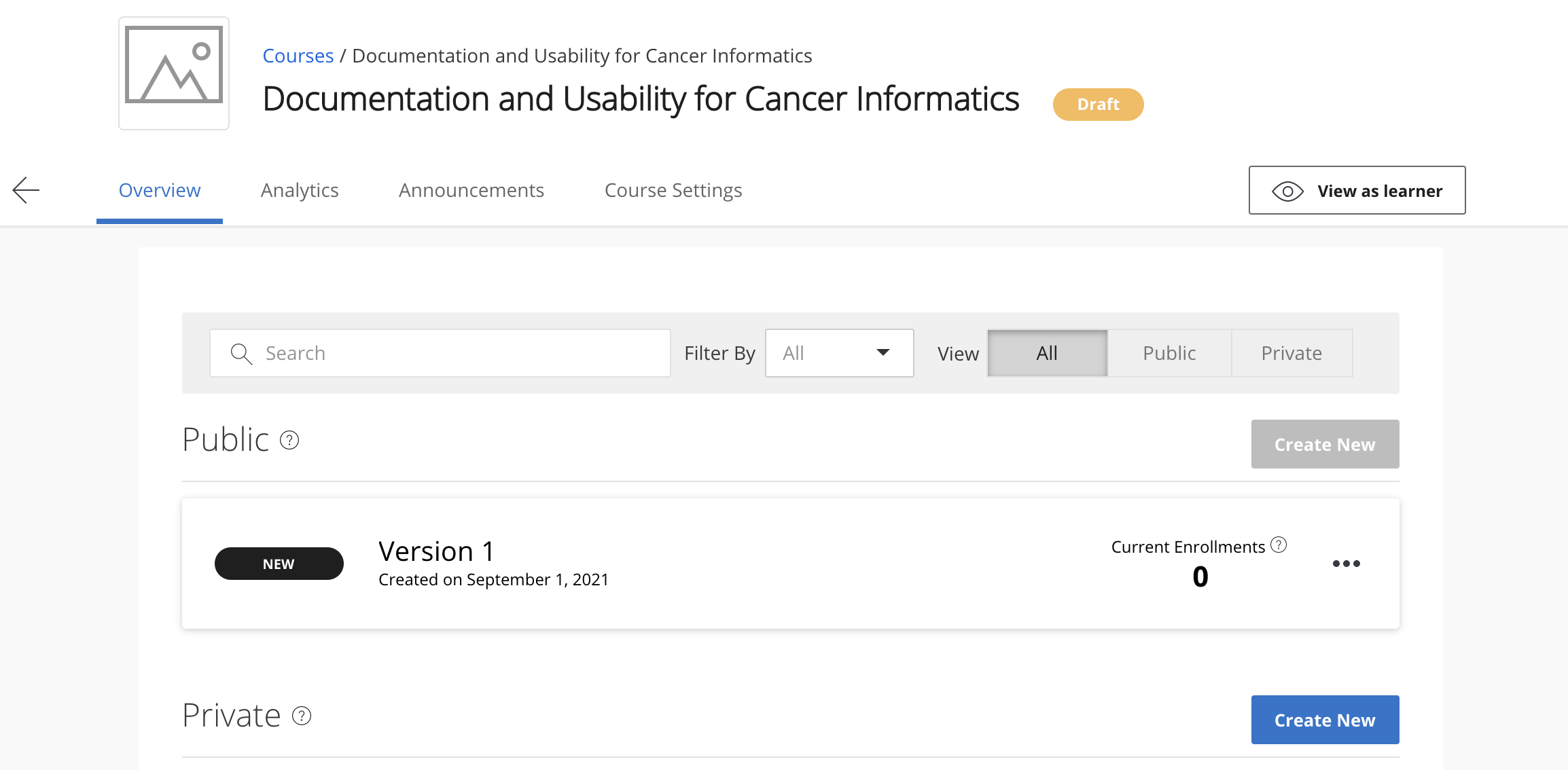
Add a link to your feedback form by clicking the link icon. Click Preview and then Publish to save your edits.

