R, Databases and Docker

M. Edward (Ed) Borasky, editor

2018-09-07

Contents

Chapter 1

Introduction

1.1 Using R to query a DBMS in your organization

- Large data stores in organizations are stored in databases that have specific access constraints and structural characteristics. Data documentation may be incomplete, often emphsizes operational issues rather than analytical ones, and often needs to be confirmed on the fly. Data volumes and query performance are important design constraints.
- R users frequently need to make sense of complex data structures and coding schemes to address incompletely formed questions so that exploratory data analysis has to be fast. Exploratory techniques for the purpose should not be reinvented (and so would benefit from more public instruction or discussion).
- Learning to navigate the interfaces (passwords, packages, etc.) between R and a database is difficult to simulate outside corporate walls. Resources for interface problem diagnosis behind corporate walls may or may not address all the issues that R users face.

1.2 Docker's role

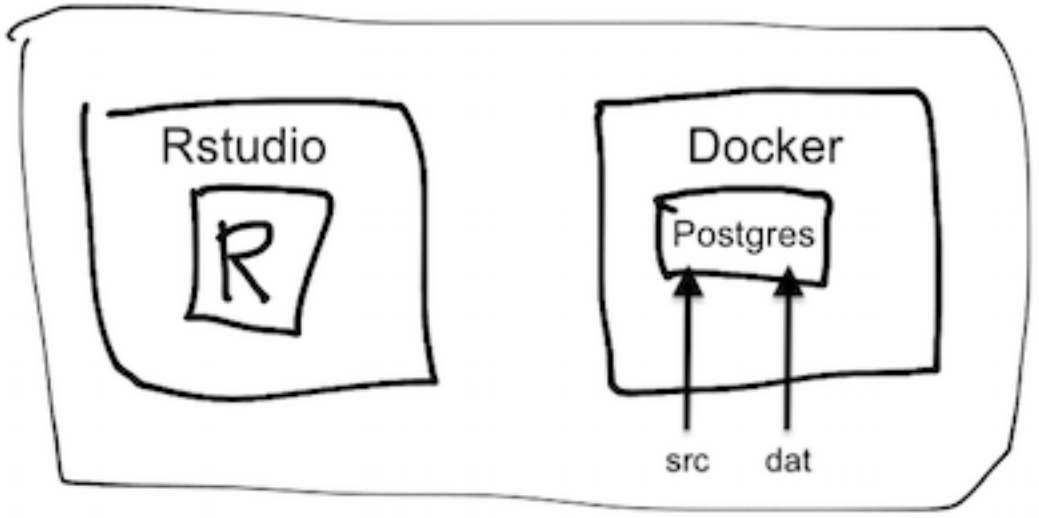
Noam Ross's "Docker for the UseR" suggests that there are four distinct use-cases for useRs. This book explores #2:

1. Make a fixed working environment for reproducible analysis
2. Access a service outside of R **(e.g., Postgres)**
3. Create an R based service (e.g., with `plumber`)
4. Send our compute job somewhere else

- Docker is a relatively easy way to simulate the relationship between an R/Rstudio session and a database – all on on a single machine.
- You may want to run PostgresSQL on a Docker container, avoiding any OS or system dependencies that might come up.

1.3 Docker and R on your machine

Here is how R and Docker fit on your operating system in this tutorial:



(This diagram needs to be updated as our directory structure evolves.)

1.4 Who are we?

- M. Edward (Ed) Borasky - @znmeb
- John David Smith - @smithjd
- Scott Came - @scottcame
- Ian Franz - @ianfrantz
- Sophie Yang - @SophieMYang
- Jim Tyhurst - @jimtyhurst
- Paul Refalo - @paulrefalo

1.5 Prerequisites

You will need

- A computer running Windows, MacOS, or Linux (Any Linux distro that will run Docker Community Edition, R and RStudio will work),
- R, and Rstudio and
- Docker hosting.

The database we use is PostgreSQL 10, but you do not need to install that - it's installed via a Docker image. RStudio 1.2 is highly recommended but not required.

1.6 Install Docker

Install Docker. Note that this can be tricky.

- On a Mac
- On UNIX flavors
- For Windows, consider these issues and follow these instructions.

