CS1810 Environment Setup

Welcome to CS1810! In this handout, we will be setting up a container (to be explained soon) for you to use in this class. This includes all of the projects on beautiful algorithms and exploreinformatics. In exploreinformatics, we are going to explore how the alignment algorithms and HMMs from class are applied in a real-world setting. To get us started, we need to install a bunch of packages. Unfortunately, a lot of the bioinformatics packages have not adapted to the new M1/M2 Apple chips. So, instead of creating complex conda environments, we will be using Docker!

This container will contain all packages (Python and R) that you might need in CS 1810. We encourage you to use the container to run your code!

Note: If you run into issues with your Docker at any point during the course, please reference the Docker Debugging Guide at the bottom of this page.

Environment Setup

Your computer is probably running Mac OS X, Windows, or Linux. These different operating systems all come with different libraries and preinstalled software. In this class, we will use a container, a technology that will allow every student to access the same environment setup and software packages. The container runs a Linux-based operating system, Ubuntu.

Docker

Docker is one of the most popular container solutions and widely used in industry. In CS 1810, we use Docker because it lets you run a course container on Windows, macOS, or Linux. This allows us to circumvent any issues that may arise from installing packages individually. These Docker images and containers are incredibly popular in bioinformatics due to their portability and accessibility. While we understand this is not a trivial setup process, we believe that this is an important and unique skill to gain.

Task: Download and install Docker.

You may download Docker here. On Linux machines, follow the instructions here.
After downloading Docker, follow Docker's instructions to install it on your OS. Use the recommended settings and accept if Docker asks for privileged access. Finally, if prompted, you do not need a Docker Desktop account to complete this setup process.
On Windows or macOS, open the Docker Desktop application after it has been installed. You may see a message similar to "Docker Desktop is Starting...". Once this message goes away, your Docker has started successfully!
Verify Docker is installed by executing the following command in a terminal:

```
$ docker --version
```

A Docker version number should be printed.
After installing Docker, a Docker process (the Docker daemon) will run in the background. Run the following command to verify:

```
$ docker info
```

This should print some information about your Docker installation.
If you see the following error:

```
ERROR: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is
    the docker daemon running?
```

it means Docker hasn't started running yet. On Windows or macOS, ensure your Docker Desktop is running. On Linux, try the command `sudo systemctl docker restart` in a terminal.
If you're running into other problems while setting up Docker, check out the Docker debugging guide at the bottom of this document!

Windows-based computers only

To run the following steps in this lab, you will need to set up Windows Subsystem for Linux (WSL). WSL should already be enabled after you install Docker, but you may still need to install a Linux distribution. This will run in an actual Linux VM, and you will run your Docker container within that VM (turtles all the way down for you!).

1. **Verify that Hyper-V/Virtual Machine Platform is enabled**
   Hyper-V/Virtual Machine Platform are Windows's hardware virtualization products, which may be required to build containers, use WSL, and install Docker. To verify that these are enabled, go to Settings > Apps > Optional Features > More Windows Features, and verify that the boxes for "Hyper-V" and "Virtual Machine Platform" are checked. (If they are not present, do not worry about them). From here, follow the system prompts. Proceed to step 2.
2. **Do I have a Linux distribution (Linux distro) installed?**
   Run
   ```
   wsl -l -v
   ```
   in the Command Prompt or Powershell. If there is only "Docker Desktop" and "Docker Desktop Data", you do not have a Linux distribution installed. Proceed to step 3.
   Otherwise, you have a Linux distro installed. Proceed to step 4.
3. **Install a Linux Distribution.**
   Run
   ```
   wsl --set-default-version 2
   ```
   to ensure Ubuntu will be installed under WSL 2.
   Install "Ubuntu 20.04" from Microsoft Store. (link here)
   Click "Open" after Ubuntu is downloaded. A terminal will open and guide you through the installation process.
4. **Ensure your Linux Distribution runs on WSL 2.**
   From the output of `wsl -l -v`, find out if your LinWux distro is using WSL 1 or WSL 2 by checking the VERSION column. If it's WSL1:
   Run
   ```
   wsl --set-version <distro name> 2
   ```
   to update your distro to use WSL 2.
5. **Set your default Linux distro**
   Run
   ```
   wsl --setdefault <distro-name>
   ```
   to configure your default Linux distro. `<distro-name>` should be "Ubuntu-20.04" if you installed using step 2.

Enter `wsl` in your Command Prompt or Powershell, and you'll enter into your WSL! For the rest of the Lab, run commands within your WSL Linux environment, unless otherwise specified.
You will also need to connect Docker with WSL. To do so, open your Docker Desktop's settings (on its top right corner), click "Resources", "WSL integration", then enable integration with your Linux distro. Then, click "Apply and Restart".

Set up CS 181 Docker Environment

In Docker, an environment is defined as a Docker image. An image specifies the operating system environment that a container provides. An image can also contain additional software dependencies and configurations. In our case, this includes a series of R and Python packages that you will use during this class. These installation steps are specified in a file, the so-called Dockerfile.

