Docker

Docker is a utility for managing virtual machines (VMs); although it can be used for many purposes, a particularly useful scientific application of Docker is to enable reproducible science. A common problem in scientific communication is that one researcher writes a set of scripts that implement some new analysis or algorithm. This researcher then uses the analysis to demonstrate a novel finding and publishes this finding. However, because the scripts were hacked together and require specific libraries and possibly specific files and directories in specific places, it is nearly impossible for anyone else to use the algorithm without rewriting it from scratch, and the original researcher is hesitant to publish something that will almost certainly break on any other computer but hers.

Docker doesn’t solve the problem of the original researcher’s scripts being a mess, but it does allow the researcher to package all of those necessary files and directories into a single disk image that is guaranteed to run on any Docker-enabled machine. To do this, the researcher constructs a Docker image containing all of the data and code that is necessary and uploads the image to Docker-hub, where any scientist can download and use it.

Useful Links

- Install Docker on a Mac
- A Quick Introduction to Docker
- A Longer Video-Introduction to Docker
- Getting Started with Docker
- Introduction to the Dockerfile Syntax

Example

One example of a scientific Docker image is the docker that applies retinotopic templates to a FreeSurfer subject's cortex. This docker, in Docker-hub, is named 'nben/apply_retinotopy_templates' (nben is the user and apply_retinotopy_templates is the docker name). Because the docker is in Docker-hub, you don’t need to download it to run it (docker will do that for you automatically). And because the docker contains a complete install of FreeSurfer, Python, and other tools it needs, you only need to install docker and docker-machine to use it.

To run this docker, you’ll first need to install docker and docker-machine (see above install links). Once that is done, you’ll need to start your docker-machine (see the above getting started guide). Once you have a docker-machine running, you can run a docker with the docker command:

```
docker run -it --rm -v /Volumes/server/Freesurfer_subjects/wl_subj004:/input nben/apply_retinotopy_templates:latest
```

The above command tells docker to obtain the latest version of the apply_retinotopy_templates by user nben. This will result in the docker image being downloaded then run. The -it option is actually a pair of flags: -t and -i, both of which deal with interactivity. The --rm option specifies that the docker-machine should garbage collect the docker image when it is finished running; the -v specifies that what follows is a directory in your local machine followed by a : followed by a directory inside the docker that your machine's directory should be mounted. In the above command, we mounted the freesurfer directory for a subject in the docker's /input directory.

Making a Docker

Making a Docker image is fairly straightforward. All the instructions for a docker go in a file usually named Dockerfile. The Dockerfile contains a list of commands that setup the docker; i.e., they are run during docker creation but not during the docker execution. Generally speaking, a Dockerfile will contain commands like COPY (copy something on your machine into the docker), RUN (run a command, like apt-get, in the docker), and ENTRYPOINT (tell the docker what to do when someone runs it). As an example, the directory /Projects/Retinotopy/ Templates/Docker on Acadia contains the Dockerfile and all necessary data for the retinotopy template docker described above. This section will walk through the commands in the Dockerfile found there and explain them.

At the beginning of the Dockerfile are several comments, all of which begin with a pound sign (#). These can be ignored. After the comments come several lines that begin with a capital word. Each of these commands is executed, in order, inside the docker image, while building the docker image. The docker image basically starts out identical to the FROM docker image. Here we will examine the commands in some detail:

1. FROM vistalab/freesurfer-core
   The FROM command must always be the first command in a Dockerfile. It specifies that this docker is based on the "vistalab/freesurfer-core" docker, which comes with FreeSurfer's core libraries installed already. Essentially, this tells docker to download the vistalab/freesurfer-core and make a copy of it as the start-point for our new docker.
2. MAINTAINER Noah C. Benson <nben@nyu.edu>
   This line just annotates the name and email address of this Docker's maintainer.
3. COPY /fsaverage_sym /opt/freesurfer/subjects/fsaverage_sym/
   This line (and the next several COPY lines) instruct the docker to copy the directory /fsaverage_sym (locally) to the path /opt/freesurfer/subjects/fsaverage_sym (inside the docker image). COPY is always recursive for directories like this, and will construct directories inside the docker when necessary. For example "COPY x /a/b/c" will create the directories /a and /a/b if they do not already exist.
4. <several more COPY commands>
5. RUN chmod 755 /opt/share/retinotopy/template/apply_template.sh
   RUN chmod 755 /opt/share/retinotopy/template/run_apply_template.sh
These commands tell the docker image to run the chmod commands during the build stage. The command, `chmod 755 <file>`, just makes the `apply_template.sh` and `run_apply_template.sh` scripts executable.

6. **RUN apt-get update**
   
   **RUN apt-get -y install python2.7 python-numpy python-scipy python-setuptools**
   
   These two commands are part of the Ubuntu core and are used to install packages (like python and git, in this case). The `apt-get` program is a sophisticated package manager that, as a general rule, you can trust to correctly resolve dependencies and install libraries or tools. The `apt-get update` command tells `apt` to update its package list (in case new versions of the packages are available) and the `apt-get -y install <packages>` command tells `apt` to resolve dependencies and install the given list of packages. The `-y` flag simply tells it that it should assume a "yes" answer to any questions it has (which is generally fine in this kind of case). Once these commands have run, then python and git, along with the listed python libraries, should be installed inside the docker.

7. **RUN cd /opt && git clone https://github.com/noahbenson/neuropythy**
   
   This command switches to the `/opt` directory then git clones the `neuropythy` package (which is used to fill the volume ribbon of a subject from the surface data).

8. **RUN cd /opt/neuropythy && python setup.py install**
   
   This command switches to the `/opt/neuropythy` directory, which contains the `setup.py` instructions for the `neuropythy` library, and uses these instructions to install the library and its dependencies.

9. **RUN ln -s /opt/neuropythy/surf2ribbon /usr/bin/surf2ribbon**
   
   This command makes a link from the `neuropythy` repo's `surf2ribbon` script to the `/usr/bin` directory so that it can be run more easily.

10. **ENV SUBJECTS_DIR /opt/freesurfer/subjects**
    
    This command specifies that the `SUBJECTS_DIR` environment variable is always set to `/opt/freesurfer/subjects` when using this docker image.

11. **ENTRYPOINT ["/opt/share/retinotopy-template/run_apply_template.sh"]**
    
    This final command specifies the entry-point of the docker when it is run (with the docker run ... command). Basically, the brackets should contain strings that specify, first, the script to be run and, afterwards, the options to pass to the script. Other options on the docker command line can also be passed along. When the docker is run, this script will be executed, with no arguments in this case.