WDL Workflows

November 15, 2023

Table of Contents

knitr::opts\_chunk$set(out.width = "100%")

# Overview

This book introduces WDL Workflows on AnVIL-powered-by-Terra. After introducing several concepts, including basic WDL syntax, we present hands-on exercises to run a workflow, write a WDL, localize a file, customize a Docker image, and join the Discourse. No local software installation is required as each exercise leverages web-based resources.



## Skills Level

*Genetics*  
**Novice**: No genetics knowledge needed

*Programming skills*  
**Novice**: No programming experience needed

## Learning Objectives

* Understand when WDL Workflows are the right tool
* Run a Workflow on AnVIL-powered-by-Terra
* Write a WDL using Broad Methods Repository
* Bring your own data to analyze
* Customize your Docker environment
* Join the conversation

## Timing

It should take approximately 1.5-2 hours to complete this course, including hands-on exercises.

## Branding

Due to differences in branding, some of the screenshots shown in this course may include different logos and colors than what you see on Terra. However, the same functions and menus are available regardless of whether you’re using AnVIL-powered-by-Terra, classic Terra, or another version of the platform.

## AnVIL Collection

Please check out our full collection of AnVIL and related resources: <https://hutchdatascience.org/AnVIL_Collection/>

# 1 Review Workflows on AnVIL-powered-by-Terra

This **review** chapter introduces several basic concepts for working with WDL Workflows. The first part highlights considerations when using Docker container technology and provides an overview of the main sections in an example WDL workflow. The second part reviews how to run a basic workflow on AnVIL-powered-by-Terra.

You can skip to Chapter 2 if you are already familiar with these topics (e.g., if you’ve successfully completed the “[Intro to workflows on Terra](https://leanpub.com/universities/courses/terra/pipelining)” Leanpub course or have run a workflow before).

**Learning Objectives**

1. Identify the four parts of WDL workflows on AnVIL-powered-by-Terra
2. Understand how Docker improves reproducibility
3. Overview of basic WDL syntax
4. Know how to find and configure Workflows
5. Import a WDL by cloning a Workspace
6. Run a workflow using AnVIL-powered-by-Terra
7. Examine a workflow’s output

## 1.1 Optional: What is a workflow?

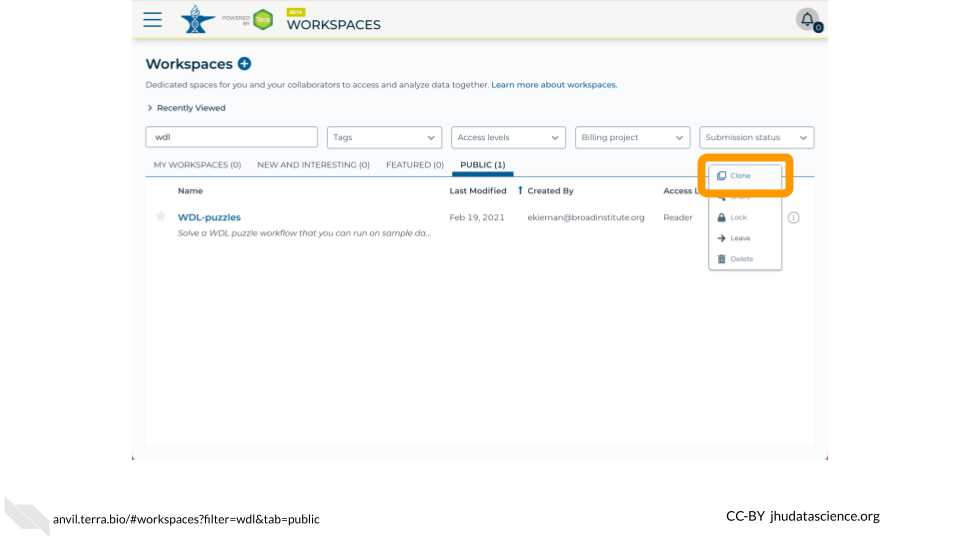
These slides are adapted from the [WDL 101 Workshop](https://support.terra.bio/hc/en-us/articles/8693717360411) (see [this Bioinformática UFMG presentation](https://zenodo.org/record/7062397#.YyM-vy1h1qu) for more information). You can view and download the Google Slides [here](https://docs.google.com/presentation/d/1Dioe_Qd3hdJ0UAP4LA6gJsm07Lr1wRpax5FLYr6xU3M).

## 1.2 Optional: Running a workflow on AnVIL-powered-by-Terra

This optional exercise provides a review of how to run a workflow on AnVIL-powered-by-Terra. It uses the fun and accessible [WDL puzzles](https://support.terra.bio/hc/en-us/articles/360056599991) workspace to run a “Hello, World!” style workflow. By showcasing how input parameters are specified and where output files can be found, this exercise helps provide context before writing your own WDL Workflow.

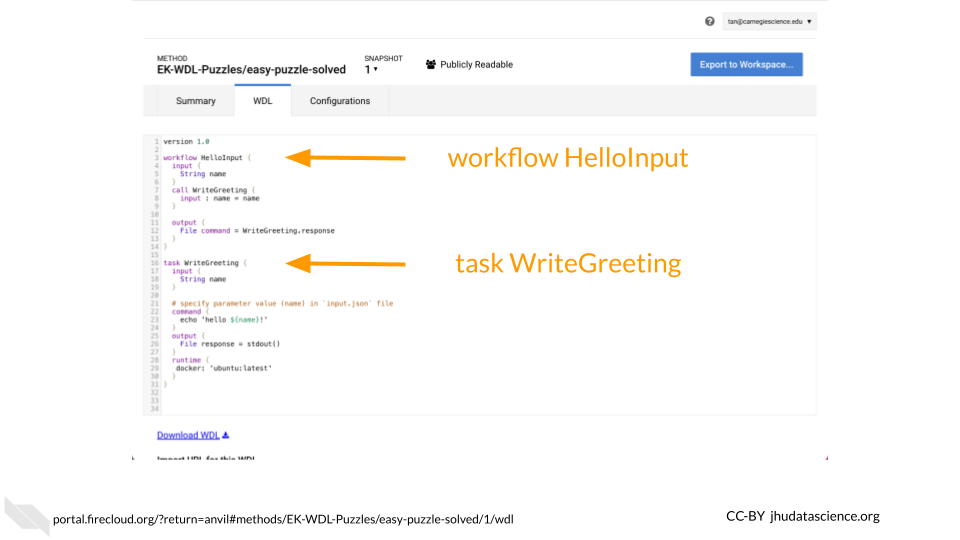
### 1.2.1 Clone Workspace

Clone the [WDL-puzzles](https://app.terra.bio/#workspaces/help-gatk/WDL-puzzles) workspace



### 1.2.2 View WDL

View the **easy-puzzle-solved** WDL script by opening the workflow in the Broad Methods Repository



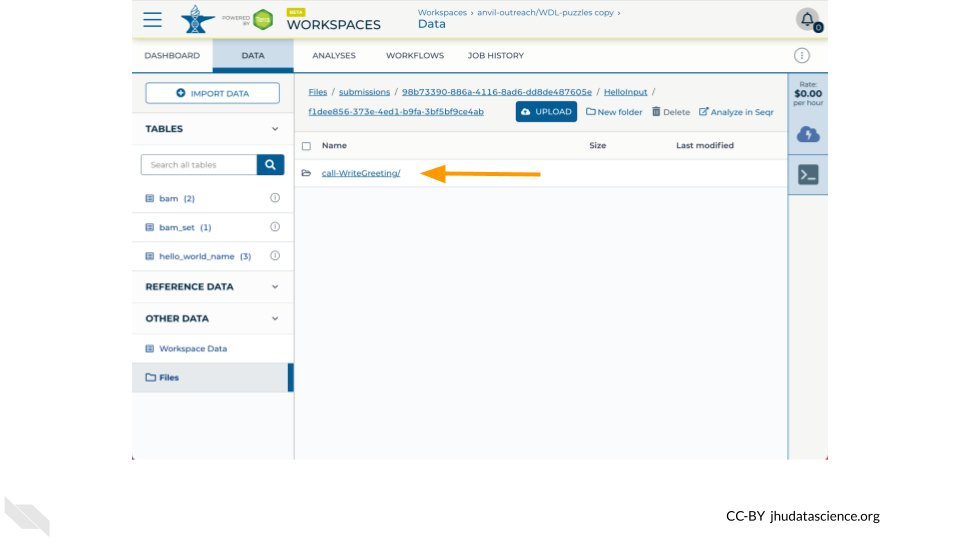
### 1.2.3 Configure Workflow

Configure and run the **easy-puzzle-solved** workflow



### 1.2.4 Examine Output

Examine the output from the **HelloInput** task in the **easy-puzzle-solved** workflow



### 1.2.5 Watch Video

See the following video for more details

You can view and download the Google Slides [here](https://docs.google.com/presentation/d/1FbTgCnAZTgVzp-lPwOVqOqU4Kw19nu825dMngfyc-Nw).

# 2 Import and Configure Workflows

This chapter describes how to import publicly available WDL workflows and their associated configuration files. In addition to covering the growing [Dockstore](https://dockstore.org) community — which implements and helps drive forward [GA4GH](https://www.ga4gh.org) standards — we cover the [Broad Methods Repository](https://portal.firecloud.org/#methods), which provides several convenient features if this is your first time developing WDL Workflows in the Cloud (see Chapter 3 for more details).

**Learning Objectives**

* Understand how to import workflows from Dockstore and the Broad Methods Repository
* Understand the role that JSON files play in configuring imported workflows, and how to select a configuration for your workflow

## 2.1 Importing workflows into AnVIL-powered-by-Terra

Chapter 1’s review exercise covers how to run a workflow that was included in a cloned AnVIL-powered-by-Terra workspace. But what if you want to add a new workflow into your workspace, without writing it yourself?

There are three ways to import workflows into an AnVIL-powered-by-Terra workspace:

* Dockstore
* Broad Methods Repository
* Other online collections

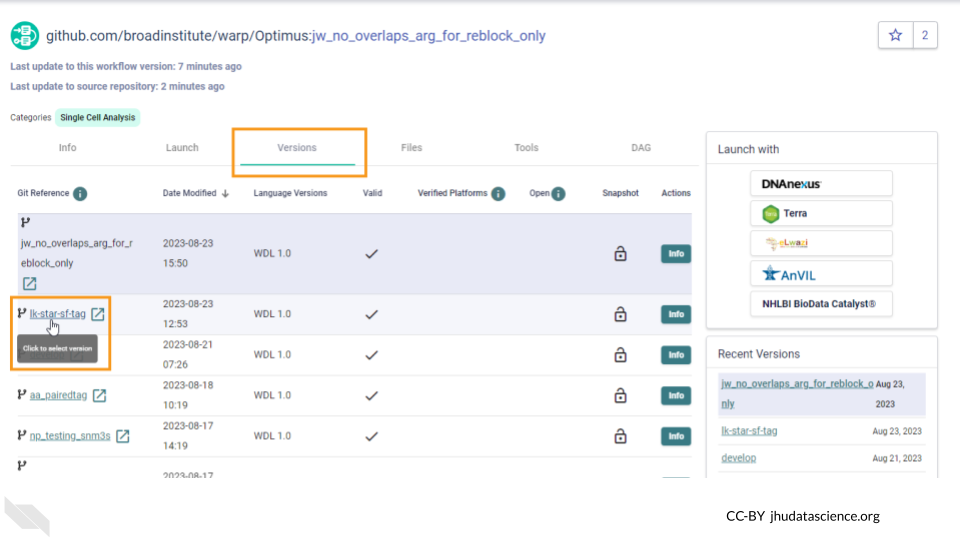
### 2.1.1 Dockstore

[Dockstore](https://dockstore.org) is an app store for bioinformatics tools, including workflows. It provides an easy interface for finding and running workflows on AnVIL-powered-by-Terra.

Complete the following steps to practice importing a workflow from Dockstore:

Use the instructions in [Step 1 of “How to import a workflow and its parameter file from Dockstore into Terra”](https://support.terra.bio/hc/en-us/articles/360038137292#heading-1) to find the Optimus workflow (search for WARP/Optimus).

Then, navigate to the **Versions** tab and click on the version you want to export to AnVIL-powered-by-Terra:



Finally, follow the instructions in [Step 2](https://support.terra.bio/hc/en-us/articles/360038137292#heading-2) to export the workflow to your cloned version of the WDL-puzzles workspace.