Next, you will download the the course's setup code and create the CS 181 Docker image!

macOS only: install Apple development tools and change Synchronization

If you're running on macOS, you will need to install a set of Apple-recommended command-line tools via the following command:

```
xcode-select --install
```

This ensures that your computer has installed git, a program we'll use below. Alternatively, you may also download and install git directly, following instructions here.
We have, at times, seen that students encounter synchronization issues when using Docker (edits made on your machine do not show up in the container). We will explain this synchronization or mounting in further detail below. Open the Docker dashboard and go to "Settings" → and switch within "Choose file sharing implementation for your containers" to VirtioFS.

Task: Do the following to set up your development environment.

Enter the directory on your computer where you want to do your coursework. For Windows users, choosing somewhere in the C drive will make the following steps easier. Then, clone the git repository that contains the cs181-projects26 codebase:

```
$ git clone https://github.com/brown-cs181/cs181-projects26.git
```

Run `cd cs181-projects26` to enter the directory you have just created.
Inside this folder, do the following:

```
$ ./build-docker  # builds a docker image you'll use in the semester
```

`./build-docker` builds your docker image, which may take a while. Please run this step with your computer connected to a power source. Docker Desktop is very energy consuming. This process downloads a 4GB Python/R environment we pushed to Docker Hub here and customizes it for you.

Entering the container

Once you created your Docker image, you will want to create a container running this image.

Task

Enter your container and explore it by running a few commands. Then, verify that all of your packages have been properly installed. Finally, exit the container by `exit`, or Ctrl-D. If you are a Windows user, then use the Windows-specific instructions below if you run into any issues. Make sure you're inside directory cs181-projects26 when running the command below.

```
username:~/cs181-projects26$ ./run-docker    # enters your Docker container
cs181-user@9899143429a2:/home$    # you're inside the container!
cs181-user@9899143429a2:/home$ uname
Linux
cs181-user@9899143429a2:/home$ echo "Hello world!"
Hello world!
cs181-user@9899143429a2:/home$ ls -lah
total 4.0K
drwxr-xr-x  4 cs181-user cs181-user  128 Sep  2 20:41 .
drwxr-xr-x  1 root       root       4.0K Sep  4 01:26 ..
drwxr-xr-x 10 cs181-user cs181-user  320 Aug 30 22:45 cs181-user
drwxr-xr-x  5 cs181-user cs181-user  160 Aug 30 14:26 test_scripts
cs181-user@9899143429a2:~$ exit  # or Ctrl-D
```

Don't worry if the number after `cs181-user` is different. This is an identifier that uniquely identifies this container. You may run any Linux commands inside this container. Further, you may install further python packages using `pip install`. To exit, enter `exit` or use Ctrl-D.

Task

Within your container, make sure that all packages have been properly installed.

```
cs181-user@9899143429a2:/home$ # Make sure that you are in the container!!
cs181-user@9899143429a2:/home$ cd test_scripts/; ./test-environment
Package hmmlearn is installed.
Package numpy is installed.
...
Package  ChIPseeker  is installed and can be loaded.
cs181-user@9899143429a2:/home$ samtools --version
samtools 1.12
Using htslib 1.12
Copyright (C) 2021 Genome Research Ltd.
```

The output from the test script might be a little long but check to make sure that there are no lines that say that a package was not properly installed. If a package is not properly installed, please contact the course staff.

Instructions for Windows only

If you see errors along the lines of `bash: '\r': command not found` or `bash: /home/cs300-user/.bash_profile: line 5: syntax error: unexpected end of file` when you enter the container, you need to convert your files' line endings (characters that delineate the end of a line) from Windows to UNIX (Linux) format. To do so, do the following:

1. Enter into WSL. You may do so by opening Ubuntu in the start menu or through the `wsl` command.

2. Use `cd` to navigate to the folder where you just cloned the setup repository. If you cloned your setup repository to your C: drive, use `cd /mnt/c` to enter the C: drive from your WSL.
3. Run

   ```
   sudo apt-get update
   ```

   then install dos2unix with

   ```
   sudo apt-get -y install dos2unix
   ```

4. Run

   ```
   dos2unix ./run-docker
   ```

   (you should only need to run this once).
5. Next, run

   ```
   ./run-docker
   ```

If you see the next line begins with `cs181-user@@9899143429a2:~$ ...`, you're inside the container, hopefully without errors!

Shared Folders

"If my docker container is a separate (virtual) computer than my laptop, how will I move files between the two?", you may ask. Great question!

Inside of the container, your home directory (/home) is actually a mount of the home directory inside your cs181-projects26 directory. Any changes you make in one will be visible in the other.

If you move or rename your cs181-projects26 folder...
Your Docker container will still try to mount to the original path, even after you rename, remove, or move the folder cs181-projects26.