## 14.5 Grading formula

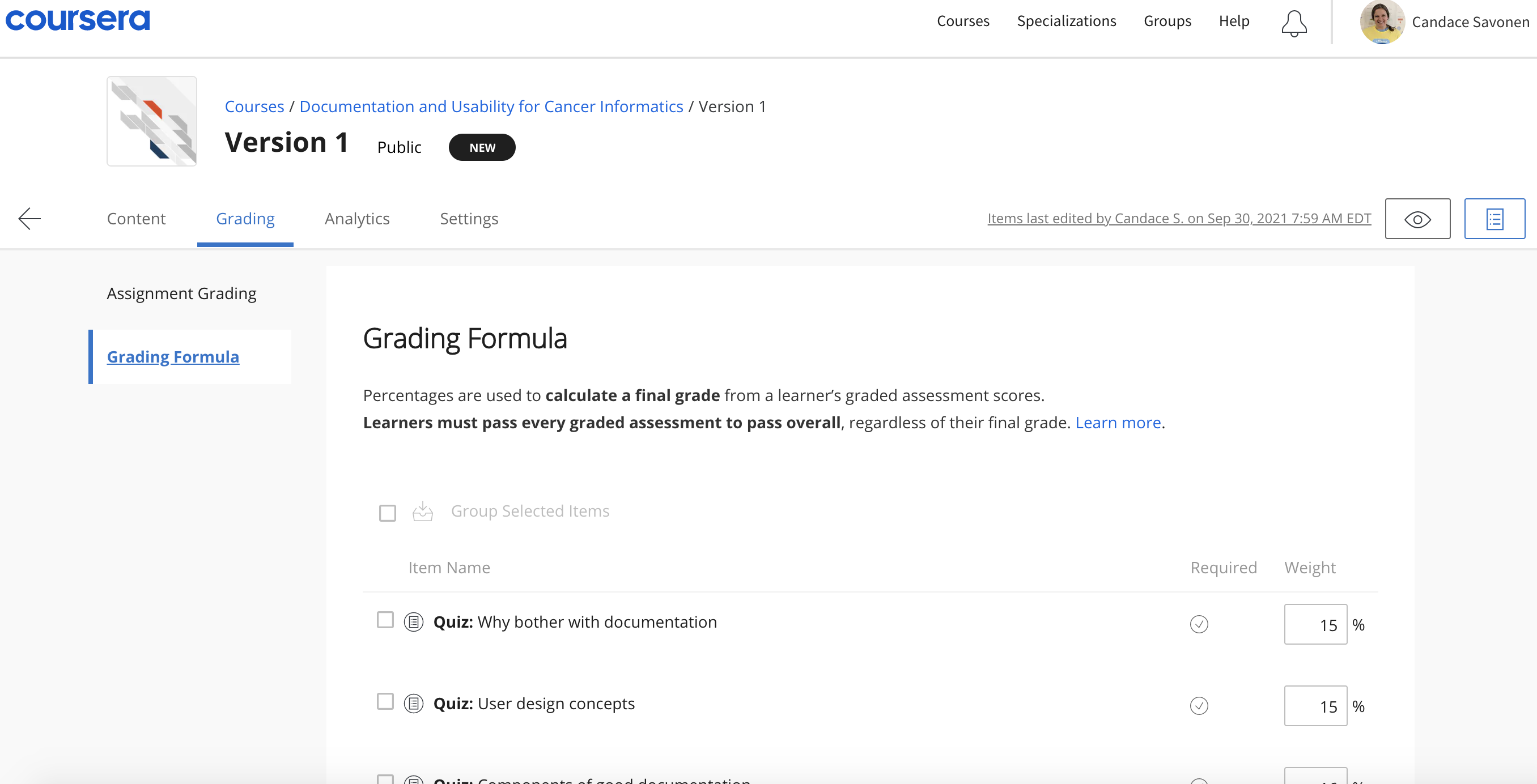
For final grades in the course, you will need to set the Grading Formula. To get to this page, you can click on Go to Course. And on your main course page click Edit Course in the right corner.



This will bring you to your version page.



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



## 14.6 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.

## 14.7 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.

## 14.8 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

# 15 Automation info and set up

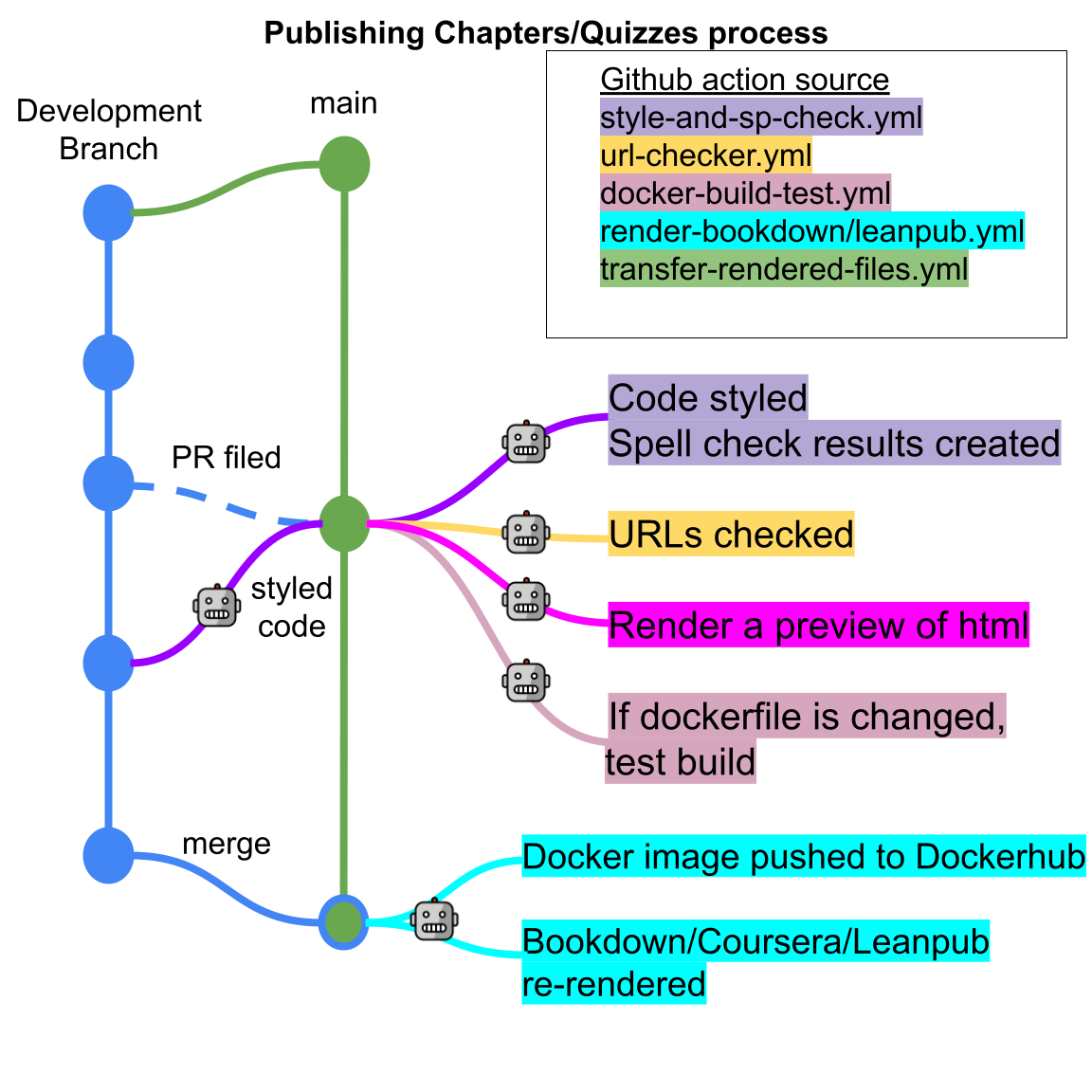
# 16 Automation info and set up

# 17 How to set up and customize GitHub actions robots

Here’s a summary of the Github actions set up in this repository.