### 2.1.2 Broad Methods Repository

The Broad Methods Repository contains a collection of pre-written workflows that can be run on AnVIL-powered-by-Terra. Notably, the Repository provides a web-based WDL editor, making it easy to create and modify workflows in the Cloud. Follow [these instructions](https://support.terra.bio/hc/en-us/articles/360025674392-Finding-the-workflow-method-you-need-and-its-JSON-in-the-Methods-Repository) to open the Broad Methods Repository, find an interesting workflow, and export it to your workspace.

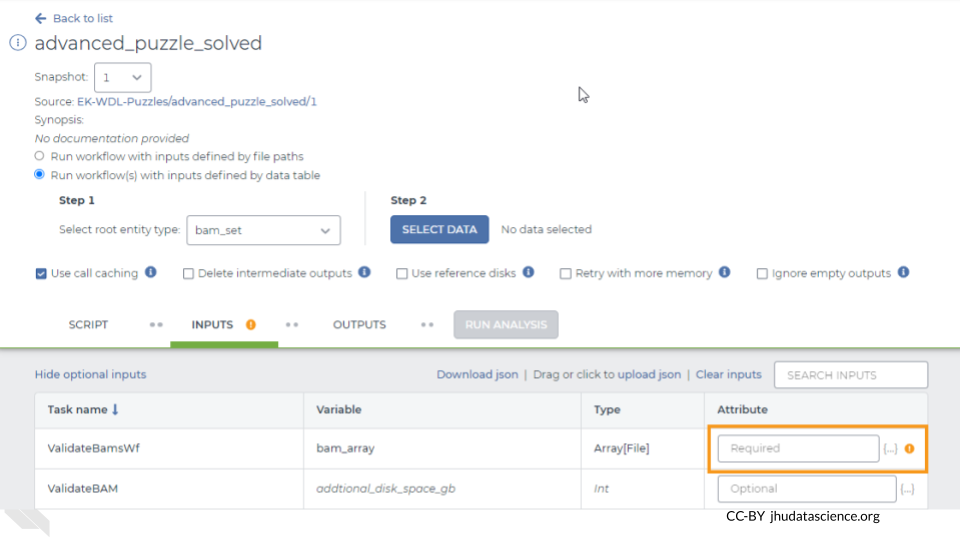
### 2.1.3 Other online collections

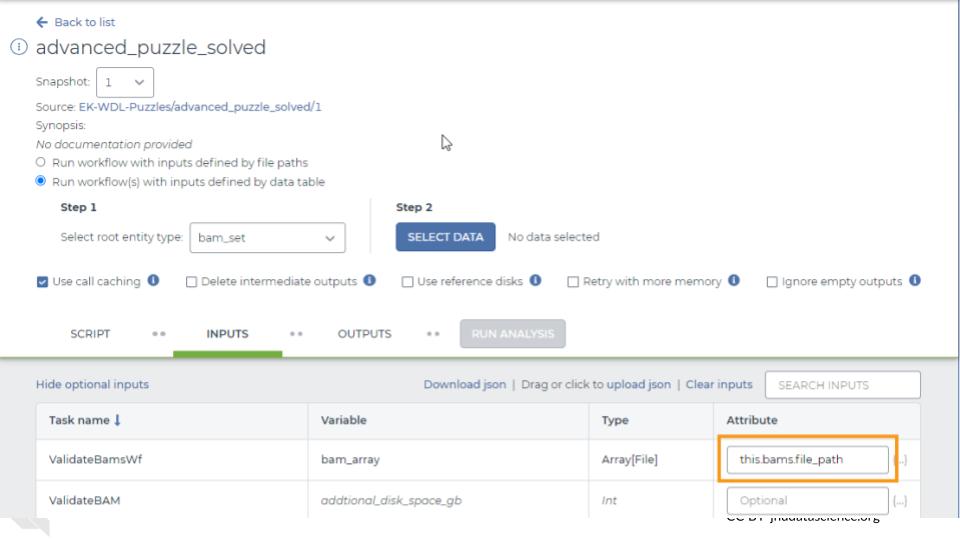
WDL is a widely-used tool, and WDL workflows are publicly available through other sources beyond Dockstore and the Broad Methods Repository. These include [WARP](https://broadinstitute.github.io/warp), [ENCODE](https://www.encodeproject.org/pipelines), the [Genome Analysis Toolkit](https://gatk.broadinstitute.org/hc/en-us), and [GitHub](https://github.com). Note that exporting workflows from these sources is a two-step process: first, copy the workflow’s script to a new method in the Broad Methods Repository (see Chapter 3) or a Dockstore repository. Then, export the workflow to an AnVIL-powered-by-Terra workspace.

## 2.2 Configuring workflows with JSON files

To run a workflow in AnVIL-powered-by-Terra, you need to specify its inputs and outputs. You can do this by hand, filling in the workflow’s configuration each time you run it. But if your workflow’s inputs are reusable across workflow submissions – for example, if you always expect to run the workflow on a column named “bam” in your data table – it’s faster and less error-prone to pre-configure your workflow.

To pre-configure a workflow, you can upload a file that specifies the default values for some set of your workflow’s inputs. JSON is the most common format for this configuration file. AnVIL-powered-by-Terra will automatically fill these values into the workflow configuration form. If necessary. you can always change these values when running the workflow.



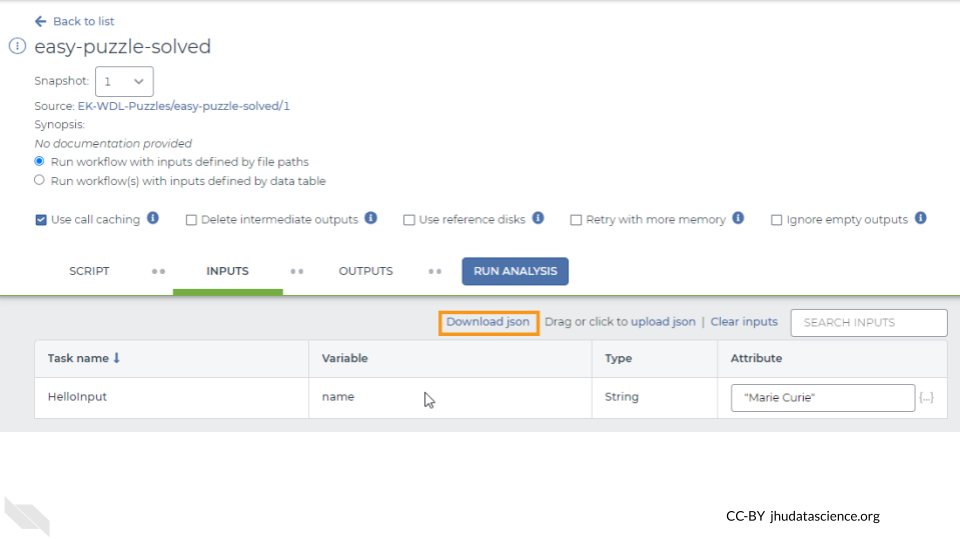


### 2.2.1 Optional practice: pre-configure a workflow

To pre-configure a workflow that you have imported to your workspace, upload a JSON file.

An easy way to do this is to download a JSON file from an existing workflow, edit it, and upload the edited version to the workflow that you’re trying to configure.

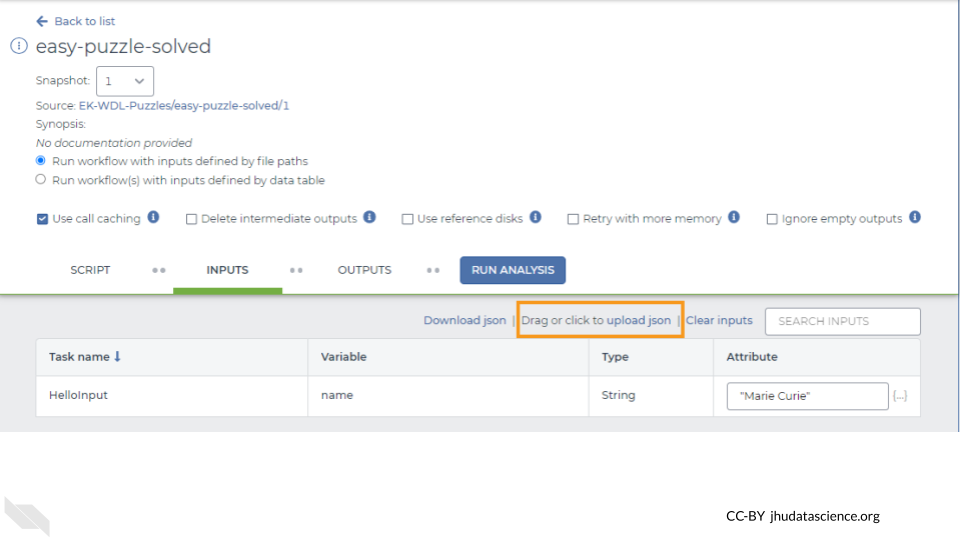
To practice this, open the **easy-puzzle-solved** workflow in your clone of the WDL-puzzles workspace and download the JSON file:



Open the JSON file in a text editor. It should look something like this, if the last time you ran it your input was “Marie Curie”:

{"HelloInput.name":"Marie Curie"}

Edit the file to set the input name to your name, instead of Marie Curie. Save the file, go back to your workflow and upload the edited JSON file by clicking the “Drag or click to upload json” link:



The input for the “name” variable should now be set to your name. Be sure to **click Save** to ensure that this change persists the next time you open the workflow.

### 2.2.2 A few notes on syntax

If you’re configuring multiple inputs, specify them like this:

{"taskName.variableName1":"attribute1", "taskName.variableName2":"attribute2"}

If your inputs will come from a data table, specify them like this:

{"taskName.variableName":"${this.columnName}"}

### 2.2.3 Importing a pre-configured workflow

In some cases, the workflow’s authors have provided a configuration file, and you can export this file to AnVIL-powered-by-Terra along with the workflow, rather than uploading the JSON file yourself. See [How to import a workflow and its parameter file from Dockstore into Terra](https://support.terra.bio/hc/en-us/articles/360038137292) and [Create, edit, and share a new workflow](https://support.terra.bio/hc/en-us/articles/360031366091) (steps 1.6-1.7) for instructions on how to do this for a workflow in Dockstore or the Broad Methods Repository.

### 2.2.4 More information

To learn more about how to configure a workflow’s inputs through the UI, see [How to configure workflow inputs](https://support.terra.bio/hc/en-us/articles/4415971884827). For more detail on how to use JSON files to pre-configure a workflow, read [Getting workflows up and running faster with JSONs](https://support.terra.bio/hc/en-us/articles/360027735471).

## 2.3 Further Reading

If you’re interested in a deeper dive into this chapter’s topics, check out these optional articles:

* For more information on how to monitor your workflow, read [What to expect when you submit a workflow](https://support.terra.bio/hc/en-us/articles/7093972754971).
* To learn about the costs involved in running workflows, read [How much did my workflow cost?](https://support.terra.bio/hc/en-us/articles/360037862771)
* For customizing the compute resources used to run your workflow, read [Workflow setup: VM and other options](https://support.terra.bio/hc/en-us/articles/360026521831).
* For more detail on importing workflows from Dockstore, read [How to import a workflow and its parameter file from Dockstore into Terra](https://support.terra.bio/hc/en-us/articles/360038137292).
* For more detail on importing workflows from the Broad Methods Repository, read [Finding the workflow (method) you need (and its JSON) in the Methods Repository](https://support.terra.bio/hc/en-us/articles/360025674392).
* To learn how to host and share a workflow through the Broad Methods Repository, read [Create, edit, and share a new workflow](https://support.terra.bio/hc/en-us/articles/360031366091).

