1 Introduction

AlgoRun is a docker-based software container template designed to package computational algorithms. This document shows steps of how to create an AlgoRun container of an implemented algorithm.

2 Download and Install Docker

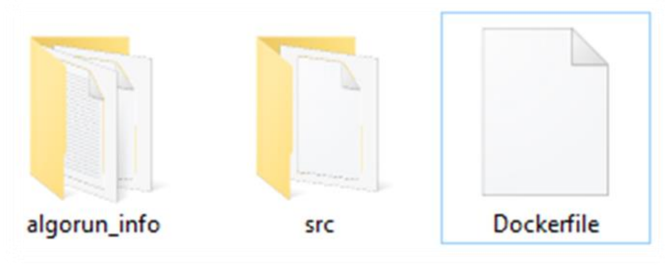
Before starting, download and install Docker on your local machine. Docker can be installed on Mac OS, Linux as well as Windows. Follow the instructions on: https://docs.docker.com/v1.8/installation/

3 Download AlgoRun

Download AlgoRun from: https://github.com/algorun/skeleton

4 Create an AlgoRun Container

Unzip the downloaded skeleton-master.zip file. The resulting folder has the following structure:



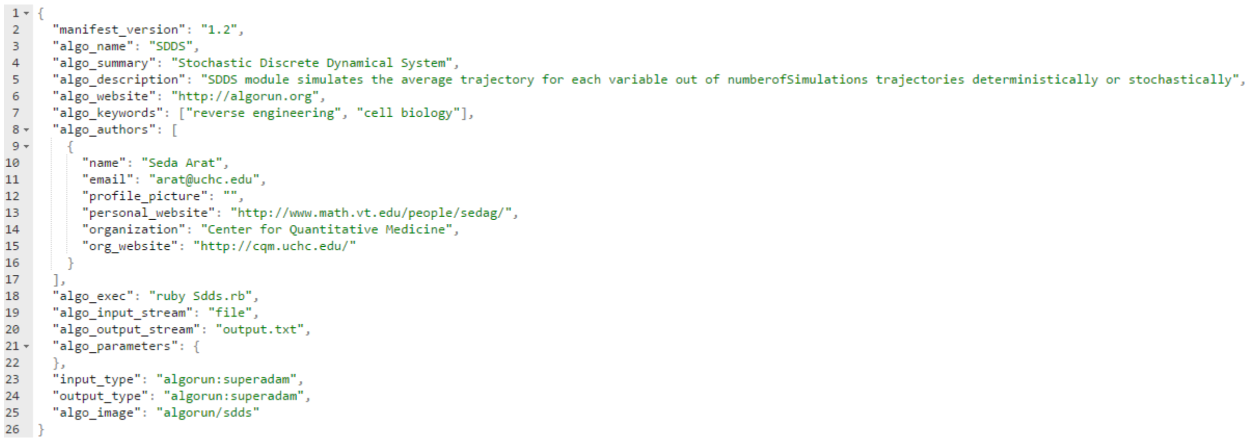
- A Dockerfile is a text document that contains all commands needed to build the software container. Docker builds a software container automatically by reading the instructions from the Dockerfile.
- AlgoRun template container uses the `src` and `algorun_info` folders to deposit all your source code and to describe the implemented algorithm in a standard format.

STEP 1: Add all source code files of your algorithms into the `src` folder.

STEP 2: Edit the `Dockerfile` to make sure your algorithm dependencies will get installed in the container. AlgoRun is based on Ubuntu 15.10 Linux system so you can leverage Ubuntu packaging system to get your dependencies installed.

For more information about Dockerfile, please refer to the Docker documentation (https://docs.docker.com/v1.8/reference/builder/).