Note: if you are not a part of jhudsl organization, you will need to set follow these instructions to set up your GitHub secrets before these actions will work.

In summary, here’s what the GitHub actions do for you (when set up): - Re-render Bookdown after merging to main - Create a preview of the rendered version with changes you’ve made on a pull request - Re convert files for Coursera - Spell check - Fix code styling - Checks for broken URLs - Copy over Leanpub-needed files to the Leanpub repository - Re-render Leanpub files (on the other repository) - Checks if Docker image changes successfully build - Push changed Docker images to Dockerhub

Here’s a diagram to summarize: 

These Github actions also work across repositories to support converting Bookdown content into formats ready for publishing on Coursera or Leanpub.

If you are only looking to use this template for creating a Bookdown course, this diagram is not as pertinent.

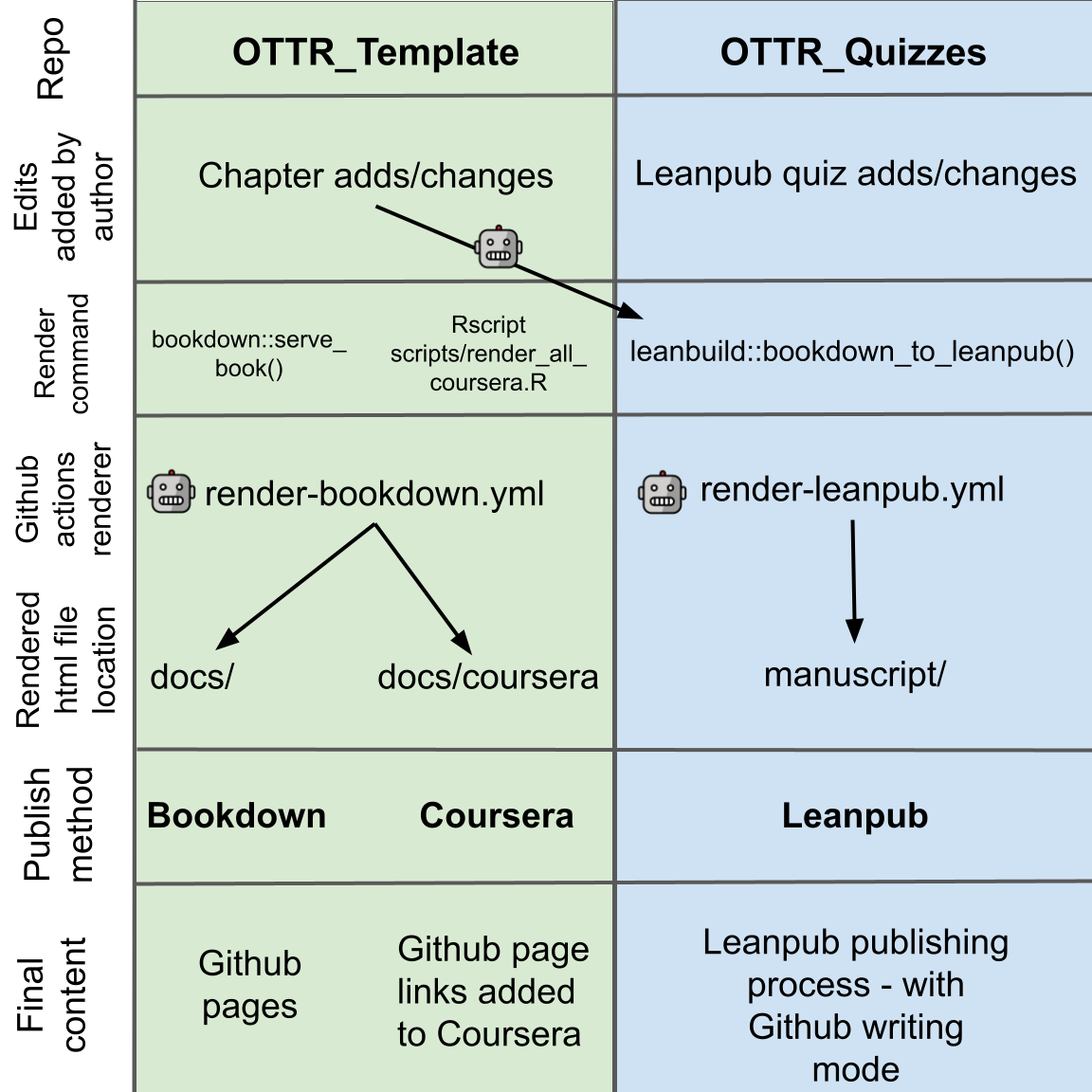


Figure : Summary of publishing process

### 17.0.1 Preview of render

After you open a pull request, a preview of the renders as they will appear after the pull request is accepted is run and linked to in a comment on the pull request. Upon each commit these previews will re-render and edit the comment with the time of the latest render. These Github Actions are located in [render-preview.yml](https://github.com/jhudsl/OTTR_Template/tree/main/.github/workflows/render-preview.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.

### 17.0.2 Spell check

Github actions will automatically [run a spell check on all Rmds](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. 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 2 spelling errors and you’ll need to resolve those before being able to merge your pull request.