# 3 Write WDL

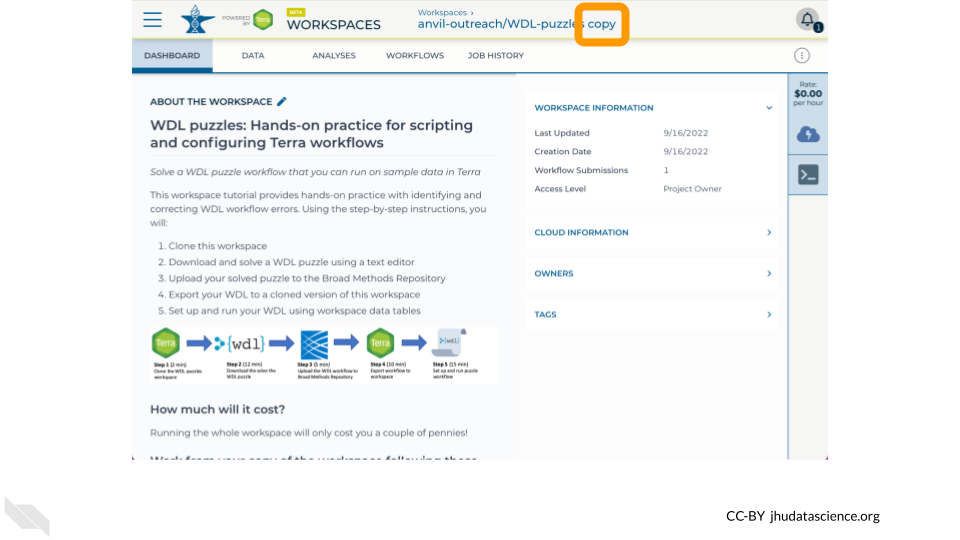
Now that you’ve successfully run a Workflow on AnVIL-powered-by-Terra, this tutorial demonstrates how you can create and edit a WDL using the [Broad Methods Repository](https://portal.firecloud.org/?return=anvil#methods). While this “legacy” Methods repository does not have many of the features present in the open-source [Dockstore](https://dockstore.org/) platform, it does offer a convenient web-based editor. This material is adapted from the [WDL 101 Workshop](https://support.terra.bio/hc/en-us/articles/8693717360411); you can read about other ways the Broad Methods Repository can be used in [this Terra Support article](https://support.terra.bio/hc/en-us/articles/360031366091).

**Learning Objectives**

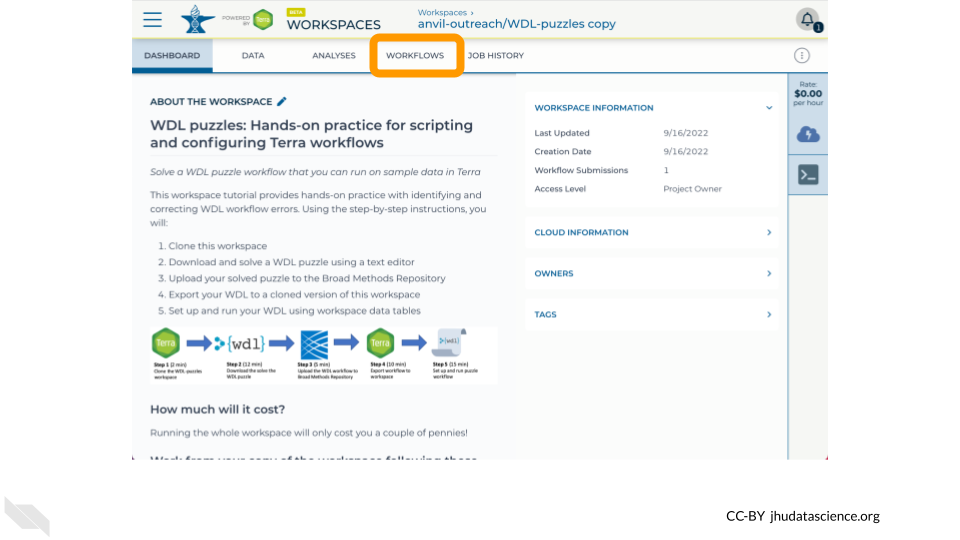
1. Access Broad Methods Repository
2. Write a basic “hello-world” WDL
3. Export WDL to AnVIL-powered-by-Terra and run it

## 3.1 Access Broad Methods Repository

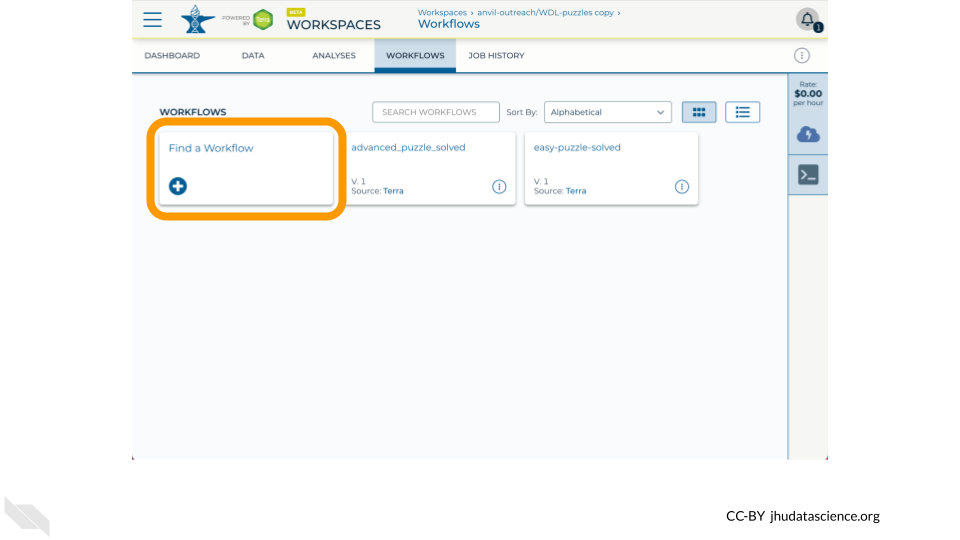
Let’s start by navigating to the [WDL-puzzles workspace](https://app.terra.bio/#workspaces/help-gatk/WDL-puzzles) on AnVIL-powered-by-Terra. If you completed the review exercise in Chapter 1, you already cloned this workspace. If not, clone it to create your own copy. Please **double check your workspace name** to ensure that this is the copy that you made rather than the original as you will not be able to use the original workspace to create a new WDL or run a workflow.



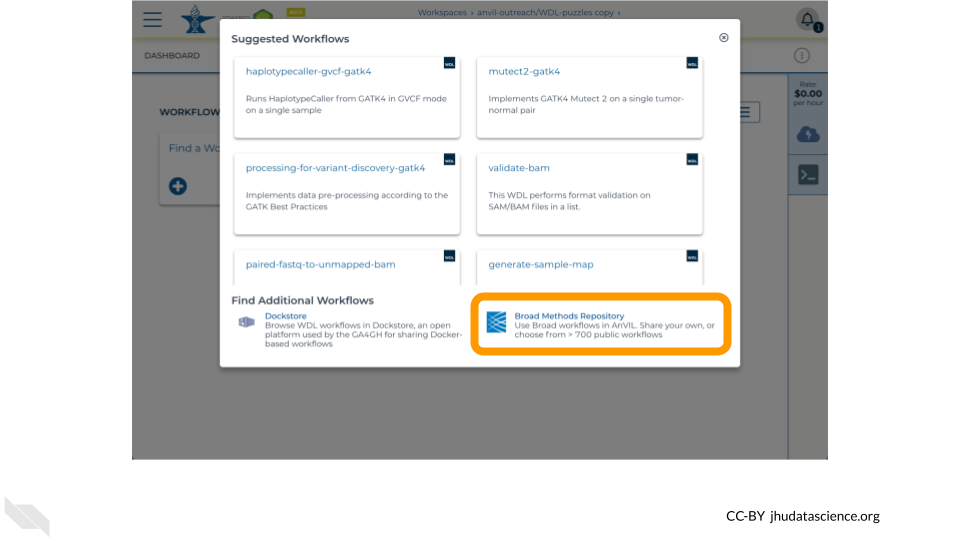
Once you’ve double checked that you are in a workspace that you can modify and compute, click on the Workflows tab.



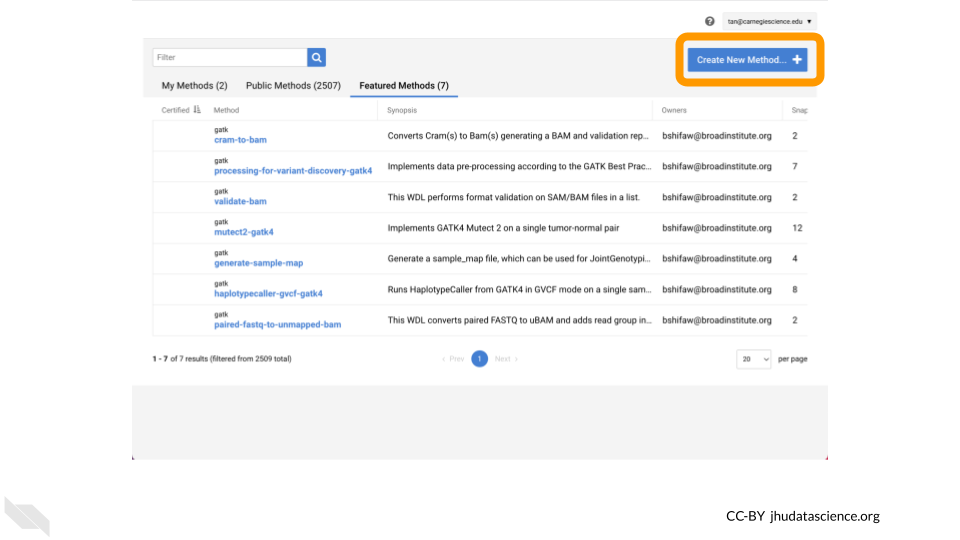
Click on the Find a Workflow card.



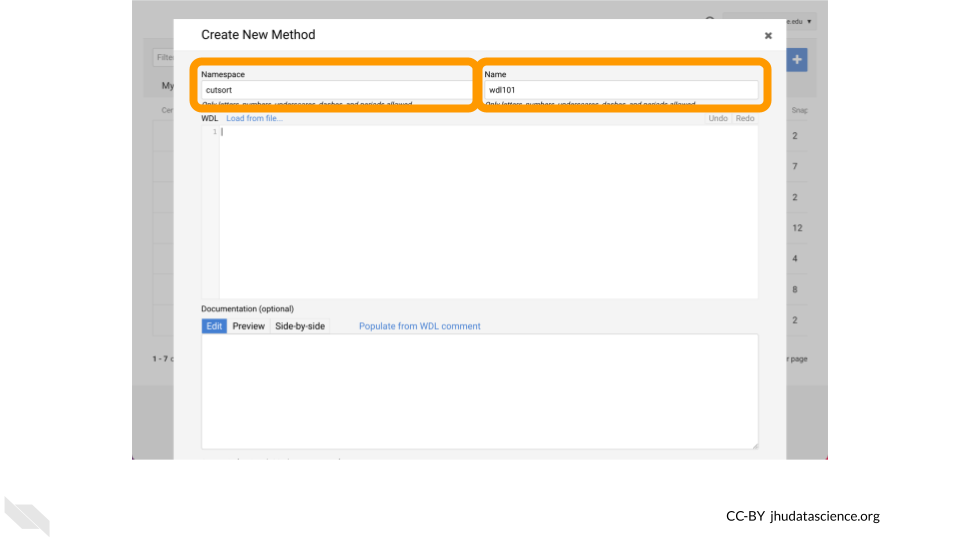
Select the Broad Methods Repository option.



Click Create New Method.

