Bring Your Own Tool: FastQC Exercise

This exercise covers the basics of tool wrapping with Common Workflow Language (CWL) using Seven Bridges' Web Composer.

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y build-essential zlib1g-dev \
    libgsl0-dev wget unzip

RUN apt-get -y install software-properties-common

RUN apt-get update

RUN apt-get -y install openjdk-8-jdk openjdk-8-jre && apt-get clean

WORKDIR /opt

RUN wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.8.zip

RUN unzip fastqc_v0.11.8.zip && rm fastqc_v0.11.8.zip

RUN chmod 755 /opt/FastQC/fastqc

RUN ln -s /opt/FastQC/fastqc /usr/local/bin/fastqc

COPY Dockerfile /opt/

# Maintainer

MAINTAINER Phillip Brooks, Seven Bridges Genomics, <phillip.brooks@sbgenomics.com>
```

In order to follow this exercise, please download the <u>FastQC_SDK_Exercise.zip</u> archive file from the "training resources" project and unpack it on your local machine.

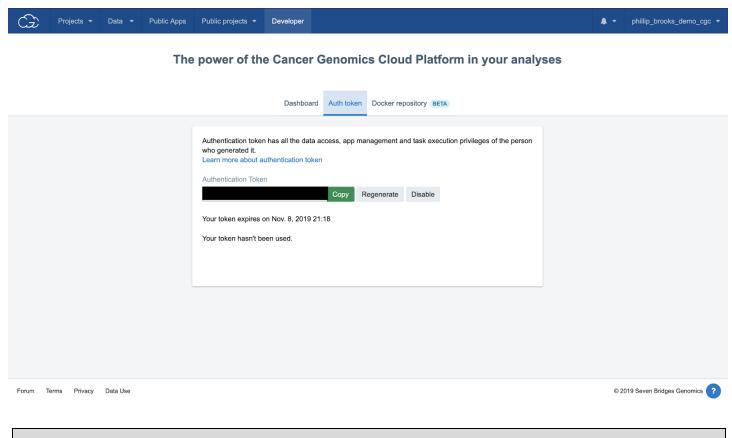
CREATE THE DOCKER IMAGE

 We will use the provided Dockerfile to build an image (feel free to edit the MAINTAINER info to your own and Save As "Dockerfile").

Let's have a quick run through of what the commands in this Dockerfile will be doing:

- First, we're defining the base image as Ubuntu version 16.04 from Dockerhub. You could also choose ubuntu:latest as the base image, but this is not recommended, as it breaks reproducibility over time.
- Next, we're fetching some required packages, including Java (OpenJDK), we'll need for installing the FastQC tool.
- Finally, we're changing into "/opt/" as the working directory, downloading and installing the FastQC software and copying the Dockerfile used to build the image into "/opt/" as well to store a record of how image was created.
- 2. With docker running on your local machine, open the terminal and login to the Seven Bridges image repository (use your platform username and developer Authentication Token as password):

You may obtain your authentication token via the <u>Developer tab</u> on the Seven Bridges Platform.



docker login cgc-images.sbgenomics.com

username = username Password = authentication token

4. Run the docker build command to build the image (please enter the command as a single line):

docker build -t cgc-images.sbgenomics.com/<username>/fastqc:0.11.8 .

where <work_dir> is the directory containing the Dockerfile. Please note, when naming your docker tag URL, take care to avoid spaces and special characters such as ", \$" etc.

