DockFlow: Bioconductor Workflow Containerization and Orchestration with liftr

Nan Xiao[1,*], Tengfei Yin[1], Miaozhu Li[2] [1]Seven Bridges Genomics [2]Duke University *me@nanx.me

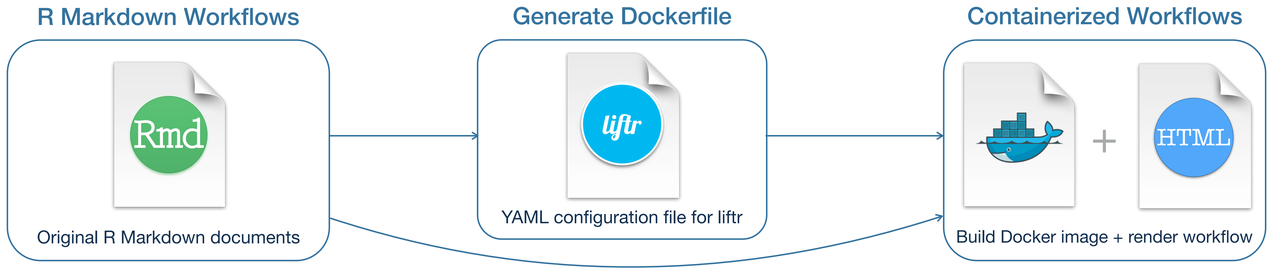
Fork me on GitHub

ABSTRACT

We have accumulated numerous excellent software packages for analyzing large-scale biomedical data on the way to delivering on the promise of human genomics. Bioconductor workflows (https://bioconductor.org/help/workflows/) illustrated the feasibility of organizing and demonstrating such software collections in a reproducible and human-readable way. Going forward, how to implement fully automatic workflow execution and persistently reproducible report compilation on an industrial-scale becomes challenging from the engineering perspective. For example, the software tools across workflows usually require drastically different system dependencies and execution environments and thus need to be isolated completely. As one of the first efforts exploring the possibility of bioinformatics workflow containerization and orchestration using Docker, the DockFlow project aims to containerize every existing Bioconductor workflow in a clean, smooth, and scalable way. We show that with the help of our R package liftr, it is possible to achieve the goal of persistent reproducible workflow containerization by simply creating and managing a YAML configuration file for each workflow. We will also share our experience and the pitfalls encountered during such containerization efforts, which may offer some best practices and valuable references for creating reproducible bioinformatics workflows in the future. The DockFlow project website: https://dockflow.org.

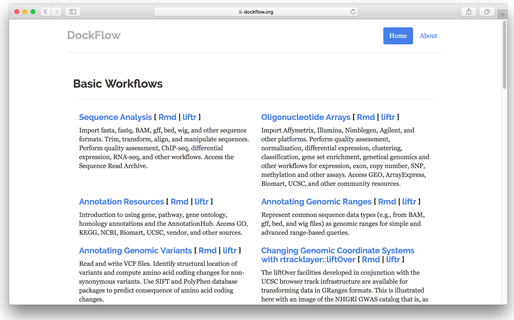
METHOD

To containerize the workflows, a simple YAML configuration file specifying the system and R package dependencies for rendering each workflow is created for 18 current Bioconductor workflows. The liftr package (https://liftr.me) is used to build the Docker images and render the workflows within the images. Less than 100 lines of R code are used in total. Source code is freely available from https://github.com/road2stat/dockflow.

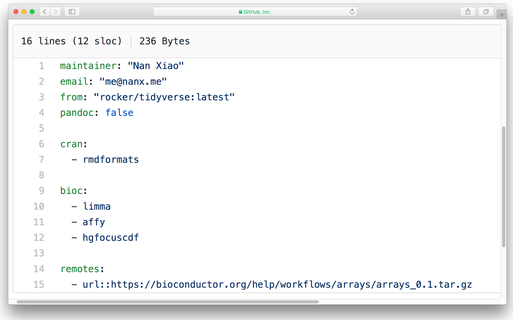


DELIVERABLES

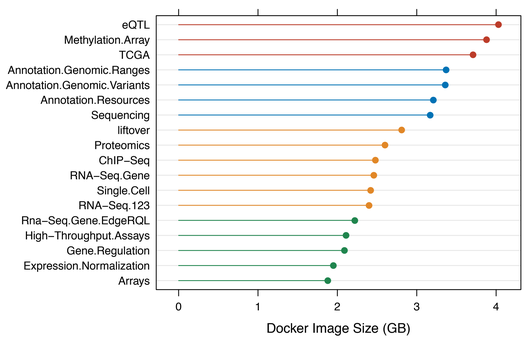
DockFlow.org - Rendered Workflows



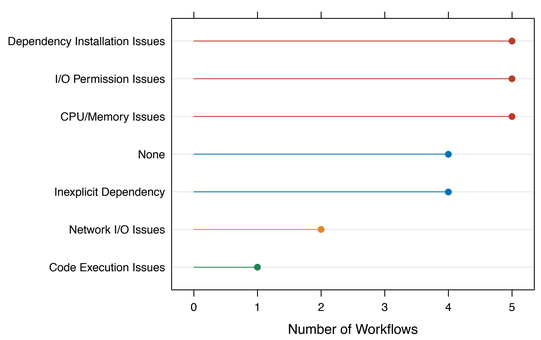
YAML Configuration for Every Workflow



Distribution of Docker Image Size



Distribution of Issue Category



ANALYSIS

Best Practices for Authoring R Markdown Workflows

Declare every necessary dependency

Some of the system or R package dependencies are not obvious but essential for rendering the workflow. Try to declare them explicitly in the R Markdown to save time for containerization.

Avoid install R packages on-the-fly

Trying to install packages in the R code chunks may cause issues because R does not have write permissions in the Docker image. Try declaring all dependencies in the R Markdown file.

Prepare code parallelization

To render the workflows efficiently, some workflows may use the parallelization features offered by various R packages. When setting the number of CPU cores, authors should try to determine the number of cores on-the-fly instead of using a fixed number.

Download huge files from the cloud

It is a reasonable need for the workflow to download some large data files in the R code chunks, although sometimes this may fail due to unpredictable connection issues. We recommended to incorporate some automatic retry mechanisms to be robust.

Best Practices for Containerizing R Markdown Workflows

Solve I/O Permission Issues

Some of the annotation packages may try to create file-based caches. Such operations may encounter errors since R does not have write permissions in the Docker image. We can try setting the cache location to be in the same directory of the workflow.

Minimize Docker engine resource constraints

Some of the bioinformatics workflows may require some more computational resources (CPU or memory) to build the image or render. Try to adjust such parameters for Docker (esp. if you use Docker for Mac) to avoid possible OOM kill issues.

Create minimal viable dependencies

Only keep the required dependencies in the YAML configuration file for liftr, because each additional dependency will increase the size of the Docker image to be built. Consequently, this will increase the future network load for pulling such images.

Debug with patience and love

Containerize large workflows may not be easy at first. Solve the possible issues with patience, and try to enjoy the process!