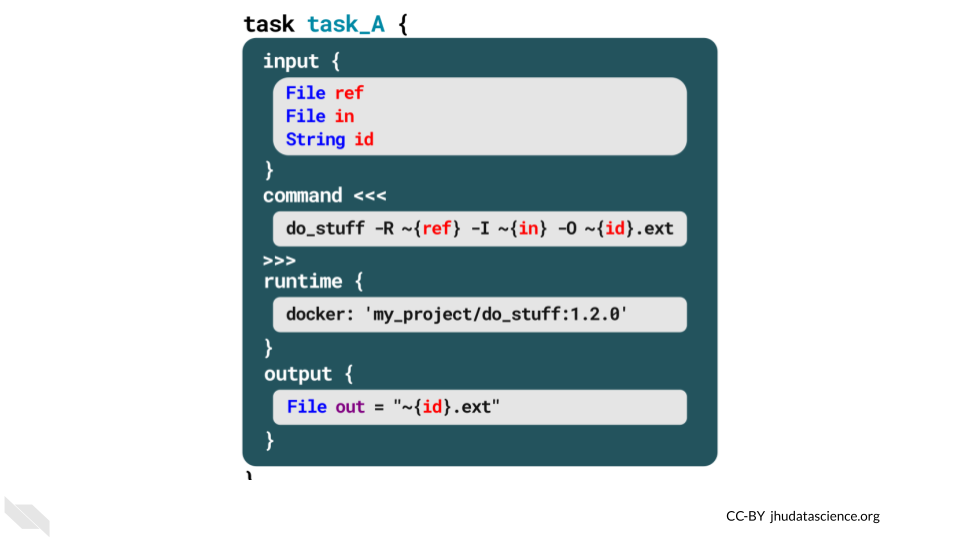


Add a namespace to the first text box to organize your WDLs. Your username (perhaps prepended with your lab name) is a reasonable namespace as this must be unique across all of Broad Methods Repository. Afterwards, add a name such as wdl101 to name your WDL.



## 3.2 WDL Script Structure

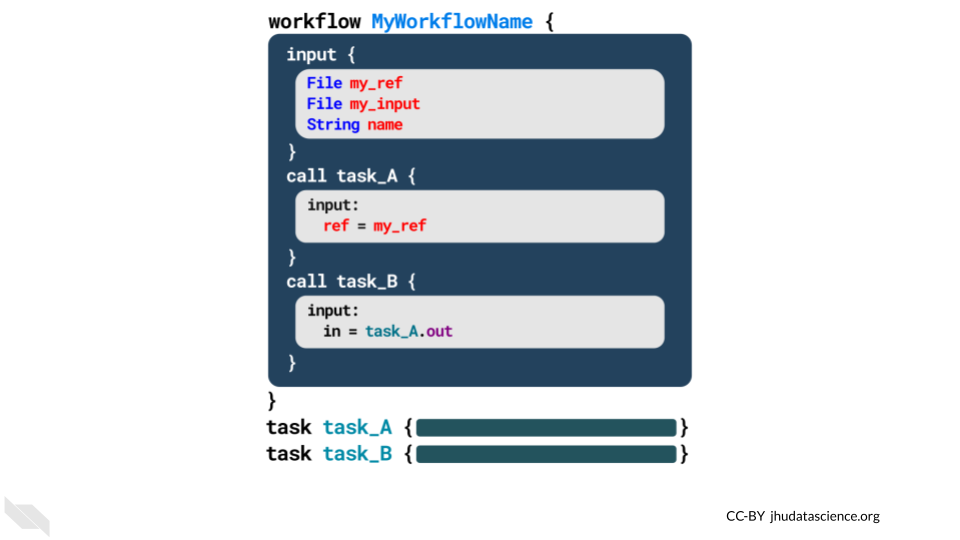
Workflows are typically broken down into **tasks**, which each execute one step in the workflow’s analysis. Tasks are written in the Workflow Description Language (**WDL**, pronounced “widdle”). WDL tasks have a consistent structure, illustrated in this example:



Each task has 4 sections:

1. **Input** - defines the inputs that the task expects and their types.
2. **Command** - uses bash commands to call software tools (e.g., Python) to transform the input data.
3. **Runtime** - specifies the Docker container in which the workflow will run – we’ll discuss this more in Chapter 5.
4. **Output** - defines the output variable(s) that the transformed data will be written to.

The **workflow definition** strings together the workflow’s tasks:



In this example, the workflow called “MyWorkflowName” calls two tasks (“task\_A” and “task\_B”). The workflow starts with an **input section**, just as the tasks do. It then calls each of the tasks, using the output from prior tasks as the inputs for subsequent tasks. Finally, workflows often include an **output** section (although the example above does not).

The workflow definition and task scripts can be written in separate WDL scripts or combined into a single script (as shown in this example).

## 3.3 Write a hello-input WDL

Let’s now create a basic WDL! This simple “Hello, World!” style workflow will take a string as input, call a single task, and save the output of that task to your workspace bucket. The task that is called will run the [Bash](https://swcarpentry.github.io/shell-novice/01-intro.html) echo command to print the input string to stdout.

First note that we are using the [WDL 1.0 spec](https://github.com/openwdl/wdl/tree/main/versions).

version 1.0

Let’s add a workflow HelloInput that calls a single task WriteGreeting.

version 1.0  
workflow HelloInput {  
}  
  
task WriteGreeting{  
}

To create the task, we will define input, command, output, and runtime blocks. Note that the command block is defined as a “[here doc](https://en.wikipedia.org/wiki/Here_document)” and prints the input string to stdout.

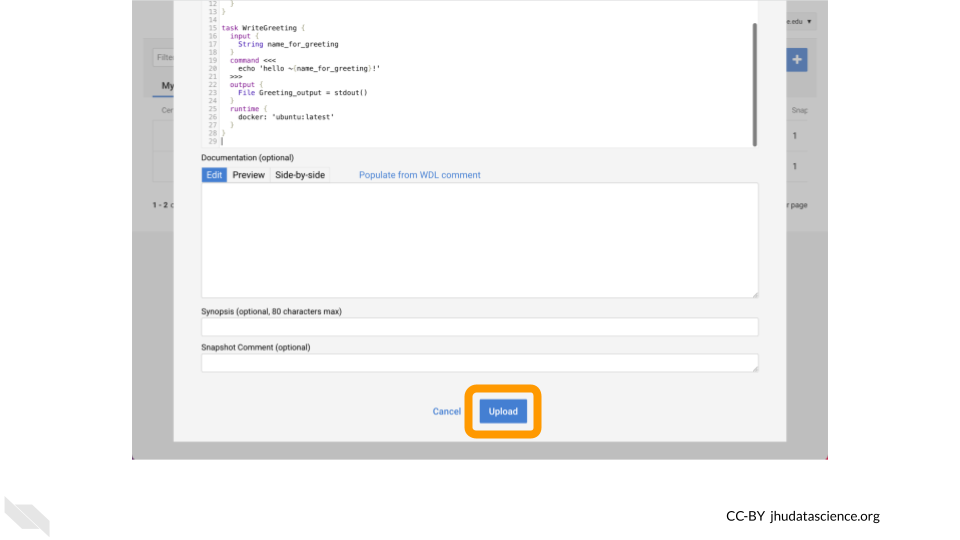
version 1.0  
workflow HelloInput {  
}  
  
task WriteGreeting {  
 input {  
 String name\_for\_greeting  
 }  
 command <<<  
 echo 'hello ~{name\_for\_greeting}!'  
 >>>  
 output {  
 File Greeting\_output = stdout()  
 }  
 runtime {  
 docker: 'ubuntu:latest'  
 }  
}

Putting it all together, we now create the workflow by defining an input string stored in a variable named name\_input, calling the task by passing name\_input to name\_for\_greeting, and storing what is returned by the task in a File labeled final\_output.

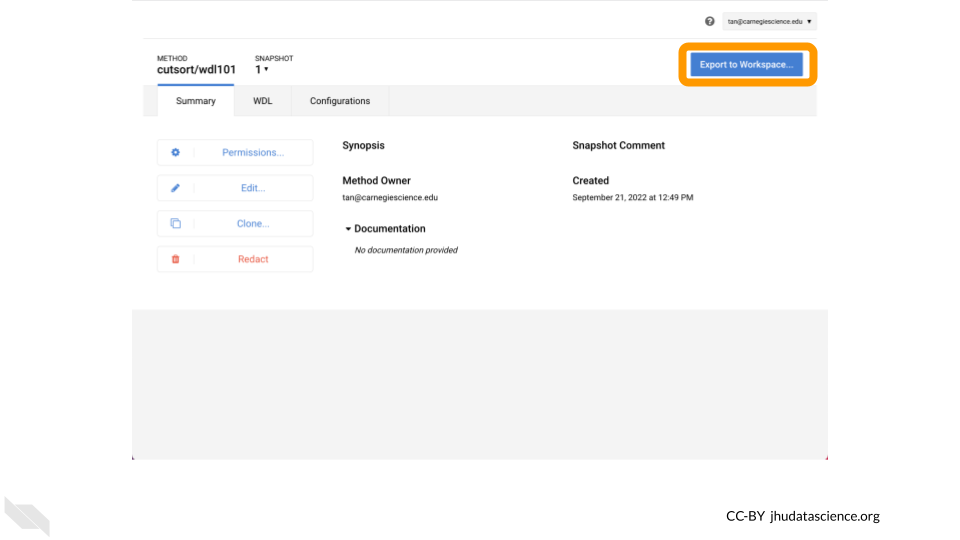
version 1.0  
workflow HelloInput {  
 input {  
 String name\_input  
 }  
 call WriteGreeting {  
 input:   
 name\_for\_greeting = name\_input  
 }  
 output {  
 File final\_output = WriteGreeting.Greeting\_output  
 }  
}  
  
task WriteGreeting {  
 input {  
 String name\_for\_greeting  
 }  
 command <<<  
 echo 'hello ~{name\_for\_greeting}!'  
 >>>  
 output {  
 File Greeting\_output = stdout()  
 }  
 runtime {  
 docker: 'ubuntu:latest'  
 }  
}

## 3.4 Export to AnVIL-powered-by-Terra and run

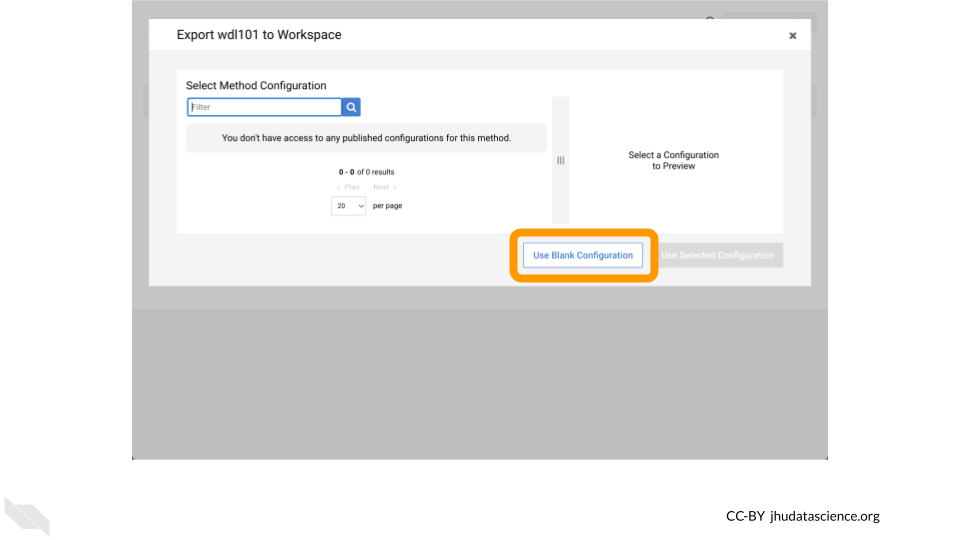
Once your WDL is complete, click on Upload.



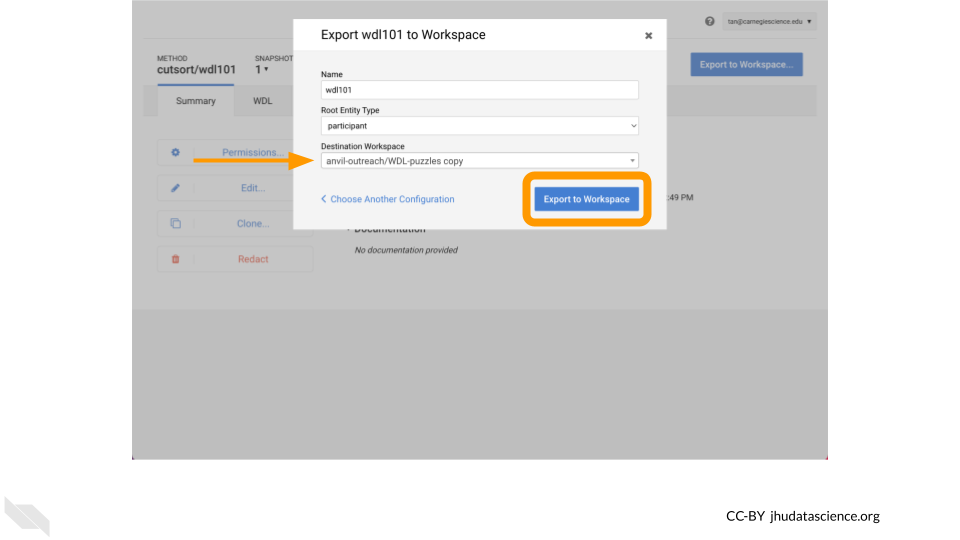
Now click on Export to Workspace.