1.7 Download the repo

First step: download this repo. It contains source code to build a Docker container that has the dvdrental database in Postgress and shows how to interact with the database from R.

Chapter 2

Docker Hosting for Windows

2.1 Hardware requirements

You will need an Intel or AMD processor with 64-bit hardware and the hardware virtualization feature. Most machines you buy today will have that, but older ones may not. You will need to go into the BIOS / firmware and enable the virtualization feature. You will need at least 4 gigabytes of RAM!

2.2 Software requirements

You will need Windows 7 64-bit or later. If you can afford it, I highly recommend upgrading to Windows 10 Pro.

2.2.1 Windows 7, 8, 8.1 and Windows 10 Home (64 bit)

Install Docker Toolbox. The instructions are here: https://docs.docker.com/toolbox/toolbox_install_windows/. Make sure you try the test cases and they work!

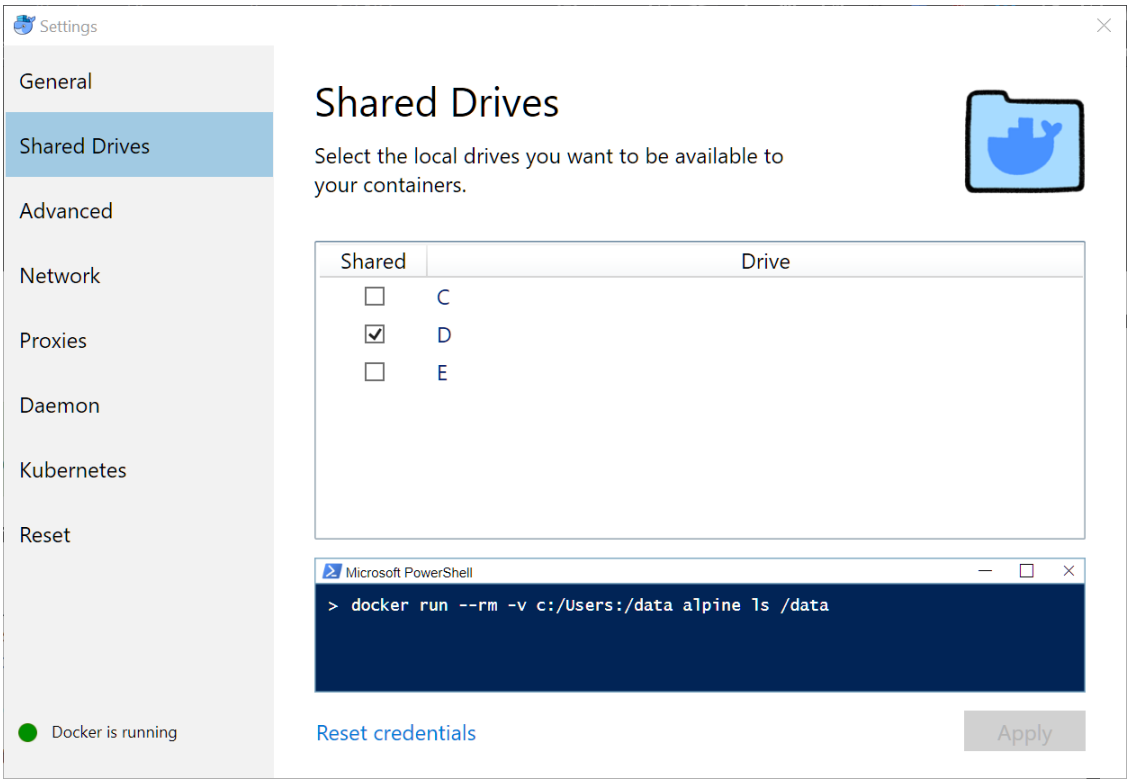
2.2.2 Windows 10 Pro

Install Docker for Windows *stable*. The instructions are here: https://docs.docker.com/docker-for-windows/install/#start-docker-for-windows. Again, make sure you try the test cases and they work.

2.3 Docker for Windows settings

2.3.1 Shared drives

If you're going to mount host files into container filesystems, you need to set up shared drives. Open the Docker settings dialog and select `Shared Drives`. Check the drives you want to share. In this screenshot, the `D:` drive is my 1 terabyte hard drive.