To resolve those spelling errors, go to this repository’s Actions tab. Then, click on the GitHub action from the PR you just submitted. Scroll all the way down to Artifacts and click spell-check-results. This will download a zip file with a TSV that lists all the spelling errors.

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.

### 17.0.3 Code styling

Github actions will run the [styler package to all style R in all Rmds](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. Style changes will automatically be committed back to your branch.

### 17.0.4 URL Checking

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 GitHub actions file: .github/workflows/url-checker.yml and add the imposter URL on to the exclude-urls: argument with a comma in between.

### 17.0.5 Customizing render-bookdown.yml

If you plan on doing a lot of customizing with GitHub actions or would like to become more familiar, we recommend reading [this article](https://itnext.io/getting-started-with-github-actions-fe94167dbc6d) to get your feet wet. Then [this reference guide](https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions) in the GitHub actions docs is super useful.

Note that build-all and docker-build-test are not something we recommend requiring for status checks because docker-build-test is only run if there are changes to the Dockerfile and build-all is only run upon the acceptance and merging of a pull request.

However for simplicity purposes there are two sections this Github action that can you keep off if you won’t be making changes to the Docker image or you won’t be wanting it to sync to Google Slides automatically.

### 17.0.6 For a course that needs to publish to the Leanpub repository

transfer-rendered-files.yml is a Github action that will copy over the output docs/ files rendered by Bookdown to a parallel Leanpub repository.

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.

There are two edits to [.github/workflow/transfer-rendered-files.yml](https://github.com/jhudsl/OTTR_Template/blob/main/.github/workflows/transfer-rendered-files.yml) that need to be done to turn on the automatic copying of files between these repos:

1. Change line 28 to the repository name you would like the docs/ files to be transferred to.

repository: jhudsl/Course\_Template\_Quizzes

1. Uncomment lines 13-18 in this file:

# Only run after the render finishes running  
#workflow\_run:  
# workflows: [ "Build, Render, and Push" ]  
# branches: [ main ]  
# types:  
# - completed

### 17.0.7 For a course that needs to publish to Coursera

The [render-bookdown.yml](https://github.com/jhudsl/OTTR_Template/blob/main/.github/workflows/render-bookdown.yml) github action the chapter content for Coursera by using this command within the docker image:

leanbuild::render\_coursera()

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 Leanpub companion repository has a [Leanpub -> Coursera quiz conversion script](https://github.com/jhudsl/OTTR_Quizzes/blob/main/scripts/coursera_quiz_conversion.R) if you choose to create quizzes and publish on both Leanpub and Coursera.

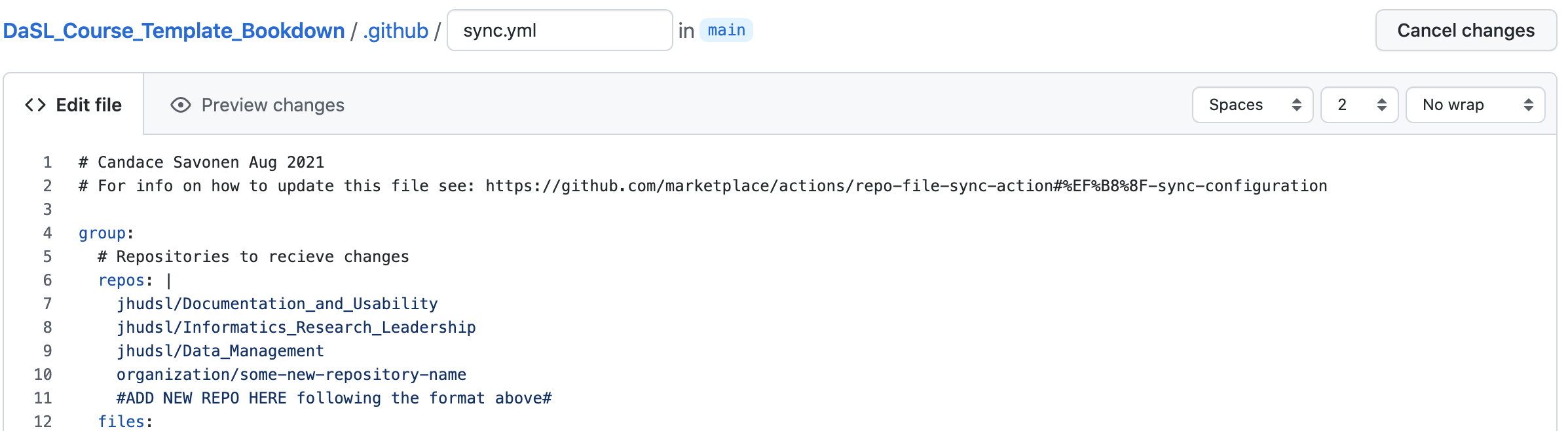
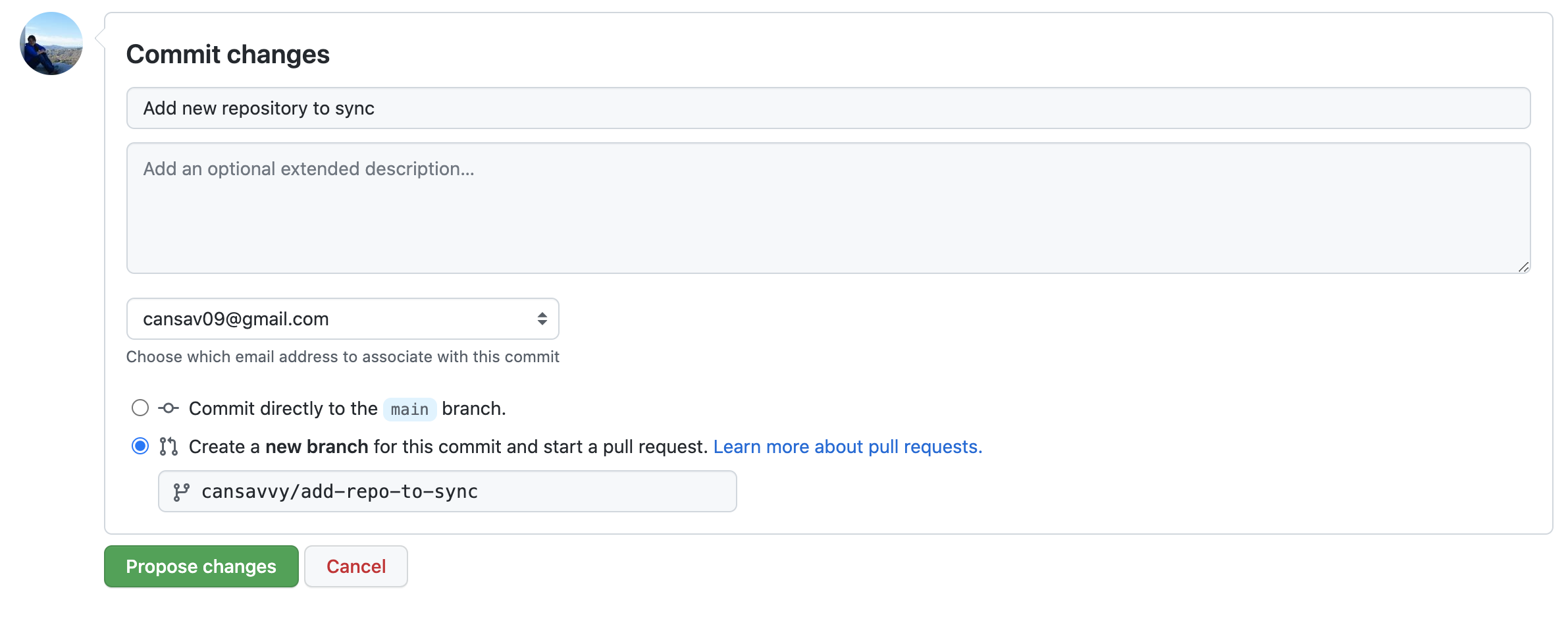
# 18 Receiving template updates