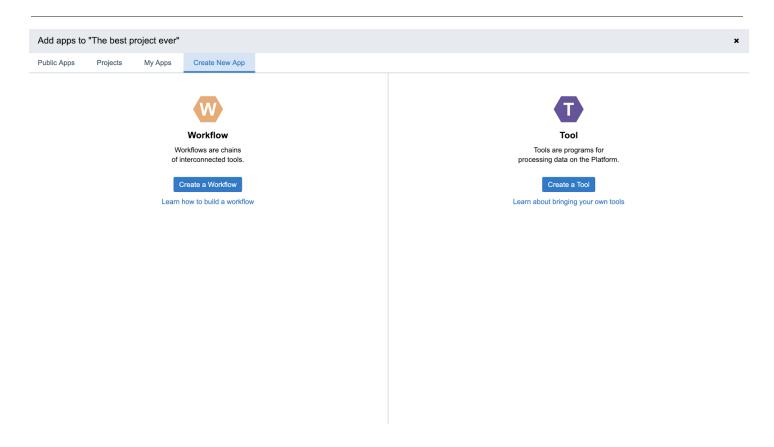
```
(base) $ docker build -t cgc-images.sbgenomics.com/phillip_brooks_demo_cgc/fastqc:0.11.8 .
Sending build context to Docker daemon 922.3MB
Step 1/12 : FROM ubuntu:16.04
 ---> 13c9f1285025
Step 2/12 : RUN apt-get update && apt-get install -y build-essential zlib1g-dev
                                                                                     libgsl0-dev wget unzip
  --> Using cache
   -> 0d553b21cc73
Step 3/12: RUN apt-get -y install software-properties-common
 ---> Using cache
   -> abd93490d755
Step 4/12: RUN apt-get update
  --> Using cache
 ---> dbeb3d1ca0b1
Step 5/12 : RUN apt-get -y install openjdk-8-jdk openjdk-8-jre && apt-get clean
 ---> Using cache
 ---> a4ed9b4eecb0
Step 6/12 : WORKDIR /opt
   -> Using cache
 ---> 3d383d1a1e1e
Step 7/12: RUN wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.8.zip
   -> Using cache
   -> 84f0c7fae837
```

4. Push the image to the SB repo:

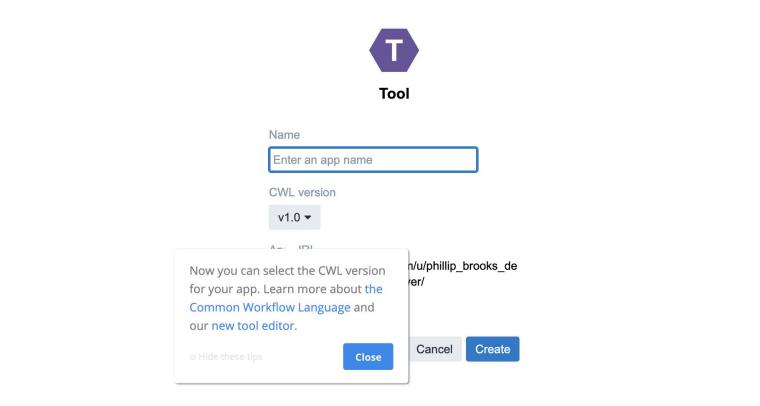
```
docker push cgc-images.sbgenomics.com/<username>/fastqc:0.11.8
```

CREATE THE CWL WRAPPER USING WEB COMPOSER

- 1. Navigate to the "Apps" tab and click "+ Add app", "Create New App".
- 2. Click "Create New App" and then the green "Create a Tool" button.



3. Name your tool "FastQC", select "v.1.0" and click "Create".



4. You will now see the Web Composer interface

The Web Composer interface has three working tabs: "App Info", "Visual Editor" and "Code". We will use the Visual Editor to construct the desired command line step by step and create a CWL description for this tool. Alternatively, you may directly copy the raw CWL from the "fastqc.cwl" file in the file package you have downloaded earlier and paste this in the "Code" tab. Doing so will generate an already completed FastQC wrapper for your convenience. If you are following this route, you may read through the following steps to understand how this wrapper was created, and push this tool to the Platform using the steps described within "PUSH YOUR TOOL TO THE SEVEN BRIDGES PLATFORM" section.

5. FastQC tool has a number of input and output parameter settings. Rather than go through every single one, we will add only a few of the parameters, and define required Base Command, Arguments, Inputs and Outputs. The desired command line we are aiming to build is as follows:

```
fastqc --noextract --outdir . /path/to/input-1.ext /path/to/input-2.ext
```

Let's look at this FastQC command line in detail to understand what each portion means first:

- fastqc: This is our "base command". In other words how we call the main executable of the program we are wrapping.
- --noextract: This is an "argument" to prevent FastQC from creating separate folders for each
 of the input samples. For this exercise we are only interested in zipped FastQC results and
 HTML reports.
- --outdir.: This is another "argument" of FastQC to ensure output files are written to the
 execution directory. The Platform can only pick output files that are present in the execution
 directory.
- /path/to/input-1.ext and /path/to/input-1.ext: These are "inputs" of the FastQC tools. You see them filled with random file paths. When you execute a task, they are replaced with actual file paths of the input files.

Now that we have understood how to execute FastQC, we can start creating our CWL wrapper.

6. Let's start by setting the Docker Image:

Use the image URL you pushed to the SB repository, for example:

cgc-images.sbgenomics.com/<user-name>/<image_name>:<tag>



7. Add BASE COMMAND: Click the "+ Add Base Command" button and type in:





8. Add ARGUMENT: Click the "Add an Argument" button and configure new argument as following:

Use command line binding: YES

Prefix: (leave empty)
Value: --noextract

Separate value and prefix: YES or NO

Position: 0

*Note: We want this argument to be placed first after the end of the base command.

Arguments differ from "inputs", as arguments are not directly open to the user manipulation during task creation. You may utilise arguments to lock down fundamental aspects of your tool execution. For instance, in this case we want to force FastQC to adopt a certain behaviour: never create result directories, rather keep results in compressed format to reduce file cluttering in the project.

Let's look at the settings we can define for arguments:

- **Use command line binding:** This Yes/No switch determines whether the specified argument should be included in the generated command line.
- **Prefix:** Prefix field allows tool wrapper to define the argument prefix.
- Value: This field allows tool wrapper to define the value that will come after the prefix. Tool
 wrappers may fill this field with fixed values, or create a JavaScript expression using "</>
 button for dynamic generation of the argument value. You may visit <u>Seven Bridges Knowledge</u>