STEP 3: Edit the `manifest.json` file inside the `algorun_info` folder. Below is an example of the manifest file.



```json
{
  "manifest_version": "1.2",
  "algo_name": "SDDS",
  "algo_summary": "Stochastic Discrete Dynamical System",
  "algo_description": "SDDS module simulates the average trajectory for each variable out of numberofSimulations trajectories deterministically or stochastically",
  "algo_website": "http://algorun.org",
  "algo_keywords": ["reverse engineering", "cell biology"],
  "algo_authors": [
    {
      "name": "Seda Arat",
      "email": "arat@uchc.edu",
      "profile_picture": "",
      "personal_website": "http://www.math.vt.edu/people/sedag/",
      "organization": "Center for Quantitative Medicine",
      "org_website": "http://cqm.uchc.edu/"
    }
  ],
  "algo_exec": "ruby Sdds.rb",
  "algo_input_stream": "file",
  "algo_output_stream": "output.txt",
  "algo_parameters": {
  },
  "input_type": "algorun:superadam",
  "output_type": "algorun:superadam",
  "algo_image": "algorun/sdds"
}
```

*Source code: an example of **manifest.json** file*

In addition to adding algorithm's information, the following fields are necessary for AlgoRun to correctly execute the algorithm source code:

- "`algo_exec`": is the command used to start algorithm executed.
- "`algo_input_stream`": is how the algorithm reads the input data. Values can be one of the following:
  - `direct`: if the algorithm input is passed directly as the first parameter in the command line.
  - `file`: if the algorithm reads input from a file. File name is then passed as the first command line argument.
  - `stdin`: if the input is passed through the standard input stream using '<' operator.
- "`algo_output_stream`": is the path of the file where the algorithm outputs its result or `stdout` if the algorithm prints the result to the standard output stream.

Command line options can be exposed in the "`algo_parameters`" field (Refer to examples document for a detailed example using parameters). AlgoRun website uses "`input_type`" and "`output_type`" to easily identify algorithms that can communicate together. Please refer to http://algorun.org/input-output-types to see what input and output types you should use. Users can also download and use the algorithm Docker image locally from Docker Hub if the value "`algo_image`" is provided.

STEP 4: Provide input and output examples in the `input_example.txt` and `output_example.txt` files respectively.

STEP 5: Build the algorithm container from the command line using `docker build` command.

Example: `docker build -t <algorithm_name> .`

5 Run the algorithm

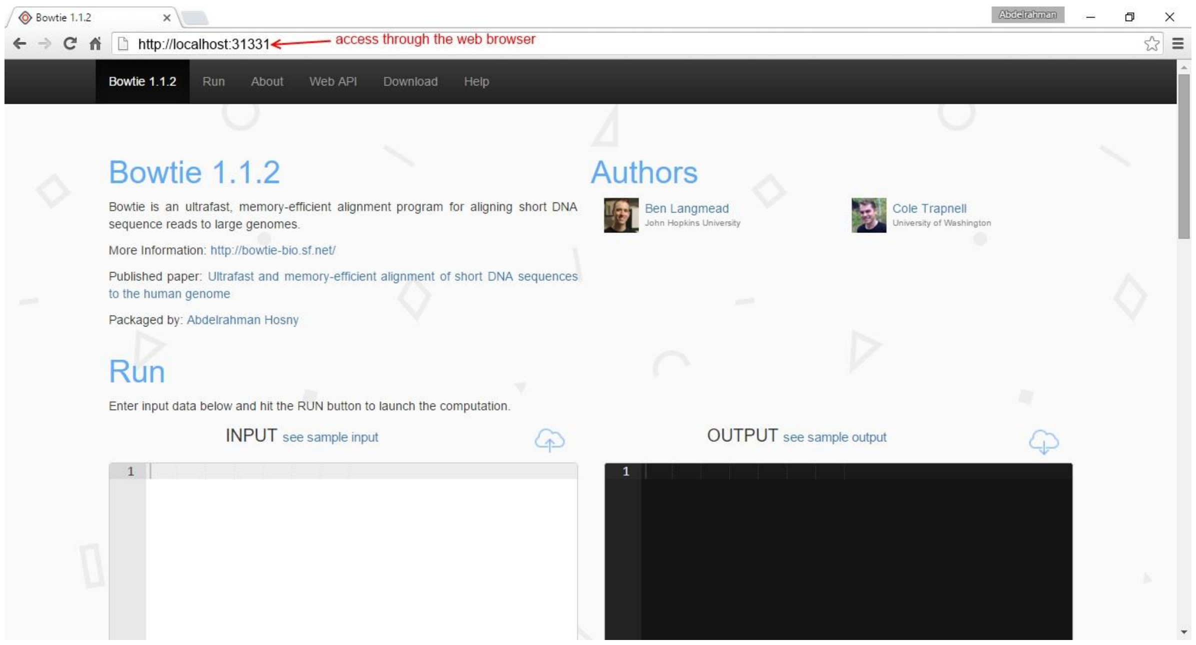
Once the algorithm container has been created, it must first be deployed before the user can start using the packaged algorithm. The container can be deployed with the following command:

```
docker run -p 31331:8765 --name <container_name> <algorithm_name>
```