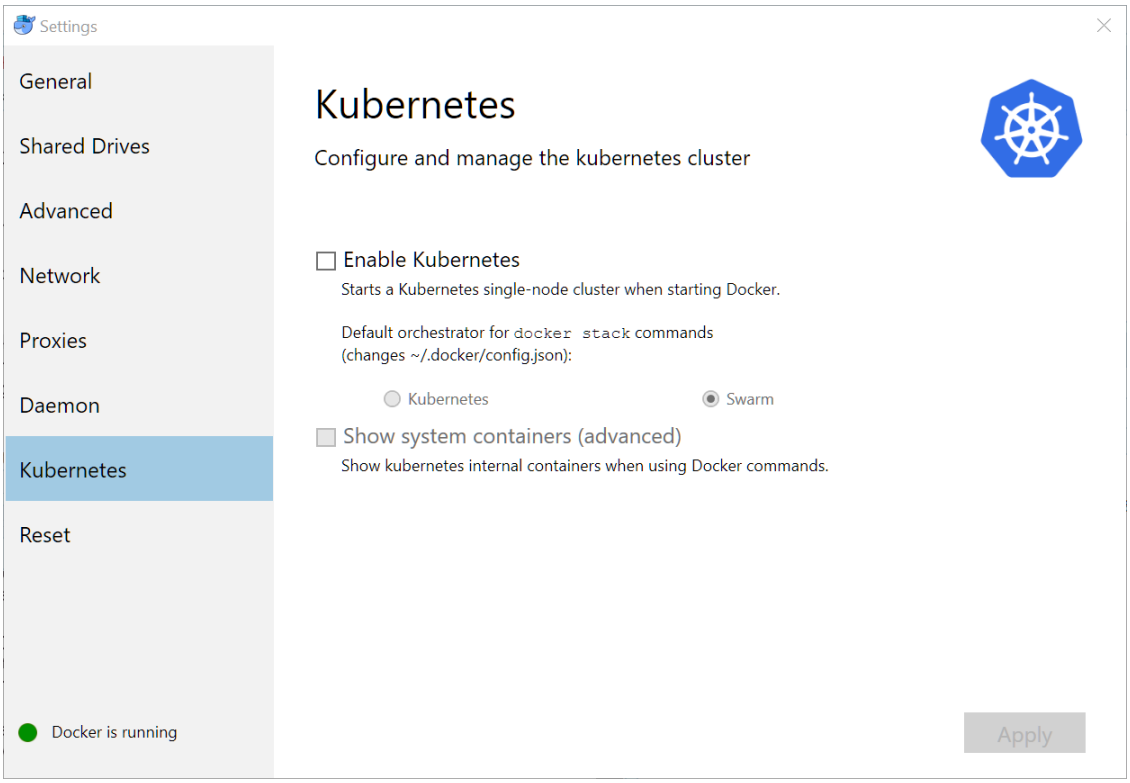


2.3.2 Kubernetes

Kubernetes is a container orchestration / cloud management package that's a major DevOps tool. It's heavily supported by Red Hat and Google, and as a result is becoming a required skill for DevOps.

However, it's overkill for this project at the moment, and it doesn't seem to be compatible with the Docker Compose we're using. So you should make sure it's not enabled.

Go to the `Kubernetes` dialog and make sure the `Enable Kubernetes` checkbox is cleared.



2.4 Git, GitHub and line endings

Git was originally developed for Linux - in fact, it was created by Linus Torvalds to manage hundreds of different versions of the Linux kernel on different machines all around the world. As usage has grown, it's achieved a huge following and is the version control system used by most large open source projects.

If you're on Windows, there are some things about Git and GitHub you need to watch. First of all, there are quite a few tools for running Git on Windows, but the RStudio default and recommended one is Git for Windows (https://git-scm.com/download/win).

By default, text files on Linux end with a single linefeed (`\n`) character. But on Windows, text files end with a carriage return and a line feed (`\r\n`). See https://en.wikipedia.org/wiki/Newline for the gory details.