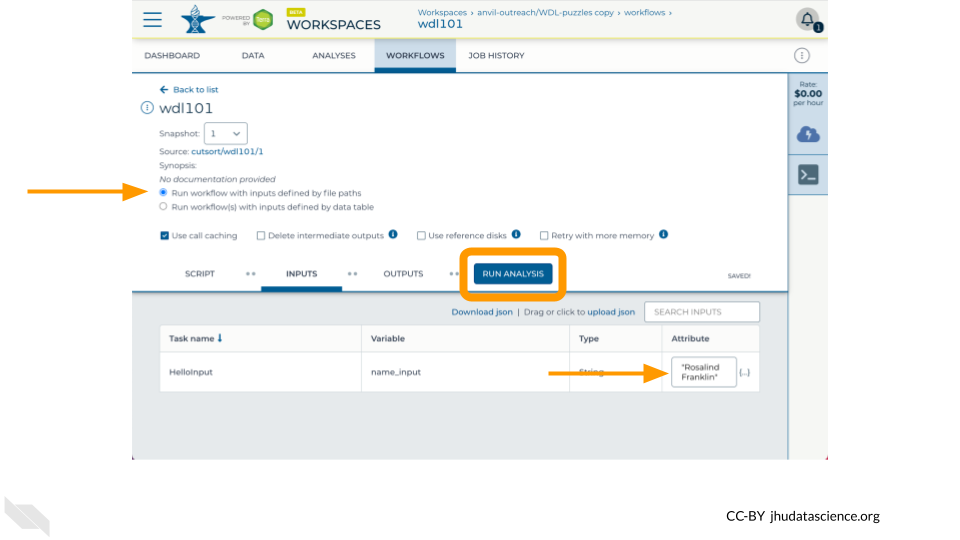
Select Use Blank Configuration.



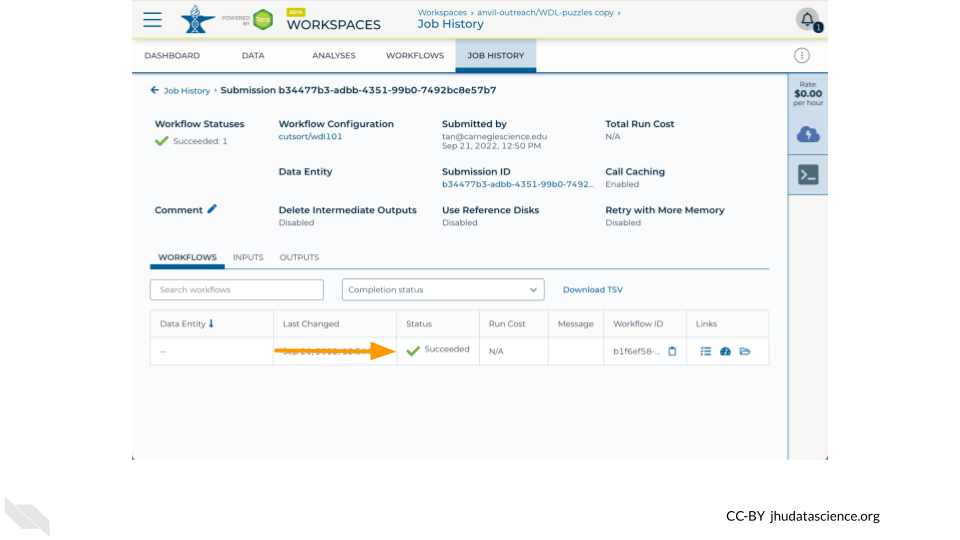
Select a Destination Workspace such as your clone of WDL-puzzles. Afterwards, click Export to Workspace.



Lastly, configure your Workflow as you did in Chapter 1 (e.g. inputs defined by file paths, name in double quotes), click Save, and then click Run Analysis.



Voila! Here’s what you hopefully see after successfully running your WDL101 Training Example !



# 4 Calculate idxstats on multiple files

Now that you’ve successfully written an introductory WDL, this chapter demonstrates how WDL Workflows can easily perform an analysis across multiple genomic data files. This material is adapted from the [WDL Bootcamp workshop](https://support.terra.bio/hc/en-us/articles/18618717942427). For more hands-on WDL-writing exercises, see [Hands-on practice for scripting and configuring Terra workflows](https://support.terra.bio/hc/en-us/articles/360056599991).

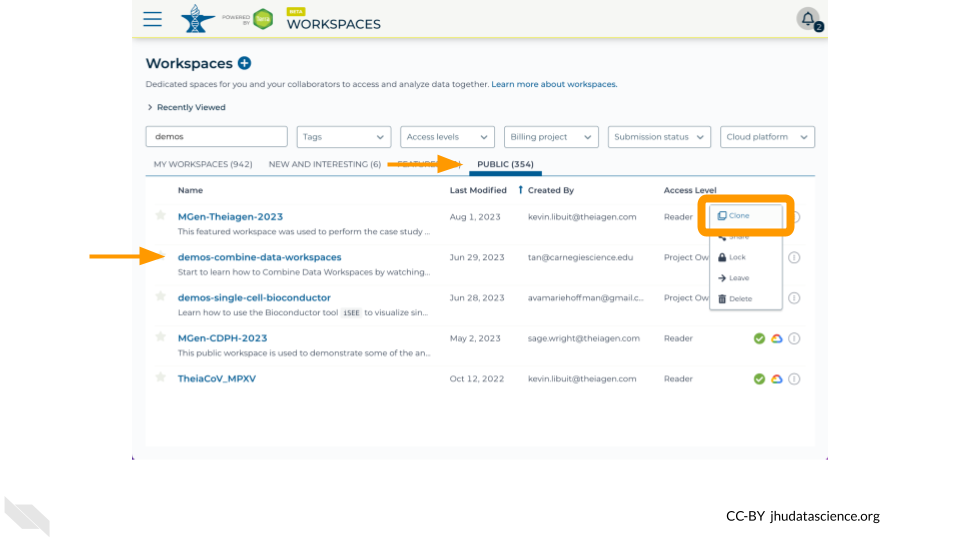
**Learning Objectives**

* Solidify your understanding of a WDL script’s structure
* Modify a template to write a more complex WDL
* Understand how to customize a workflow’s setup with WDL

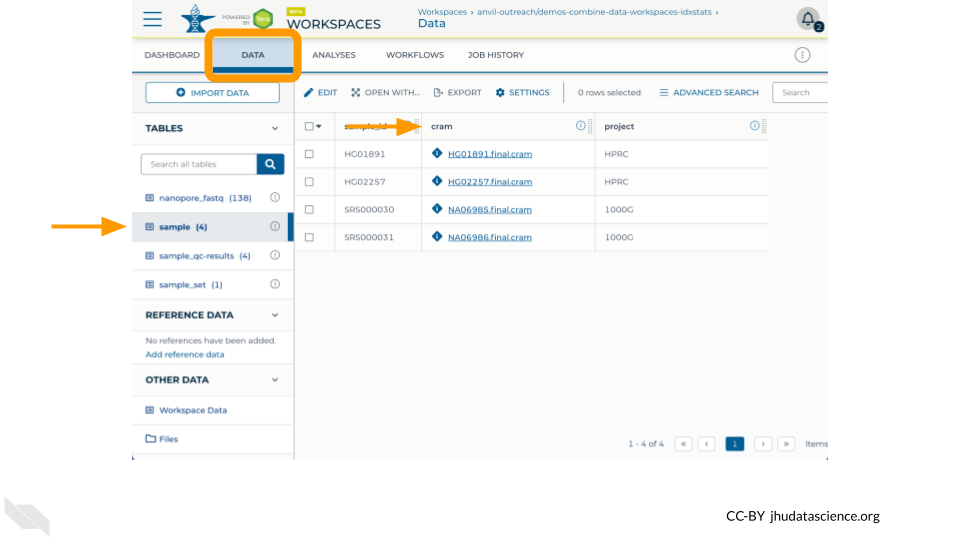
## 4.1 Clone data workspace

Let’s start by navigating to the [demos-combine-data-workspaces](https://anvil.terra.bio/#workspaces/anvil-outreach/demos-combine-data-workspaces) on AnVIL-powered-by-Terra. This workspace contains a Data Table named sample which contains references to four .cram files, two from the [1000 Genomes Project](https://anvil.terra.bio/#workspaces/anvil-datastorage/1000G-high-coverage-2019) and two from the [Human Pangenome Reference Consortium](https://anvil.terra.bio/#workspaces/anvil-datastorage/AnVIL_HPRC).

Clone this workspace to create a place where you can organize your analysis.



Examine the sample table in the Data tab to ensure that you see references to four .cram files.



In the next steps, you will write a WDL to analyze these .cram files, run the Workflow, and examine the output.

## 4.2 Write an idxstats WDL

To build off of your hello-input WDL, let’s practice writing a more complex WDL. In this exercise, you’ll fill in a template script to calculate Quality Control (QC) metrics for a BAM/CRAM file using the samtools idxstats function. For this exercise, there are two WDL runtime parameters that we **must** update for the Workflow to succeed:

* docker – Specify a Docker image that contains necessary software
* disks – Increase the disk size for each provisioned resource

Start by [downloading the template script](https://drive.google.com/file/d/1OH4L5LQNquDhNvycRHzWVH6Z1HR5R7kD) shown below. Open it in a text editor and modify it to call samtools idxstats on a bam file.

version 1.0  
workflow samtoolsIdxstats {  
 input {  
 File bamfile   
 }  
 call {  
 input:   
 }  
 output {  
 }  
}  
  
task {  
 input {  
 }  
 command <<<  
 >>>  
 output {  
 }  
 runtime {  
 docker: ''  
 }  
}

Importantly, you **must**:

* Specify a Docker image that contains SAMtools (e.g. ekiernan/wdl-101:v1)
* Increase the disk size for each provisioned resource, e.g. local-disk 50 HDD

Hints:

* Follow the same general method as in the “Hello World” exercise in section 3.3.
* In this case, the input will be a BAM file.
* The output will be a file called idxststats.txt.
* The task will be to calculate QC metrics, using this command:

samtools index ~{bamfile}  
samtools idxstats ~{bamfile} > counts.txt

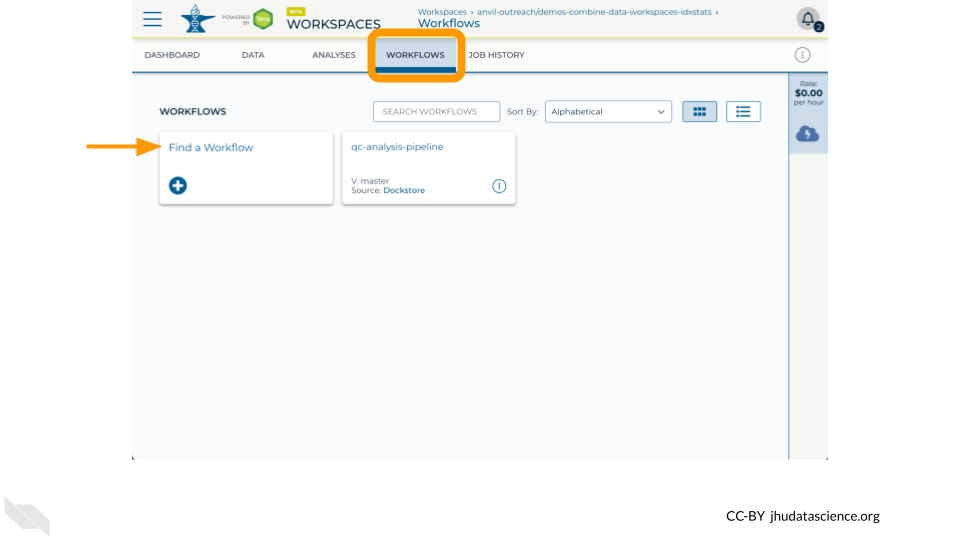
Then, compare your version to the completed version below:

version 1.0  
workflow samtoolsIdxstats {  
 input {  
 File bamfile  
 }  
 call idxstats {  
 input:   
 bamfile = bamfile  
 }  
 output {  
 File results = idxstats.idxstats  
 }  
}  
  
task idxstats {  
 input {  
 File bamfile  
 }  
 command <<<  
 samtools index ~{bamfile}  
 samtools idxstats ~{bamfile} > idxstats.txt  
 >>>  
 output {  
 File idxstats = "idxstats.txt"  
 }  
 runtime {  
 disks: 'local-disk 50 HDD'  
 docker: 'ekiernan/wdl-101:v1'  
 }  
}