 Center to learn more about dynamic expressions generated with JavaScript.
- **Separate value and prefix:** This Yes/No switch determines whether "prefix" and "value" should be separated with a space in the command line.
- **Position:** This field determines the position of the argument within the command line. You may set this as 0 to position the argument at the first place after the base command, or a very large number, such as 99, to position the argument at the very end of the command line.



For this argument, we have left the prefix empty as it is a boolean type argument and it does not take an actual value. Since we want this flag to be added to the generated command line under all circumstances, we have added the "--noextract" argument directly as a value.

Next, let's specify another Argument.

9. Add ARGUMENT: Click the "+ Add an Argument" button and configure as following:

Use command line binding: YES

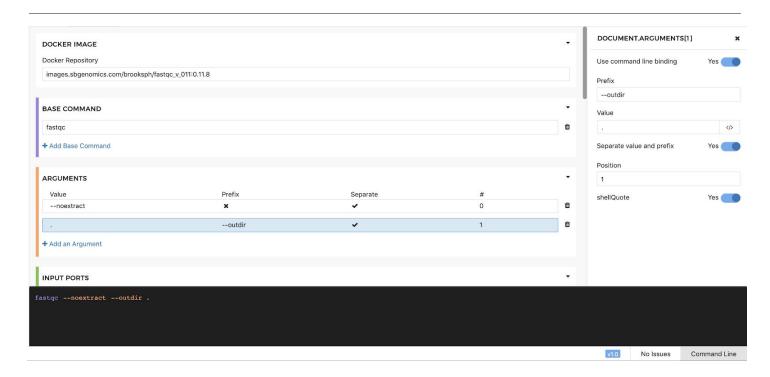
Prefix: --outdir

Value: .

Separate value and prefix: YES

Position: 1

This time we have set a prefix and a value, since this is an argument that takes a value. Note how the sample command line changes as we add new arguments:



Now that we have completed our two arguments, we can start adding inputs.

10. Add INPUT PORT: Click the "Add an Input" button and configure as following:

Required: YES ID: input_files Type: array Items Type: File

Include in the command line: Yes

Value Transform:

Prefix:

Position: 100 (or any large number, we want this to be placed at the end of the command line)

Separate value and prefix: Yes

Item Separator: None Stage Input: --none--

shellQuote:

Load Content: No

Add secondary file: No Secondary Files defined.

Label: Input FASTQ Files

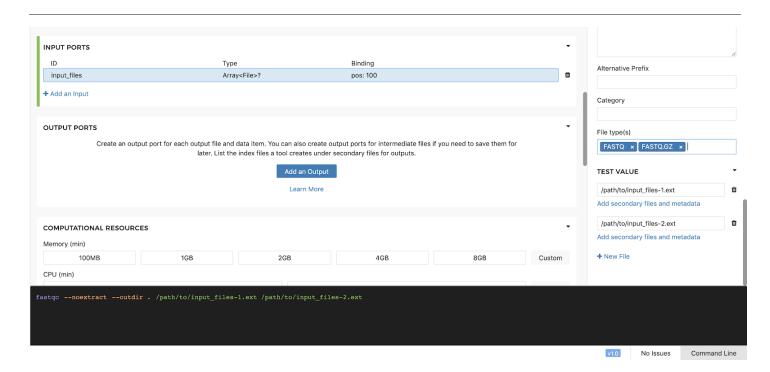
Description: Input FASTQ Files.

Alternative Prefix:

Category:

File type(s): FASTQ, FASTQ.GZ

You may find more information about how to define the input settings at the <u>Seven Bridges Knowledge</u> Center.



9b. (optional) Add another INPUT PORT: Click "+ Add an Input" and set the following fields in the inspector panel:

Required: NO ID: threads Type: int

Allow array as well as a single item: No

Include in the command line: Yes

Value Transform: Prefix: --threads
Position: 3

Separate value and prefix: Yes

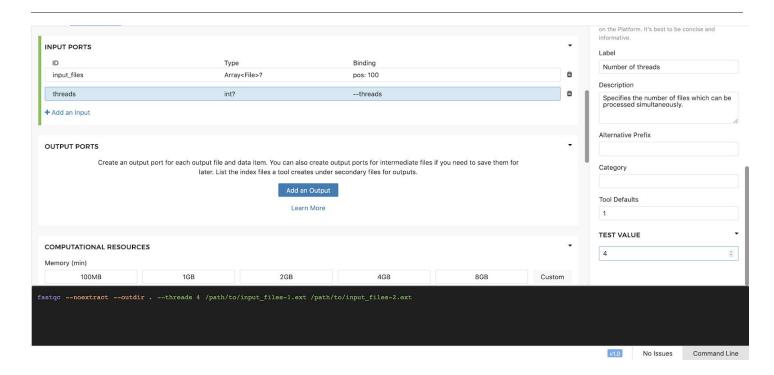
Label: Number of Threads

Description: Specifies the number of files which can be processed simultaneously.