After moving your folder, you'll need to delete the old container and start a new container. You can do so with:

```
./run-docker --clean
```

You should be able to enter a container, and see all of your work now! As long as there is a `home directory`, the container will be able to mount in the new location. Keep in mind that if you installed any packages via `sudo apt` or `pip` etc. that they will have to be reinstalled in the container.

If you have filesystem sync issues on MacOS...
This might be caused by gRPC FUSE, which is sometimes buggy on MacOS. Go into your docker settings and make sure that the gRPC FUSE checkbox is not checked, or that the sync method is set to VirtioFS.

Completing Course Projects and Utilizing the Container

In your cs181-user directory, which is the base home directory in the container, you may complete your course projects. The container is entirely accessed from the terminal so you may run and test your projects within the container. If, for example you wanted to use VS-Code, you could have a terminal open alongside the IDE. Then, you could make edits to your code and because

it is mounted on your `home` directory, you could immediately run your code to perform any respective testing.

We believe any package that you might want to install is already pre-installed into the Docker image. However, if there is something you would like to use at any point in the project you may use `pip install` and `install.package()` for Python and R. Finally, you have been granted sudo access within the image. This allows you to install command-line packages such as vim!

Feel free to post on EdStem if you have any questions on how the setup itself actually works.

Docker Debugging Guide

Note: Please post on EdStem if you are still having issues after trying these steps.

TLDR: Terminating and restarting your container can solve a lot of issues!

Here are a few strategies that you can try on your own to resolve container issues. While these strategies might not resolve every issue, they are good starting points.

Recall from Installation

If you changed the file structure on which the container is mounted (e.g., by renaming the folder you put your CS 1810 cs181-projects23 folder into), you will certainly run into Docker issues. You do not have to rebuild the container altogether (i.e., go through the entirety of the above setup). However, you do have to delete the previous container you were using because Docker will try to mount your container on a folder that no longer exists. That can be done with:

```
$ ./run-docker --clean
```

If you get an error with this, please look below. Also, as a reminder, you will have to re-install any custom packages (such as vim) that you had installed prior to deleting and rebuilding your container.

Is Docker Daemon Running?

Your container will not run if Docker desktop is not running. You might get a message amongst a much larger error message that says:

```
ERROR: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the
    docker daemon running?
```

Make sure that your Docker Desktop is running in the background. You can do this by just opening the app from your finder or a shortcut! You can also run docker info to make sure that your Docker is running.

What is an example of this type of error?
A sample error message can look something like this:

```
$ ./run-docker
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker
    daemon running?
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker
    daemon running?
cs300 network not found, creating one...
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker
    daemon running?
```

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker
    daemon running?
Starting a new container...
docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the
    docker daemon running?.
See 'docker run --help'.
```

What will docker info look like if my Docker is active?
You will get a Docker daemon error if your docker desktop is not running when you run this command. If Docker Desktop is running, you will get an output that starts with this:

```
$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc., v0.10.0)
  compose: Docker Compose (Docker Inc., v2.15.1)
  dev: Docker Dev Environments (Docker Inc., v0.0.5)
  extension: Manages Docker extensions (Docker Inc., v0.2.17)
  sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (
    Anchore Inc., 0.6.0)
  scan: Docker Scan (Docker Inc., v0.23.0)

Server:
 Containers: 1
  Running: 1
  Paused: 0
  Stopped: 0
 Images: 1
 ...
```

Windows: Trouble Entering Your Container

Through the course infrastructure setup process on Windows, you should have integrated WSL with your Docker. If you run into issues in which you are unable to run your course container, enter the following:

```
$ wsl -l -v
```

This should tell you whether or not your WSL and Docker are running. Keep in mind that you need both running in order to enter the course container. You might need to manually terminate the Docker backend if you are running into issues and are looking to reset.

Alternatively, you can use `wsl --shutdown` without having to enter the Task Manager to terminate the Docker WSL Backend.

Error Responses from Daemon

If your terminal in your container is hanging or if you wanted to run the –clean command above, you might run into the following error message:

```
$ ./run-docker --clean
Error response from daemon: Could not kill running container ..., cannot remove -
    tried to kill container but did not receive an exit event.
```

The containers sometimes get stuck. Just closing the terminal does not actually terminate the container! There are a few ways to resolve this issue:

1. Go to your Activity Monitor / Task Manager and look up Docker. Force Quit or Quit the process. This should terminate the container and will allow you to rebuild it.
2. If that does not work, restarting your computer will resolve the stuck container.

MacOS Syncing Issue Using an IDE

In the past, we have seen an issue with MacOS in which students would make edits to files through an IDE but their edits would not show up in the container. Opening the Docker dashboard and going to "Settings" → and switch within "Choose file sharing implementation for your containers" from gRPC Fuse to VirtioFS. Then, restart your container and the syncing issue should resolve!

Note: If you are on an older version of Docker, this setting may appear in "Experimental Settings".

Otherwise, try exiting and remounting your container using `./run-docker` and your changes should show up.

Developed By: pmahable in for Fall 2023
Credits: Thank you to CS0300 at Brown University for code pertaining to building and running the Docker Image.