## 4.3 Optional: Run idxstats WDL on multiple .bam files

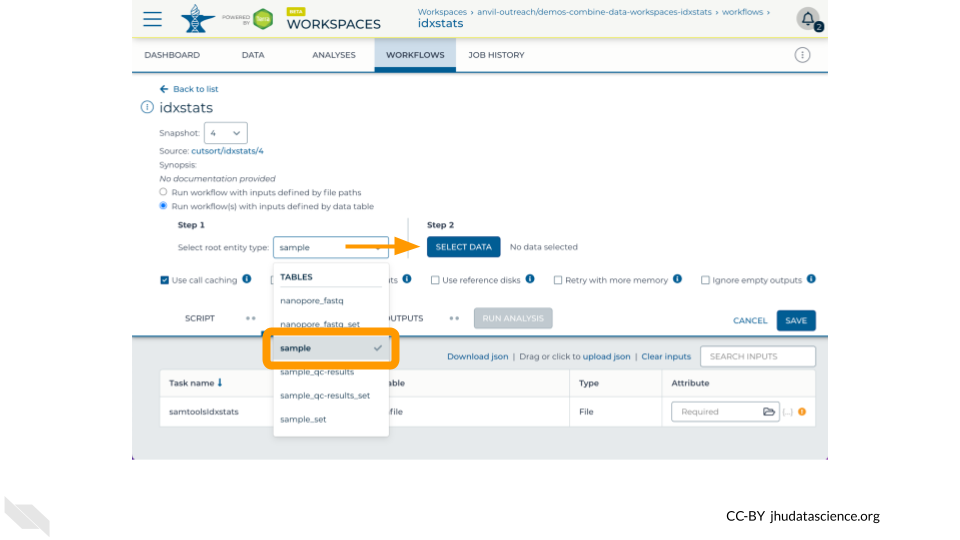
You can test this workflow out by creating a new method in the Broad Methods Repository, exporting it to your clone of the demos-combine-data-workspaces workspace, and running it on samples in the ’sample data table.

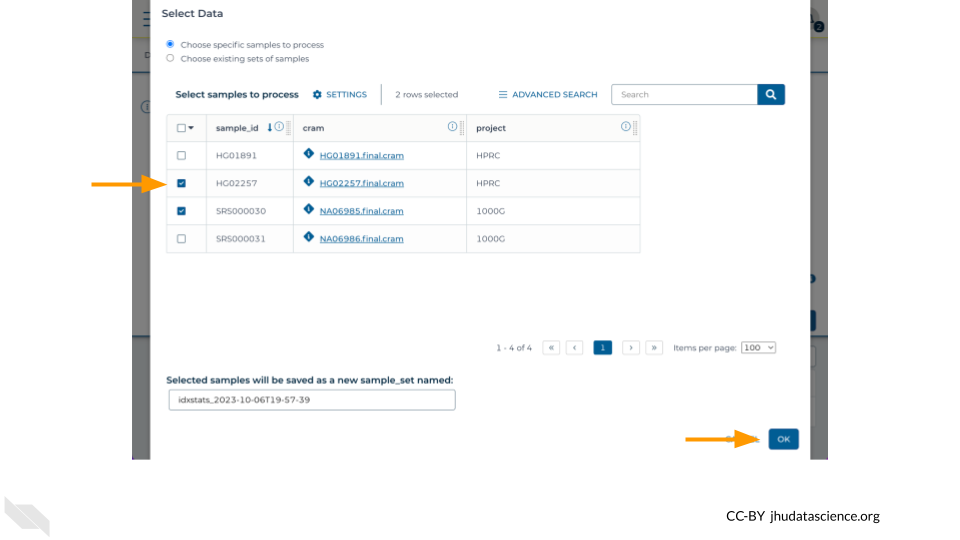
First, go to the “Workflows” tab and access the Broad Methods Repository through the “Find a Workflow” card:



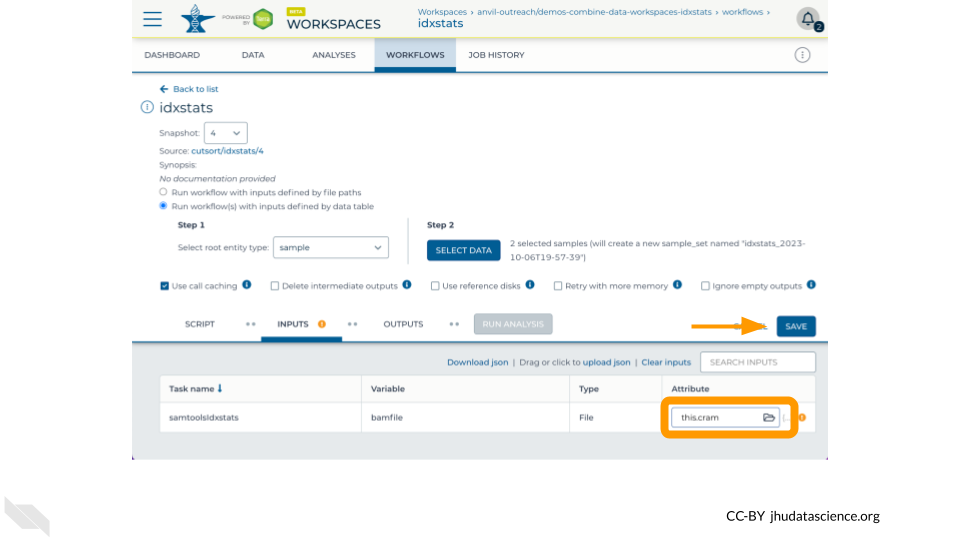
Copy and paste the idxstats WDL you wrote above and export to your workspace (see Chapter 3 if you need a refresher). Next, select “Run workflow(s) with inputs defined by data table” and choose the .cram files that you wish to analyze:

* Step 1: Select the sample table in the root entity type drop-down menu
* Step 2: Click “Select Data” and tick the checkboxes for one or more rows in the data table

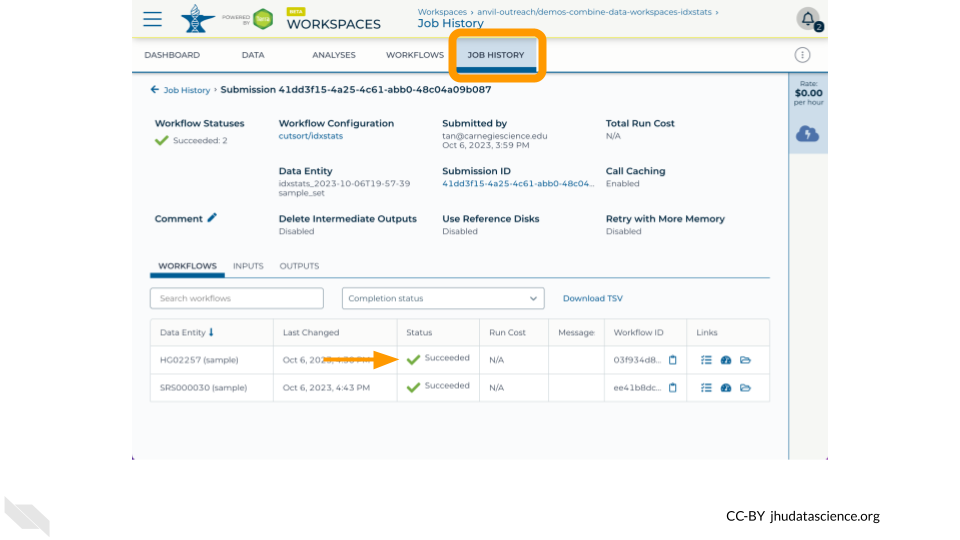




Finally, configure the “Inputs” tab by specifying this.cram as the Attribute for the variable bamfile for the task samtoolsIdxstats. **Don’t forget to click “Save”**.

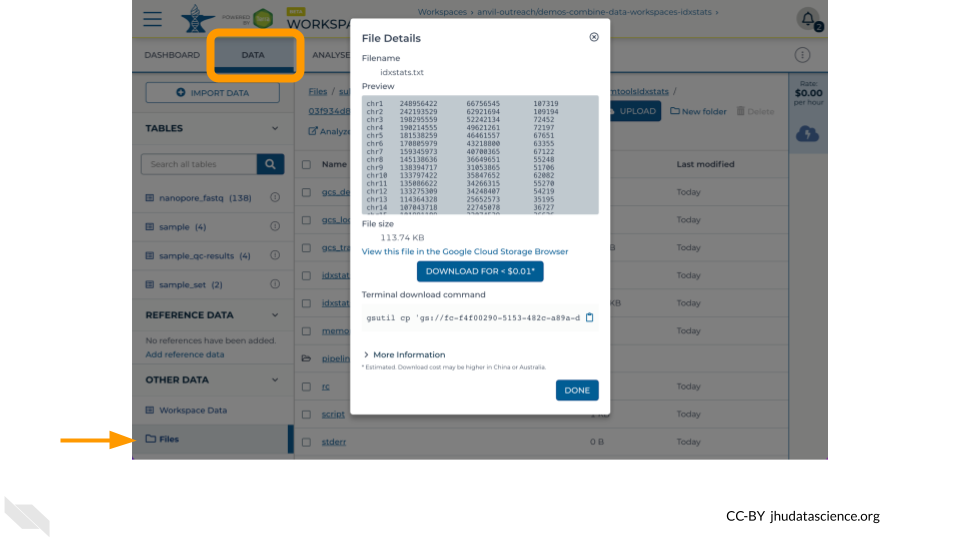


Now run the job by clicking “Run Analysis”! You can monitor the progress from “Queued” to “Running” to “Succeeded” in the “Job History” tab



Once the job is complete, navigate to the “Data” tab and click on “Files” to find the idxstats.txt output and logs by traversing through

submissions/<submission\_id>/samtoolsIdxstats/<workflow\_id>/call-idxstats/



## 4.4 Customize your Workflow’s Setup with WDL

In addition to defining the workflow’s tasks, WDL scripts can define how your workflow runs in AnVIL-powered-by-Terra.

### 4.4.1 Memory retry

Some workflows require more memory than others. But, memory is not free, so you don’t want to request more memory than you need. One solution to this tension is to start with a small amount of memory and then request more if you run out of memory. Learn how to do this from your WDL script by reading [Out of Memory Retry](https://support.terra.bio/hc/en-us/articles/4403215299355), and see [Workflow setup: VM and other options](https://support.terra.bio/hc/en-us/articles/360026521831) for a general overview of how to set up your workflow’s compute resources.

### 4.4.2 Localizing files

It can be hard to know where your data files are located within your workspace bucket – the folders aren’t intuitively named, and often your files are saved several folders deep.

Luckily, WDL scripts can localize your files for you. For more on this, see [How to configure workflow inputs](https://support.terra.bio/hc/en-us/articles/4415971884827), [How to use DRS URIs in a workflow](https://support.terra.bio/hc/en-us/articles/6635144998939), and [Creating a list file of reads for input to a workflow](https://support.terra.bio/hc/en-us/articles/360033353952).

If your workflow generates files, you can also write their location to a data table. This is useful for both intermediate files and the workflow’s final outputs. For more on this topic, see [Writing workflow outputs to the data table](https://support.terra.bio/hc/en-us/articles/4500420806299).

# 5 Customize Docker

Having run a workflow, written a WDL, and localized a file, let’s now build a custom Docker image. This tutorial demonstrates how to build a Docker image that contains [SAMtools](https://pubmed.gov/19505943) so that we can calculate [idxstats](http://www.htslib.org/doc/samtools-idxstats.html) on a small .bam file. We will use the GitHub Action [build-and-push-docker-images](https://github.com/marketplace/actions/build-and-push-docker-images) so that you do not need to have Docker installed locally. This material was modified from the [WDL Bootcamp Workshop](https://support.terra.bio/hc/en-us/articles/18618717942427). More information about making Docker images can be found in [this Terra Support article](https://support.terra.bio/hc/en-us/articles/360024737591).