Alternative Prefix:

Category:

Tool Defaults: 1



Now that we have specified all inputs and arguments defined in the target command line structure shown in Step 4, we may start defining the outputs that should be collected after the execution of the FastQC. When run with "--noextract" argument FastQC creates two files per sample:

- A ZIP file that contains FastQC report, graphics, and summary files.
- An HTML report showing the FastQC results.

Let's create output ports for both types of outputs.

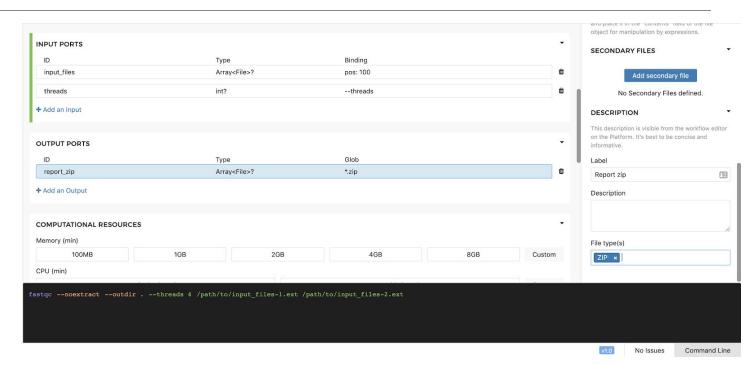
11. Add an OUTPUT PORT: Click on "**Add an Output**" button and set the following fields in the inspector panel:

Required: No ID: report_zip Type: array Items Type: File Glob: *.zip

Inherit:
Output eval:
Load content: NO

No Secondary Files defined

Label: Report zip Description: File type(s): ZIP



You may learn more about the output port settings at the <u>Seven Bridges Knowledge Center</u>. Please note that adding an output port does not cause a change in the sample command line generated at the bottom of the screen.

Now you can add our second output port for the HTML reports that you can display on the Platform.

12. Add an OUTPUT PORT: Click on "+ Add an Output" button and set the following fields in the

inspector panel: Required: No

ID: report_html

Type: array

Items Type: File

Glob: *.html

Inherit:

Output eval:

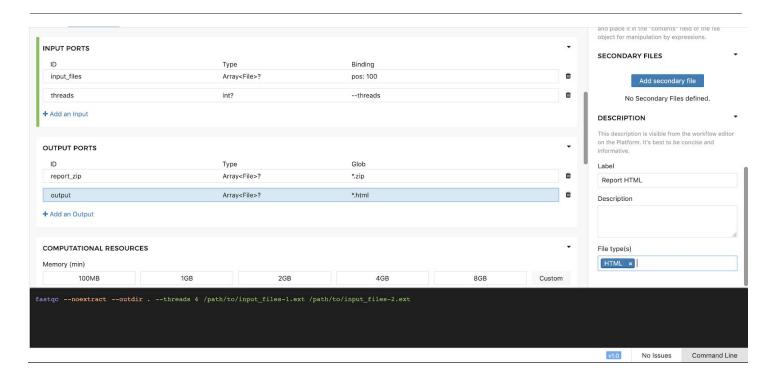
Load content: NO

No Secondary Files defined

Label: Report HTML

Description:

File type(s): HTML

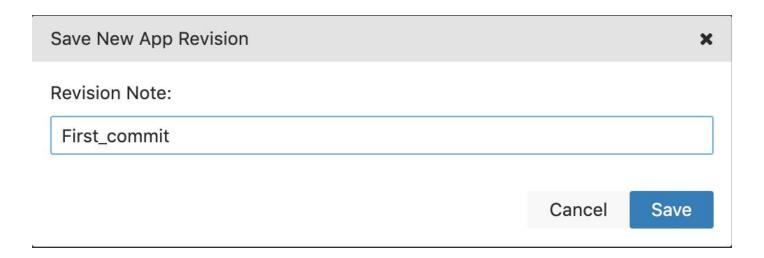


13. With this output port, you have completed wrapping your first tool.

PUSH YOUR TOOL TO THE SEVEN BRIDGES PLATFORM

1. After you've finished wrapping your tool, you may save your tool on the Platform by creating a new revision.





You will now notice that a new tab appears that displays the app version that you just pushed to the Platform project (i.e, The Greatest Project Ever project)

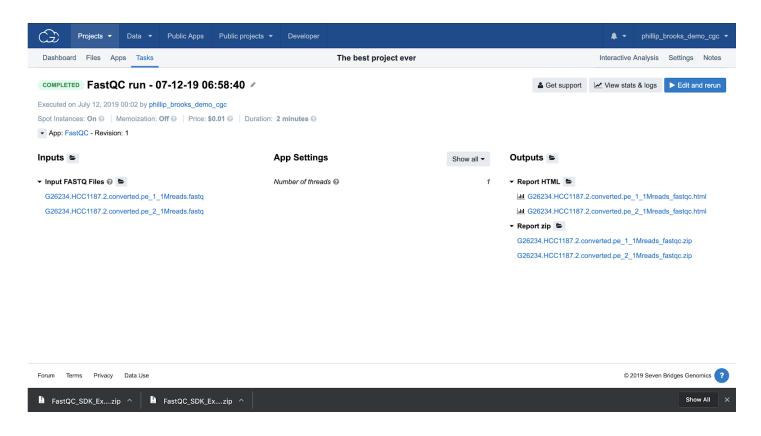
9 1

BONUS: RUN YOUR TOOL ON THE PLATFORM

Now that you've pushed your App to the platform, you can click the arrow button to create a draft task:



To test this App, you can copy the files "G26234.HCC1187.2.converted.pe_2_1Mreads.fastq" and "G26234.HCC1187.2.converted.pe_1_1Mreads.fastq" from the Public Reference Files to the Project you just pushed the app to and run the tool on the cloud.



	SevenBridges
Nice work! :)	