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.

### 18.0.1 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.*

### 18.0.2 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.

# 19 Finishing touches

# 20 Finishing touches

## 20.1 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.

### 20.1.1 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.

### 20.1.2 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>

### 20.1.3 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).

# 21 Changing title and style

### 21.0.1 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.

## 21.1 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.

### 21.1.1 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.

## 21.2 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.

### 21.2.1 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: "December, 2021"  
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.

### 21.2.2 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.

## 21.3 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 {-}

## 21.4 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.

## 21.5 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.

# 22 How to give 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:

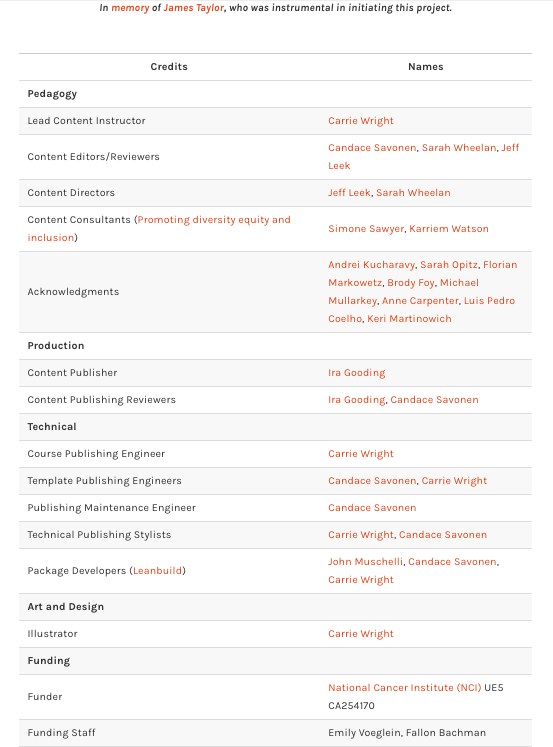


Figure : Credit section from a course that used this template

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[Leanbuild]|[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 -->  
  
