TReNDs-Setter

A Docker Specification for NeuroImaging at TReNDs

Version 0.0.0 (Pre-Alpha)

Dockerized Application

The *dockerized application* is the internal library, binary, or environment which is built within the image for the purpose of running some kind of analysis. The environment itself is created using the **Dockerfile** associated with the image. *All dependencies for the internal application must be installed on the image itself*, i.e. no external libraries or environments from the host machine should be required in order to run the application in its entirety.

Analysis Input Data.

Analysis Input Data consists of a set of neuro-imaging data and associated metadata which are input to the dockerized application for analysis. The data consist of a set of folders on the disk which are mounted into the docker image, according to the **BIDS specification**. All subject input data should be mounted into **logical locations** on the target image, which are treated as **READ ONLY** by the internal application.

The locations of analysis input data within the image should be provided *as a part of the analysis input parameters*, i.e. there should be at least one field where paths are provided to the image if necessary.

Mounting data to the image should be done *minimally and logically*, i.e. in a way which minimizes the number of distinct folders mounted to the image, and to locations on the image which uniquely identify input folders from other folders extant in the image

or on the host machine. See the section below on folder mounting for additional recommendations.

Analysis Input Parameters

Analysis input parameters consist of a set of arguments which are sent into the image in order to run an application. These arguments should be input as **JSON** files to the target image, which the *application* itself, or the *run script/wrapper* can parse. The format of the **JSON** parameters should follow a logical structure such that it can be parsed directly into a **python list or dictionary** without additional external processing (*note: if an internal application accepts JSON input, that input specification should be followed first*).

Run Script/Wrapper

The dockerized application should be runnable entirely by **a SINGLE** external command line call to a *run script or wrapper*. If the dockerized application itself takes JSON input, and can be run directly from the command line, no additional run script or wrapper may be necessary; however, if the application takes many different kinds of arguments, and cannot parse JSON itself, an additional script should be created which works with JSON input and formulates the corresponding command to run within the image.

A *Run Script* can be a bash (or other shell) script, which takes JSON input from the command line, and parses it into a command which runs the dockerized application according to the user's specifications. This script should be able to handle ***any and all kinds of input which are normally provided to the application***, and should require no additional input for its own arguments other than the JSON file. The *run script* may do additional work such as parsing the BIDS input folder to find relevant files, but this additional work should require no extra input from the user beyond what is provided in the Analysis Input Parameters. The *run script* should be fully documented to include explanations for how JSON input is parsed for the dockerized application, as well as any documentation of additional work done by the script.

A *wrapper* provides equivalent functionality to a run-script (running an application from the command line with minimal input), while also creating a logical API to run the dockerized application via imports to a given scripting environment (such as python). The use of python wrappers is encouraged, as it allows applications to run based off a minimal image where python has been installed. All wrappers should follow appropriate style specifications such as Pep8.

Use of the wrapper, such as allowed arguments and defaults, must be documented in the **README.md** for the image.

Base Image Template

The base image will utilize Alpine, a lightweight linux docker-image which is about a quarter the size of the standard ubuntu image. The base image will include the latest **python** installation in **miniconda 3**, as well as common software dependencies and utilities. The current list of software in the base image is: **gcc**, **g++, cmake, wget, curl, vim, emacs, tmux, node,** and **screen.** The following unix libraries will also be installed: **libx11, libxcomposite, libxcursor, libxdamage, libxext, libxfixes, libxt, libxtst, libxxf86vm, libasound2, libatk1.0, libcairo2, libsndfile1, libxcb1, libxslt…**

The base image implements **Flywheel hooks,** a **minimal python wrapper,** a **pybids** installation for checking input, and example **bash scripts** for parsing input from BIDS using **pybids**.

Matlab-Runtime Image

Built on top of the **Base Image**, the Matlab Compiler Runtime image template bundles the **MATLAB R2022a** runtime in addition to the packages mentioned above.

Additional Templates to be Enumerated Here

TODO

Application Folder Structure/Documentation

It is recommended that all application binaries, dependencies, and other files required for the installation be placed in a single folder named **/app.** This folder should include the **Dockerfile** as well as a **README.md** and any **LICENSES** relevant to the application. The **README.md** should list any instructions for running the base image in docker**,** and should give a quick description of the design of the docker image, as well as any design decisions made. A reference to this template, or to another template used will also be helpful.

A recommended final command in the Docker image is `ADD . /app` which will add the CWD where the image is built, and thus include the README and any local dependencies.

Home Folder and Folder Mounting

The docker image should be entirely isolated from an external folder structure with the exception of input and output directories which are mounted to **/input** and **/output** respectively (allowing for minor variation as necessary). The **$HOME** directory within the docker image (defaulted to **/home/**…) should be avoided, and replaced with another working directory**,** such as the output directory **/output**.

Additional mounted folders beyond those specified for input/output need to be clarified in the **README.md** for the docker image, and ***should not introduce any dependencies*** to the image such as binaries which are utilized as part of the analysis.

Github and Docker Hub

All images should have a unique github repo placed in the group:
https://github.com/trendscenter

It is ***highly recommended*** that the user maintaining the github repo utilize Branching and Pull-Requests for updating the repository, rather than making pushes directly to master. Pushes to the repo should follow semantic versioning (see below) as appropriate

All images should also be hosted on the TreNDsCenter docker hub:
https://hub.docker.com/orgs/trendscenter/repositories

Images pushed to docker hub should have tags indicating their **Semantic Version** (see below). The **latest** tag should only point to the latest version.

Versioning

All docker images pushed to the TReNDsCenter docker hub should use Semantic Versioning. I.e. each version tag should have three numerical values: v0.0.0, where the first value indicates the **Major Version** (incompatible API changes), the second value indicates the **Minor version** (backwards-compatible functionality changes), and the third value indicates the **Patch version** (backwards-compatible bug fixes).

Compatibility with Singularity

It is recommended that all built docker images be compatible with **Singularity (v3.10.2).** If the image is hosted on Docker-hub, the following commands will allow the image to be built with Singularity on the cluster:

```
$ module load singularity
$ export SINGULAIRTY_DOCKER_USERNAME=<your username>
$ export SINGULARITY_DOCKER_PASSWORD=<your password>
$ singularity build <image_name>.img \
      docker://trendscenter/<image_name>:<tag>
```

Previous Spec Versions

- **0.0.0 - Pre-Alpha [Current]** (*11/15/2022*)

Base Template Versions

- **0.0.0 - Pre-Alpha [Current]** (*11/16/2022*)

Matlab-Runtime Versions

- **0.0.0 - Pre-Alpha [Current]** (*11/16/2022*)