**Learning Objectives**

1. Identify what containerization is, and why it’s helpful for running workflows
2. Identify the key components of a Docker container
3. Understand how to find and use a pre-existing Docker image
4. Configure GitHub Actions
5. Build and run a Docker image
6. Create idxstats WDL
7. Run idxstats workflow
8. Understand how to specify a container for an AnVIL-powered-by-Terra workflow

## 5.1 Prerequisites

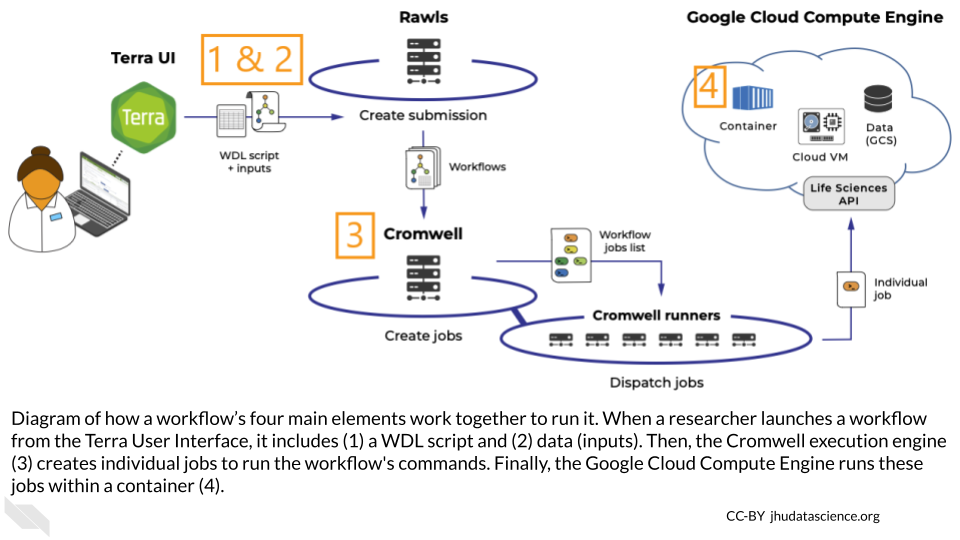
**GitHub**: [Create an account](https://github.com/signup) to build Docker images using GitHub Actions

**Docker Hub**: [Create an account](https://hub.docker.com/signup) to publish Docker images that AnVIL can access

## 5.2 Containerization with Docker

Behind the scenes, a workflow requires four main elements:

1. **Data**
2. A **workflow script** (written in WDL)
3. An **execution engine** to manage the workflow’s jobs
4. A **container** in which to run the workflow



These elements all work together to run the workflow – the container’s role is to control all of the code packages and dependencies used to run the workflow’s WDL script. This is called “**containerization**.”

### 5.2.1 Why is containerization helpful?

Containerization makes it easier to reproduce workflow analyses. For example, if two collaborators are analyzing the same data using different versions of Python, they might get different results. Containerization controls the environments in which you’re running these analyses, saving you from puzzling over incompatible results. Containerization can also ensure that other researchers can replicate your results and apply your tools to their own data. Containerization is even helpful when re-running old code – if your code packages have automatically updated since you last ran the code, using a container may prevent your code from breaking.

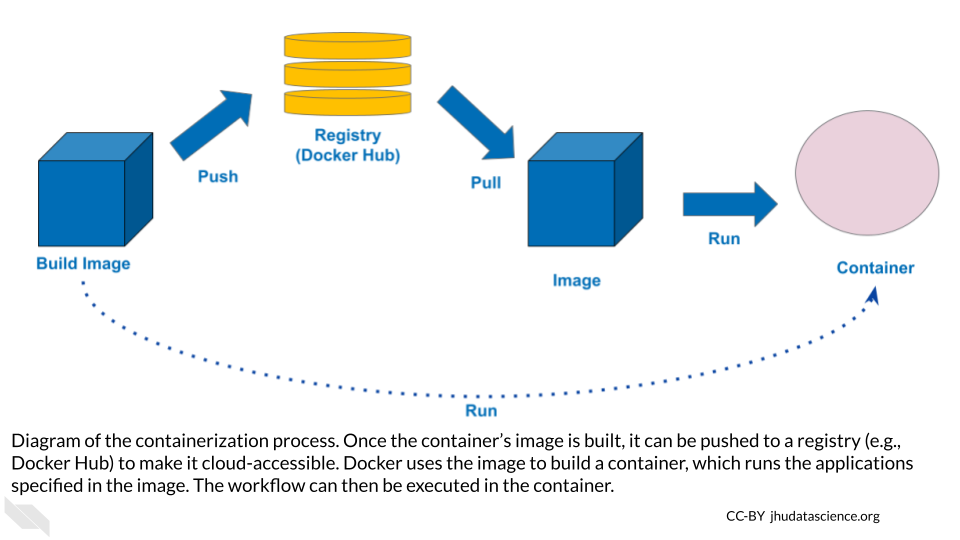
### 5.2.2 Docker containers

You can manage your containers with several services. We’ll focus on **Docker**.

Docker containers have two main components:

1. A **Docker file** that defines the container’s dependencies, environment variables, file system, and applications.
2. A **Docker image** that builds and runs a container, which contains everything defined in the Docker file.

In addition, a **registry** (e.g., DockerHub) is used to share Docker images with others, and to make your Docker image accessible from the Cloud (e.g., in AnVIL-powered-by-Terra).



### 5.2.3 Using Docker containers in your workflow

A workflow’s container constrains the code that you can write in the WDL script; for example, a WDL script with Python commands must be run in a container that includes Python. So, how can you ensure that you’re using the right container on AnVIL-powered-by-Terra?

To direct AnVIL-powered-by-Terra toward the correct container, specify the container’s image in the WDL script. We already did this in the exercise from Chapter 3, by including a “docker” variable in the “runtime” section of the task definition:

runtime {  
 docker: 'ubuntu:latest'  
 }

In this example, we simply specified the most up-to-date Ubuntu image. However, in many cases the best choice is to use a specific image, which won’t change after you’ve written the workflow.

Note that you can use different images in different tasks within your workflow.

## 5.3 Using a Published Docker Image Out of the Box

To set up your workflow’s container, you can build a Docker image from scratch, modify an existing image, or find a published image that you can use out of the box, without any modifications.

This last option is a particularly good option when you only need your container to include a single tool. You can find useful images in the following ways:

### 5.3.1 Google

A good first step is to search for “Docker image” and the name of the software that you want to include in the container on a search engine like Google or Bing.

### 5.3.2 Docker Hub and Quay.io

[Docker Hub](https://hub.docker.com) is an online platform for sharing Docker images. Use the “Explore” menu to filter for suitable images. It’s a good idea to filter for images from trusted sources, including Docker official images and images from verified publishers.

[Quay.io](https://quay.io) is another Docker-sharing platform, with similar functionality.

### 5.3.3 DockerBIO

[DockerBIO](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6266945) is a Java web application that focuses on Docker images for bioinformatics analyses. This approach requires a bit more setup, but can make it easier to find an image that’s relevant to your work.

### 5.3.4 Next steps

Once you’ve found a published image that suits your needs, clone it to create your own copy. This will ensure that your analysis is reproducible, because the image won’t change unless you edit it.

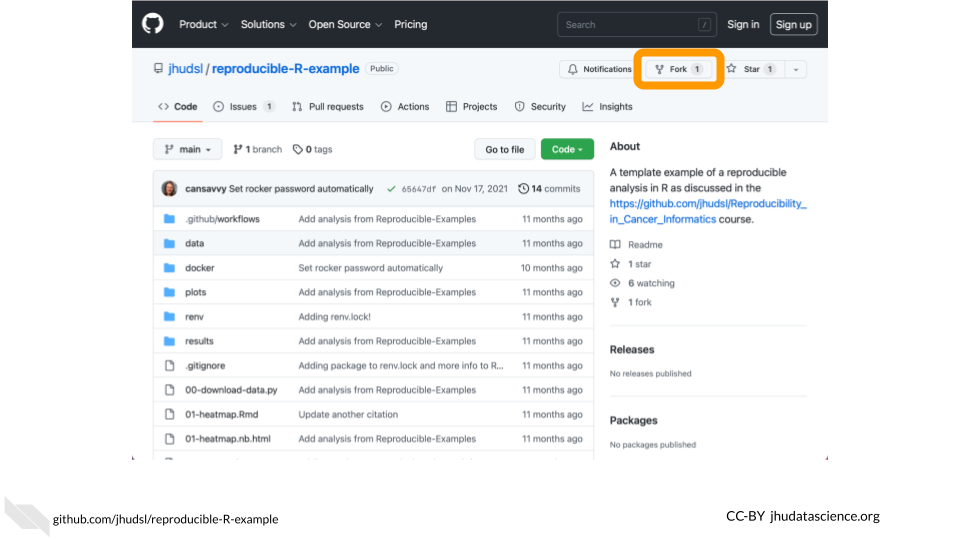
Sections 5.4 and 5.5 show you how to take a step further, and modify a published image for your workflow. To see how to use a Docker image without any modification, skip these sections and start again at section 5.6.

## 5.4 Configure GitHub Actions

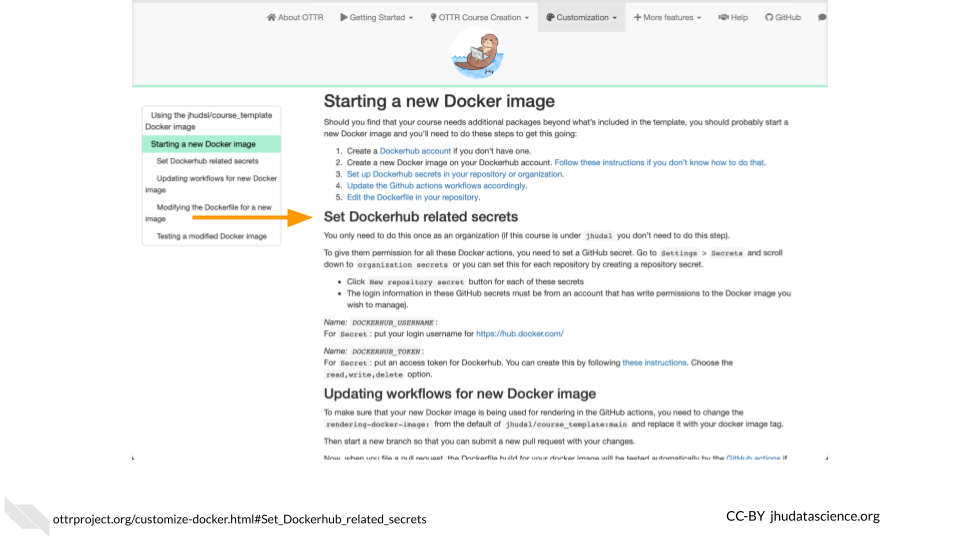
In the next section, we’ll get more in the weeds to show you how to modify a published Docker image to customize it for your workflow.

The GitHub Action [build-and-push-docker-images](https://github.com/marketplace/actions/build-and-push-docker-images) provides a Cloud-based solution to building a Docker image and pushing it to [Docker Hub](https://hub.docker.com). We will start with a repository created for the ITN course [Intro to Reproducibility in Cancer Informatics](https://jhudatascience.org/Reproducibility_in_Cancer_Informatics) that has this GitHub Action configured.

First, fork the [reproducible-R-example](https://github.com/jhudsl/reproducible-R-example) repository.



Your new repository must be configured with the proper credentials (referred to as [Actions secrets](https://docs.github.com/en/actions/security-guides/encrypted-secrets) by GitHub) to push an image to Docker Hub. Follow the [OTTR Project instructions](https://www.ottrproject.org/customize-docker.html#Set_Dockerhub_related_secrets)) to set DOCKERHUB\_USERNAME and DOCKERHUB\_TOKEN.



Update the .github/workflows/docker-management.yml file with your Docker Hub username and repository. For example, change jhudsl/reproducible-r to my\_username/my\_repository. Note that you need to change this on both line 53 and line 71.