[Leanbuild]: https://github.com/jhudsl/leanbuild

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

### 22.0.1 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.

### 22.0.2 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.

# 23 Adding Google analytics

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>

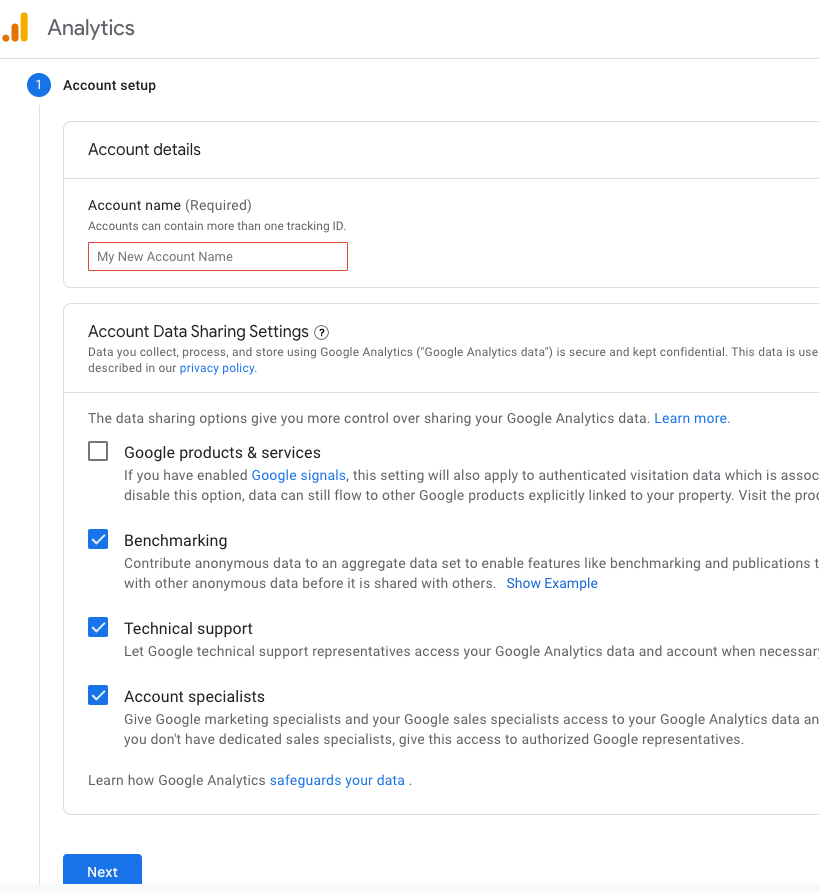


Figure : Google Analytics account setup

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

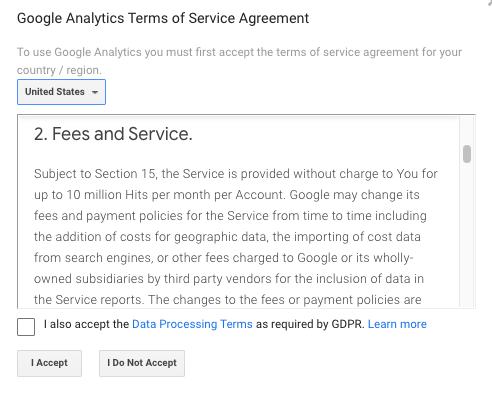


Figure : Google Analytics terms

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)

