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 Docker

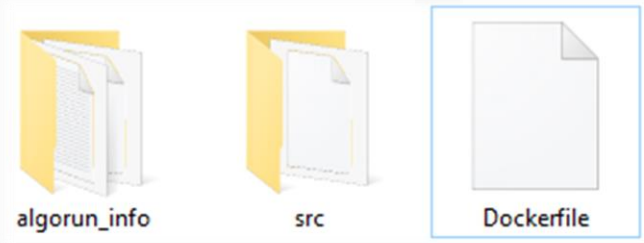
Before starting, download and install Docker on your local machine. Docker can be installed on Mac OS, Linux as well as Windows. Follow the instructions on: https://docs.docker.com/v1.8/installation/

3 Download AlgoRun

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

4 Steps to Create an AlgoRun Container

Unzip the downloaded skeleton.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.

```
{
  "manifest_version": "1.3",
  "algo_name": "Bowtie 1.1.2",
  "algo_summary": "Bowtie is an ultrafast, memory-efficient alignment program for aligning short DNA sequence reads to large genomes.",
  "algo_description": "Bowtie is an ultrafast, memory-efficient short read aligner geared toward quickly aligning large sets of short DNA sequences (reads) to large genomes. Check <a href
='http://bowtie-bio.sf.net/' target='_blank'>our website</a> for detailed explanation. This interface is meant to provide a quick and easy access to the computation without having to install
Bowtie packages. Command line options are exposed as parameters, which you can configure from the above window.",
  "algo_website": "http://bowtie-bio.sf.net/",
  "algo_keywords": ["bowtie", "DNA", "genome", "sequencing", "alignment", "Burrows-Wheeler", "indexing"],
  "published_paper": {
          "title": "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome",
          "url": "http://www.genomebiology.com/2009/10/3/R25"
  },
  "algo_authors": [
    {
      "name": "Ben Langmead",
      "email": "langmead@cs.umd.edu",
      "profile_picture": "ben.jpg",
      "personal_website": "http://www.cs.jhu.edu/~langmea/",
      "organization": "John Hopkins University",
      "org_website": "https://www.jhu.edu/"
    },
    {
      "name": "Cole Trapnell",
      "email": "coletrap@uw.edu",
      "profile_picture": "cole.png",
      "personal_website": "http://cole-trapnell-lab.github.io/team/cole-trapnell/",
      "organization": "University of Washington",
      "org_website": "http://www.uw.edu/"
    }
  ],
  "algo_packager": {
      "name": "Abdelrahman Hosny",
      "email": "abdelrahman.hosny@hotmail.com",
      "personal_website": "http://abdelrahmanhosny.com",
      "organization": "University of Connecticut",
      "org_website": "http://uconn.edu"
    },
  "algo_exec": "ruby bowtie.rb",
  "algo_input_stream": "direct",
      "algo_output_stream": "stdout",
  "algo_parameters": {},
  "input_type": "algorun:dna-sequence",
  "output_type": "algorun:aligned-dna-sequence",
  "algo_image": "algorun/bowtie"
}
```

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

In addition to adding algorithm's information, the following fields are necessary for AlgoRun to correctly run 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 Supplementary File 2 for a complete example). 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 using `docker build` command.

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

Publish your algorithm to the AlgoRun website

If you packaged your algorithm with AlgoRun and want to give your algorithm more visibility, do not hesitate 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