This docker-management.yml file points Docker toward the correct Docker image file and links your GitHub repository with your Docker account.

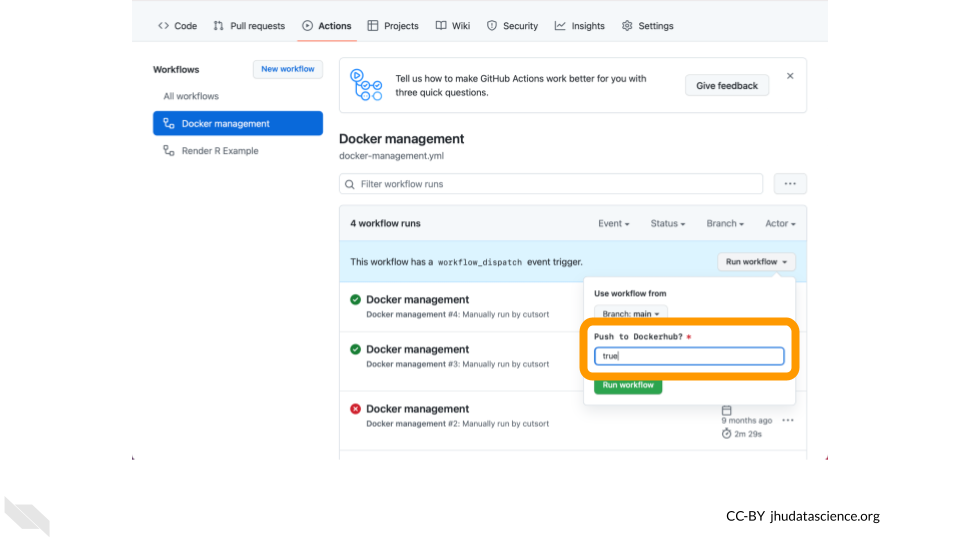
## 5.5 Build Docker image

The docker/Dockerfile file is the Docker image file, which is written in YAML syntax. It typically initializes the container with a base image that provides some basic dependencies, then installs additional software and dependencies as necessary. It may also provide metadata, set up scripts, and define commands to run when the container starts.

For a base image, we’ll start with [condaforge/mambaforge](https://github.com/conda-forge/miniforge-images) which is maintained by condaforge and provides conda on top of ubuntu-20.04. Replace the contents of the docker/Dockerfile file with the following to install [SAMtools via bioconda](https://anaconda.org/bioconda/samtools).

FROM condaforge/mambaforge  
  
RUN conda config \  
 --add channels defaults \  
 --add channels bioconda \  
 --add channels conda-forge  
  
RUN conda install -y samtools

Run the GitHub Action by navigating to the Actions tab, selecting Docker management, and clicking on Run workflow. Note that you must change Push to Dockerhub? to true.



You **must** change Push to Dockerhub? to true for your Docker image to be pushed to Docker Hub.

## 5.6 Use your image with the idxstats WDL

The next step is to modify the WDL script that you wrote in Chapter 4 to run in a container built by this Docker image. Edit your idxstats WDL script to call samtools idxstats on a bam file to specify your customized Docker image.

The result should look something like the version below, except that your version should point to the name of your own Docker image’s repository, rather than cutsort/test:

version 1.0  
workflow samtoolsIdxstats {  
 input {  
 File bamfile  
 }  
 call idxstats {  
 input:   
 bamfile = bamfile  
 }  
 output {  
 File results = idxstats.idxstats  
 }  
}  
  
task idxstats {  
 input {  
 File bamfile  
 }  
 command <<<  
 samtools index ~{bamfile}  
 samtools idxstats ~{bamfile} > idxstats.txt  
 >>>  
 output {  
 File idxstats = "idxstats.txt"  
 }  
 runtime {  
 docker: 'cutsort/test'  
 }  
}

The next step is to add this WDL script to the Broad Methods Repository in order to run the workflow in Terra.

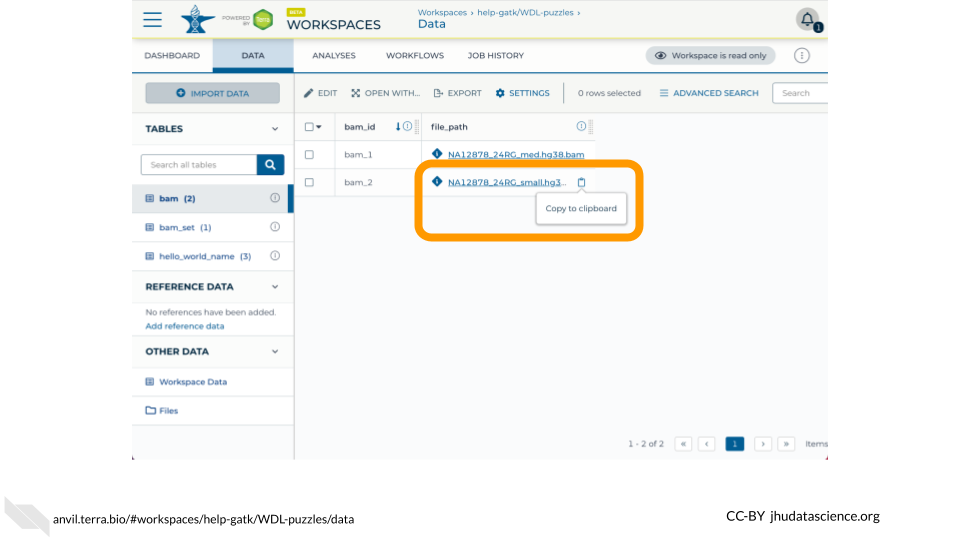
Follow the steps outlined in Chapter 3 to access the Broad Methods Repository and create a new method.

Then, copy or upload your WDL script into the new method and export it to your cloned version of the WDL-puzzles workspace.

## 5.7 Run idxstats workflow

Once you’ve exported the workflow to your AnVIL-powered-by-Terra workspace, open the workflow.

Select “Run workflow(s) with inputs defined by data table”, select the “bam” table, and select one row of the data table for a test run.



Fill in the column that stores the path to the input bam file (this.file\_path).

In the Outputs tab, give the output column a name by typing this.COLUMN\_NAME (for example, this.idxstats\_output).

Then click “run” to run the workflow, and monitor its progress. Once the workflow has finished running, you should see a new column in the “bam” table with a link to a .txt file with the QC metrics output by the workflow.

## 5.8 Best practices for using Docker

The WDL Analysis Research Pipelines (WARP) [GitHub page](https://github.com/broadinstitute/warp-tools/blob/develop/README.md) lists some recommendations for setting up Docker containers. Here are a few highlights:

* **Make sure that your image isn’t too large**, to avoid using unnecessary compute resources. Start with small images (e.g., Alpine) and include as few run steps as possible in the image file.
* Make dockers **publicly accessible**.

## 5.9 Further Reading

If you’re interested in a deeper dive into this chapter’s topics, check out these optional articles:

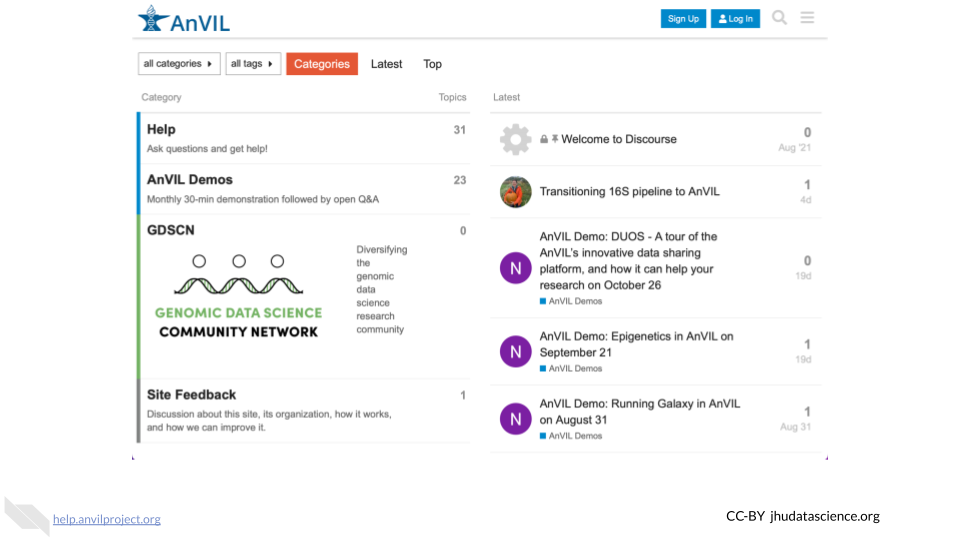
* To learn more about how to use Docker to create and store images, read [Docker/container overview](https://support.terra.bio/hc/en-us/articles/360037340472-Docker-container-overview).
* For information on using Docker to develop images locally, read [how to install docker and test that it works](https://support.terra.bio/hc/en-us/articles/360036000631-How-to-install-Docker-and-test-that-it-works).
* [How to run GATK in a Docker container](https://support.terra.bio/hc/en-us/articles/360036007791-How-to-run-GATK-in-a-Docker-container)
* [Docker Image Publishers’ Tips](https://support.terra.bio/hc/en-us/articles/4409141003547-Docker-Image-Publishers-Tips)

# 6 Join Discourse

Congratulations! You’ve practiced running WDL Workflows on AnVIL-powered-by-Terra, importing and configuring publicly available WDL workflows, writing an introductory WDL using the web-based editor at Broad Methods Repository, calculating idxstats across multiple genomic data files, and customizing your own Docker image.

Please take 5 minutes to let us know how we could improve this course at <http://broad.io/wdlBookSurvey>.

And please join the the conversation in the AnVIL Support Forums at <https://help.anvilproject.org> where you can ask any and all questions and we’ll do our best to help!



# About the Authors

These credits are based on our [course contributors table guidelines](https://github.com/jhudsl/DaSL_Course_Template_Bookdown/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 20.04.5 LTS   
## system x86\_64, linux-gnu   
## ui X11   
## language (EN)   
## collate en\_US.UTF-8   
## ctype en\_US.UTF-8   
## tz Etc/UTC   
## date 2023-11-15   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] RSPM (R 4.0.5)   
## bookdown 0.24 2023-03-28 [1] Github (rstudio/bookdown@88bc4ea)   
## cachem 1.0.7 2023-02-24 [1] CRAN (R 4.0.2)   
## callr 3.5.0 2020-10-08 [1] RSPM (R 4.0.2)   
## cli 3.6.1 2023-03-23 [1] CRAN (R 4.0.2)   
## 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 0.20 2023-01-17 [1] CRAN (R 4.0.2)   
## fastmap 1.1.1 2023-02-24 [1] CRAN (R 4.0.2)   
## fs 1.5.0 2020-07-31 [1] RSPM (R 4.0.3)   
## glue 1.4.2 2020-08-27 [1] RSPM (R 4.0.5)   
## htmltools 0.5.5 2023-03-23 [1] CRAN (R 4.0.2)   
## knitr 1.33 2023-03-28 [1] Github (yihui/knitr@a1052d1)   
## magrittr 2.0.3 2022-03-30 [1] CRAN (R 4.0.2)   
## memoise 2.0.1 2021-11-26 [1] CRAN (R 4.0.2)   
## 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 1.4.0 2020-10-07 [1] RSPM (R 4.0.2)   
## 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 1.1.0 2023-03-14 [1] CRAN (R 4.0.2)   
## rmarkdown 2.10 2023-03-28 [1] Github (rstudio/rmarkdown@02d3c25)  
## rprojroot 2.0.3 2022-04-02 [1] CRAN (R 4.0.2)   
## sessioninfo 1.1.1 2018-11-05 [1] RSPM (R 4.0.3)   
## stringi 1.5.3 2020-09-09 [1] RSPM (R 4.0.3)   
## stringr 1.4.0 2019-02-10 [1] RSPM (R 4.0.3)   
## testthat 3.0.1 2023-03-28 [1] Github (R-lib/testthat@e99155a)   
## usethis 1.6.3 2020-09-17 [1] RSPM (R 4.0.2)   
## withr 2.3.0 2020-09-22 [1] RSPM (R 4.0.2)   
## xfun 0.26 2023-03-28 [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