Now that the algorithm container has been deployed on your local machine (localhost), AlgoRun provides the user with three different ways to run the algorithm.

5.1 Web User Interface

The easiest and quickest way to run the packaged algorithm is to open the web browser and type http://localhost:31331



Bowtie 1.1.2

Bowtie is an ultrafast, memory-efficient alignment program for aligning short DNA sequence reads to large genomes.

More Information: http://bowtie-bio.sf.net/

Published paper: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome

Packaged by: Abdelrahman Hosny

Authors

Ben Langmead, John Hopkins University

Cole Trapnell, University of Washington

Run

Enter input data below and hit the RUN button to launch the computation.

INPUT see sample input

OUTPUT see sample output

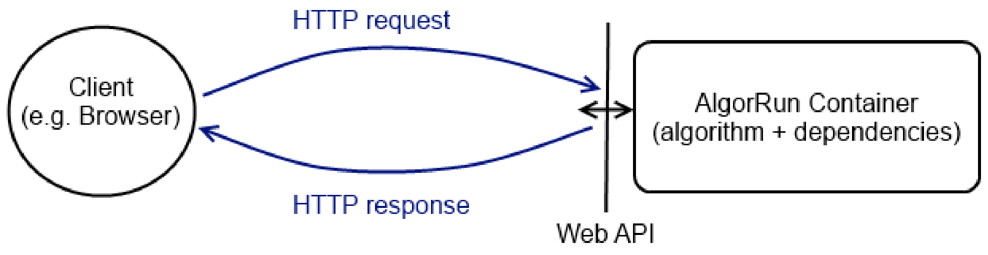
The web user interface of an algorithm packaged with AlgoRun.
*Type in the address **http://localhost:31331** in a web browser to open the web page of the running algorithm container.*

5.2 Web API

A web API is an Application Programming Interface (API) used to offer programmatic access to remote resources or services (in our case "computations from an algorithm") that can be accessed by clients such as web browsers or any http-enabled third-party applications. AlgoRun containers are pre-included with a RESTful API[1] that allows access to the computation through the traditional HTTP POST request. Clients communicate to web APIs through a request/response protocol. To ask the web service to perform a computation, a client sends an HTTP request. The body of the request

[1] REpresentational State Transfer (REST) APIs uses Hyper Text Transfer Protocol (HTTP) requests as the main scheme of communication.

includes necessary input data for the algorithm behind the web service to start. The response of the web service includes the result of the computation.



AlgoRun Web API Communication Scheme

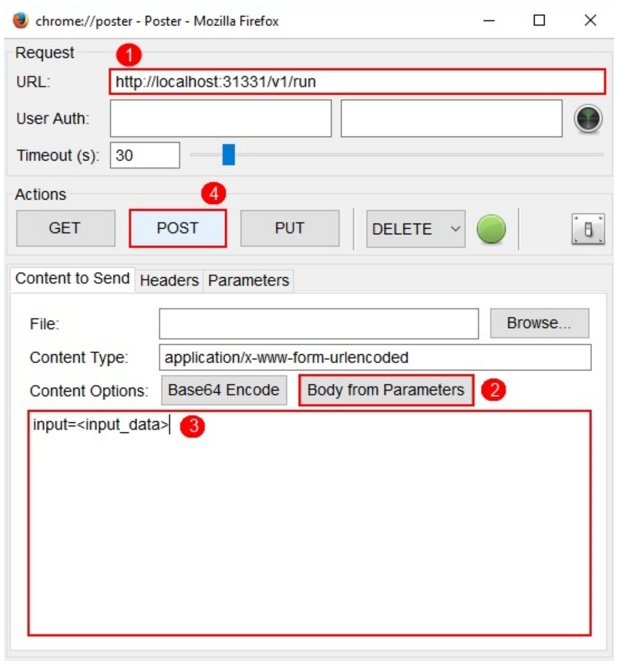
The HTTP request should be sent to a specific address, which is called an endpoint. The two main endpoints exposed by AlgoRun containers are shown below.