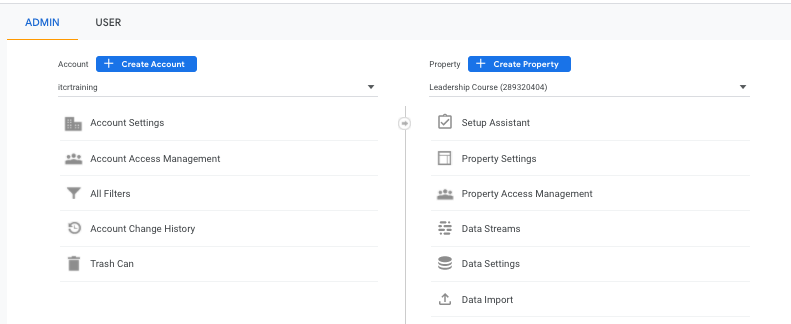


Figure : Google Analytics Property creation

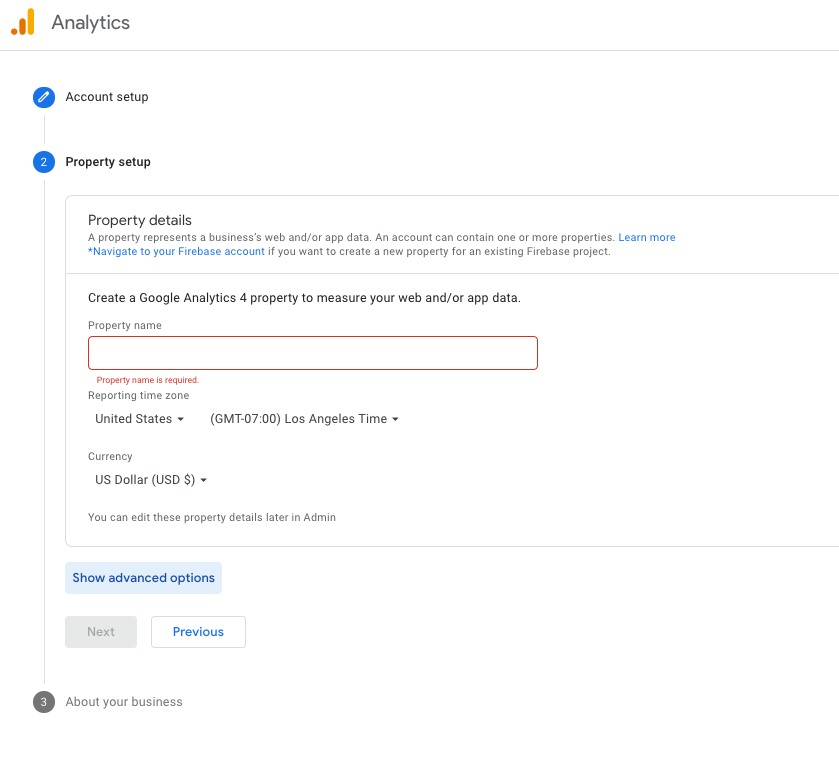


Figure : Google Analytics Property

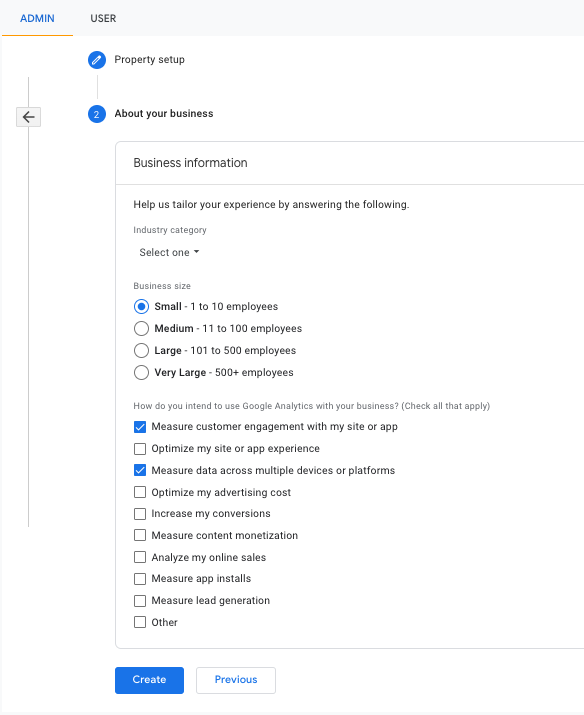


Figure : Google Analytics Property choices

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

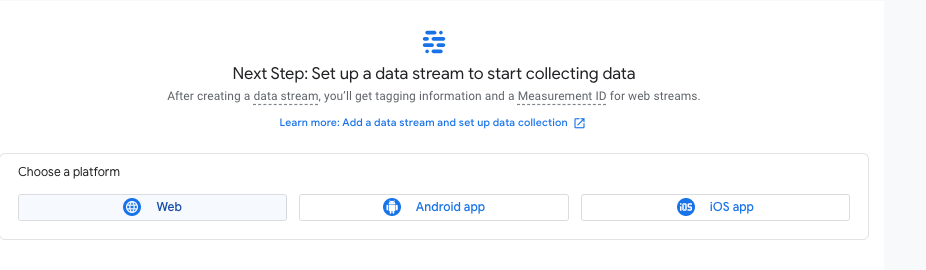


Figure : Google Analytics stream options

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)

