Setting up a pbdR Docker Container

Prepared by the pbdR Core Team

Contents

1 Installing Docker

Before proceeding, you must first install Docker on your system. This is either painless or completely impossible, depending on what OS version you are running.

1.1 Windows

Windows 10 or later is required. If you are not sure which version of windows you are running, see the Microsoft support article Which Windows operating system am I running.

If you meet the requirements, then you should install the Docker Community Edition for Windows. There are also detailed installation instructions available if something goes wrong.

1.2 Mac

OS X El Capitan 10.11 or later is required. If you are not sure which version of OS X / macOS you are running, see the Apple support article Find the macOS version number on your Mac.

If you meet the requirements, then you should install the Docker Community Edition for Mac. There are also detailed installation instructions available if something goes wrong.

1.3 Linux

Many modern distributions have Docker packages available in the software repository. For example:

- deb (Debian, Ubuntu): `apt-get install docker.io`
- rpm (Fedora, Centos): `yum install docker-io`

If your distribution does not have Docker available in the repo, try searching the Docker Community Editions.

2 Using Docker

2.1 Basics

Generally working with someone else's docker container is pretty simple (building your own can be another matter entirely). Most often you will enter a `docker run` command at the command line and then interact with the container either at the command line, or via a web service or the like.

There is a "hello world" test you can run to make sure that everything is working. Simply enter:

```
docker run hello-world
```

and you should see a quick bit out output explaining that everything worked properly. If you get an error like `Cannot connect to the Docker daemon`, then you need to run the container as an administrator.

For more information, you are encouraged to read the orientation and setup guide from the official Docker documentation.

2.2 Mounting Host Directories

You will probably want to be able to access files from your "host" system (not the container) inside of the container. Fortunately, this is not difficult to do. All you need to do is add a simple flag to your launch command. Namely, you add: `-v HOST_DIRECTORY:CONTAINER_DIRECTORY`, where of course the "host" and "container" directories are actual paths.

So for example, say you want to launch the workshop container described in the next section. The instructions say to run:

```
sudo docker run -i -t rbigdata/pbdr-workshop
```

But say you want to connect the host directory `/tmp/docker/` (something like `c:\tmp\docker` on windows) to `/home/pbdR/` on the container. You could run:

```
sudo docker run -i -t -v /tmp/docker:/home/pbdR rbigdata/pbdr-workshop
```

Then any file you put into your directory on your host system will be available at the home directory. For more details, see the official Docker documentation.

3 pbdR Docker Builds

Docker build configurations for the pbdR package ecosystem. Builds are hosted on Docker Hub. You can find the Dockerfiles at our github repo.

3.1 Workshop/Tutorial

If you are attending a pbdR workshop/tutorial, then you will want to install the pbdr-workshop build. Simply run:

```
sudo docker run -i -t rbigdata/pbdr-workshop
```

You can interact with the running container in the same way as outlined in the "Compute" subsection below. To learn how to access files from your host machine, see the previous section.

3.2 Compute

```
sudo docker run -i -t rbigdata/pbdr-compute
```

Create your pbdR script as usual and run it via mpirun. Take this simple "hello world" for example:

```r
suppressPackageStartupMessages(library(pbdMPI))

comm.print(
  paste("Hello from rank", comm.rank(), "of", comm.size()),
  all.rank=TRUE
)

finalize()
```

Then run the script via:

```
mpirun -np 2 r myscript.r
```

3.3 Client/Server

```
sudo docker run -i -t -p 55555:55555 rbigdata/pbdr-cs
```

To connect to the server, you will need to install the `remoter` package:

```
install.packages("remoter")
```

Then from your local R or RStudio session, you can connect to the server by running:

```
remoter::client()
```