Endpoint	Usage	Request Parameters	Response
HTTP POST /v1/run	used to run the algorithm on a given input data	**input**: <input_data>	The result of the computation in the body of the response
HTTP POST /v1/config	used to dynamically change the parameters values	**<parameter_name>**: <parameter_value>	The result of changing the parameter value

By offering standardized input and output, Web APIs are particularly useful when it comes to building complex software applications as they make it easy to integrate different algorithms that usually run on different programming environments. It enhances the modularity of the software, hence increases its robustness and makes troubleshooting problems easier. You can embed computations in large software programs in just a few lines of code, removing the hassle of installing the whole algorithm environment locally.

To test a web API, Firefox Poster Plugin[2] is a graphical user interface tool used to easily send and troubleshoot HTTP requests. See below for examples on using the above AlgoRun endpoints.

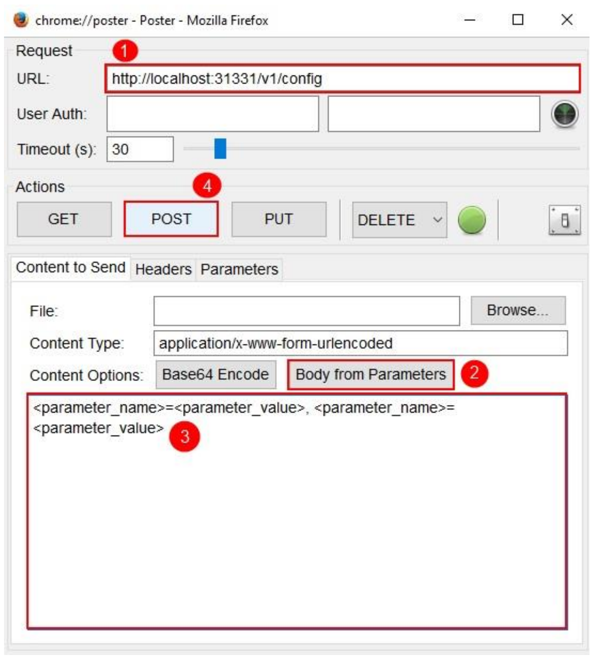
[2] Install it on Firefox browser: https://addons.mozilla.org/en-US/firefox/addon/poster/



URL: http://localhost:31331/v1/run
Timeout (s): 30
Content Type: application/x-www-form-urlencoded

```
input=<input_data>
```

Calling Web API using Firefox Poster Plugin: *(1) Type the URL of the endpoint http://localhost:31331/v1/run (2) Select Body from Parameters. (3) Type input=<paste_your_input_data_here>. (4) Click Post to initiate the request.*



URL: http://localhost:31331/v1/config
Timeout (s): 30
Content Type: application/x-www-form-urlencoded

```
<parameter_name>=<parameter_value>, <parameter_name>=
<parameter_value>
```

Configuring Parameters using Web API: *(1) Type the URL of the endpoint http://localhost:31331/v1/config (2) Select Body from Parameters. (3) Type <parameter_name>=<parameter_value>. (4) Click Post to initiate the request.*

5.3 Command Line

The traditional command line execution is still available as well.

```
docker exec -i <container_name> /bin/algorun < sample_input.txt
```

6 Publish your algorithm to the AlgoRun website

If you packaged your algorithm with AlgoRun and want to give your algorithm more visibility, we encourage you to submit it for listing on the AlgoRun website. The AlgoRun website serves as a repository for all computational algorithms that were packaged using AlgoRun: http://algorun.org

To submit your algorithm for listing, fill the form located at http://algorun.org/submit- algorithm

7 Examples

For complete examples, please refer to examples document.