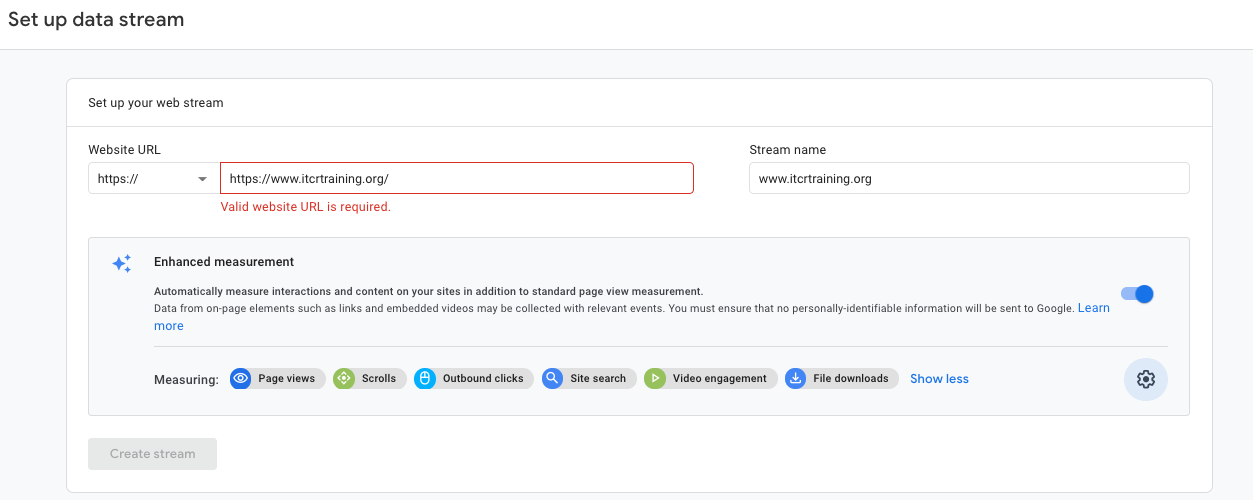


Figure : Google Analytics stream

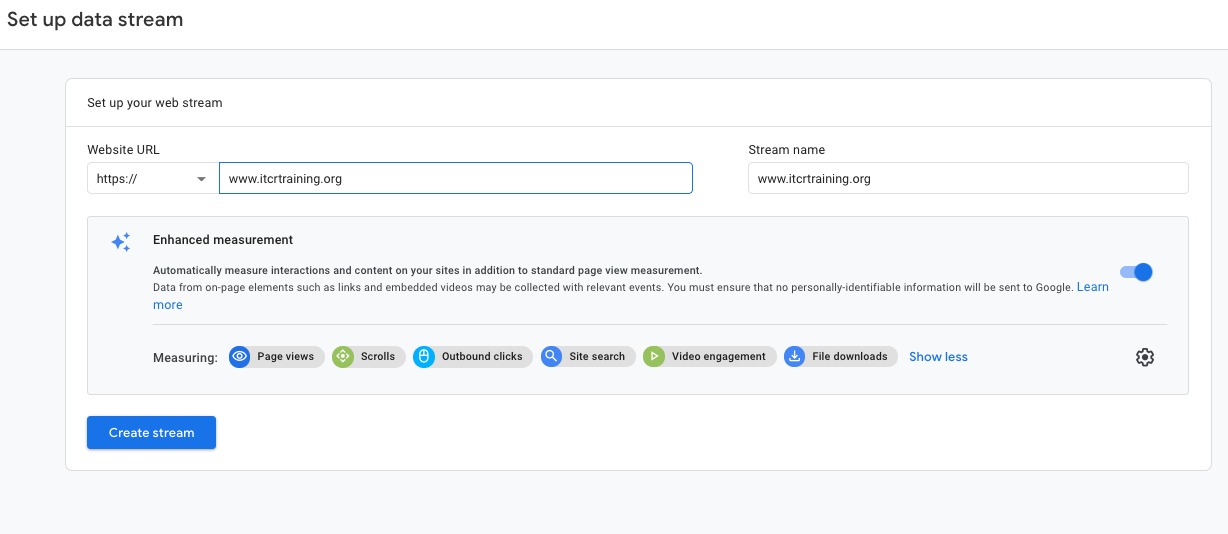
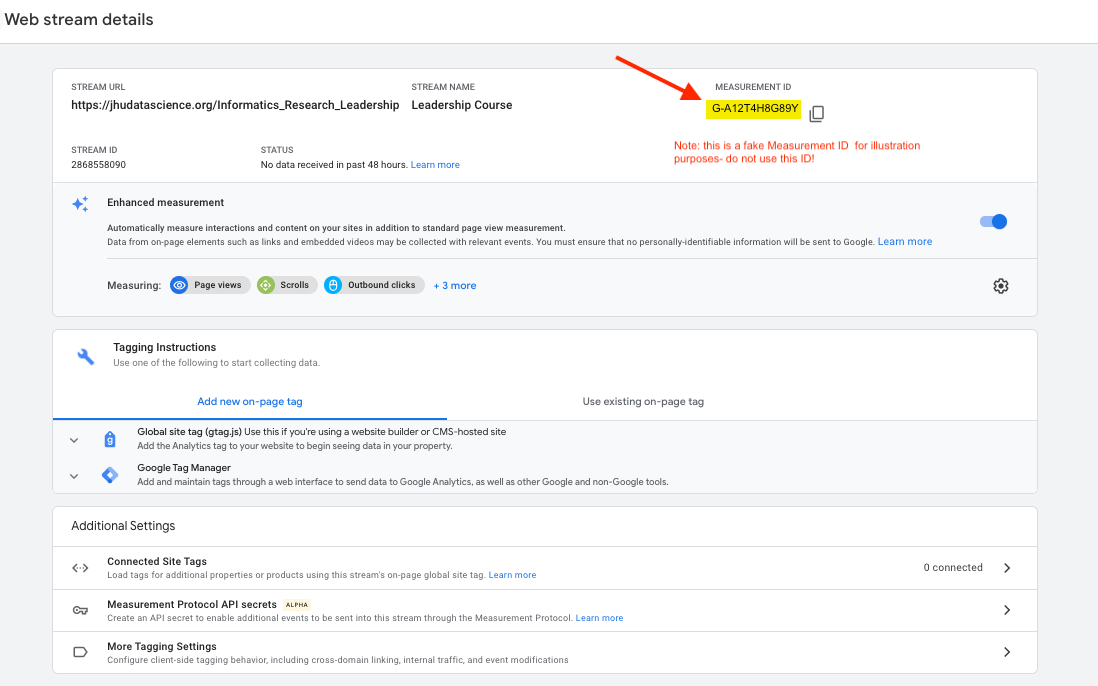
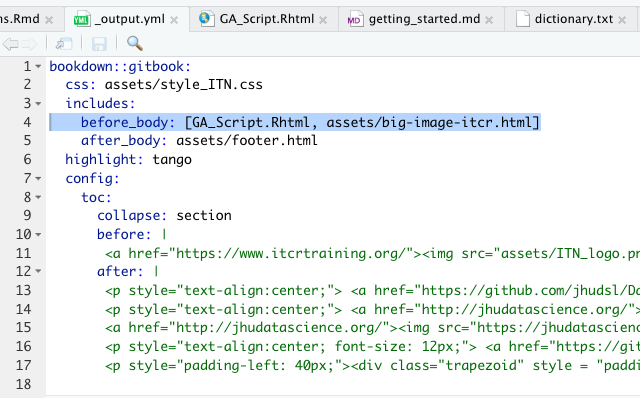


Figure : Google Analytics stream

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.

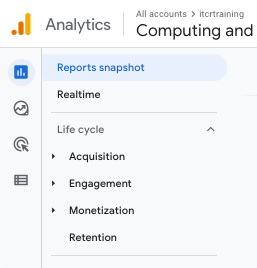


Figure : Google Analytics Reports

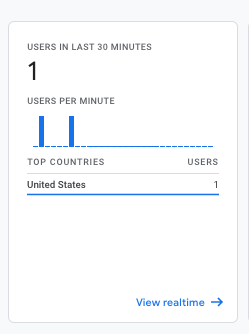


Figure : Check Google Analytics

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

# 24 Release a course for public viewing

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).

### 24.0.1 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 ([Leanbuild](https://github.com/jhudsl/leanbuild)) | [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 2021-12-19   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] RSPM (R 4.0.3)   
## backports 1.1.10 2020-09-15 [1] RSPM (R 4.0.2)   
## bookdown 0.24 2021-12-19 [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 1.6.0 2021-12-17 [1] CRAN (R 4.0.2)   
## htmltools 0.5.0 2020-06-16 [1] RSPM (R 4.0.1)   
## knitr 1.33 2021-12-19 [1] Github (yihui/knitr@a1052d1)   
## lifecycle 1.0.0 2021-02-15 [1] CRAN (R 4.0.2)   
## magrittr 1.5 2014-11-22 [1] RSPM (R 4.0.0)   
## 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 2021-12-19 [1] Github (r-lib/rlang@f0c9be5)   
## rmarkdown 2.10 2021-12-19 [1] Github (rstudio/rmarkdown@02d3c25)  
## rprojroot 1.3-2 2018-01-03 [1] RSPM (R 4.0.0)   
## 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 2021-12-19 [1] Github (R-lib/testthat@e99155a)   
## usethis 2.1.5.9000 2021-12-19 [1] Github (r-lib/usethis@6c2e204)   
## withr 2.3.0 2020-09-22 [1] RSPM (R 4.0.2)   
## xfun 0.26 2021-12-19 [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

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2021. *Rmarkdown: Dynamic Documents for r*. <https://github.com/rstudio/rmarkdown>.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. <https://bookdown.org/yihui/rmarkdown>.