Git defaults to checking files out in the native mode. So if you're on Linux, a text file will show up with the Linux convention, and if you're on Windows, it will show up with the Windows convention.

Most of the time this doesn't cause any problems. But Docker containers usually run Linux, and if you have files from a repository on Windows that you've sent to the container, the container may malfunction or give weird results.

In particular, executable `sh` or `bash` scripts will fail in a Docker container if they have Windows line endings. You may see an error message with `\r` in it, which means the shell saw the carriage return (`\r`) and gave up. But often you'll see no hint at all what the problem was.

So you need a way to tell Git that some files need to be checked out with Linux line endings. See https://help.github.com/articles/dealing-with-line-endings/ for the details. Summary:

1. You'll need a `.gitattributes` file in the root of the repository.
2. In that file, all text files (scripts, program source, data, etc.) that are destined for a Docker container will need to have the designator `<spec> text eol=lf`, where `<spec>` is the file name specifier, for example, `*.sh`.

Chapter 3

Learning Goals and Use Cases

3.1 Context: Why integrate R with databases using Docker?

- Large data stores in organizations are stored in databases that have specific access constraints and structural characteristics.
- Learning to navigate the gap between R and the database is difficult to simulate outside corporate walls.
- R users frequently need to make sense of complex data structures using diagnostic techniques that shuld not be reinvented (and so would benefit from more public instruction and commentary).
- Docker is a relatively easy way to simulate the relationship between an R/Rstudio session and database – all on on a single machine.

3.2 Learning Goals

After working through this tutorial, you can expect to be able to:

- Run queries against Postgres in an environment that simulates what you will find in a corporate setting.
- Understand some of the tradeoffs between queries aimed at exploration or informal investigation using dplyr and those where performance is important because of the size of the database or the frequency with which a query is run. You will be able to rewrite dplyr queries as SQL and submit them directly. You will have some understanding of techniques for assessing query structure and performance.
- Set up a Postgres database in a Docker enviornment and understand enough about Docker to swap databases, swap DBMS' (e.g., MySQL for Postgres, etc.)

3.3 Use cases

Imagine that you have one of several roles at **DVDs R Us** and that you need to:

- As a data scientist, I want to know the distribution of number of rentals per month per customer, so that the Marketing department can create incentives for customers in 3 segments: Frequent Renters, Average Renters, Infrequent Renters.
- As the Director of Sales, I want to see the total number of rentals per month for the past 6 months and I want to know how fast our customer base is growing/shrinking per month for the past 6 months.
- As the Director of Marketing, I want to know which categories of DVDs are the least popular, so that I can create a campaign to draw attention to rarely used inventory.
- As a shipping clerk, I want to add rental information when I fulfill a shipment order.

Figure 3.1: Entity Relationship diagram for the dvdrental database

- As the Director of Analytics, you want to test as much of the production R code in my shop against a new release of the DBMS that the IT department is implementing next month.
- etc.

3.4 Environment

This tutorial uses the Postgres version of "dvd rental" database, which can be downloaded here. Here's a glimpse of it's structure:

Chapter 4

Docker, Postgres, and dvdrental setup

(the links may not be working correctly yet…)

4.1 Docker setup

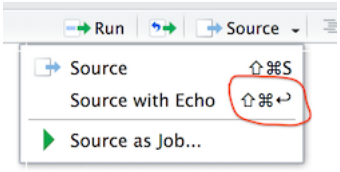
Verify that Docker is up and running.

4.2 Bring up Postres in Docker

There's a lot to learn about Docker and many uses for it, here we just cut to the chase.

- Use ./src/1_test_postgres-b.R to demonstrate that you have a persistent database by uploading `mtcars` to Postgres, then stopping the Docker container, restarting it, and finally determining that `mtcars` is still there. (Note that if you are running Postgres locally, you'll have to close it down to avoid a port conflict.) See the results here: ./src/1_test_postgres-b.md

Note: when running the scripts in this repo, there's a difference between "sourcing" a file and "source with echo". Use "source with echo":



4.3 DVD Rental database installation

- Download the backup file for the dvdrental test database and convert it to a .tar file with:

  ./src/2_get_dvdrental-zipfile.Rmd. See the results here: ./src/2_get_dvdrental-zipfile.md

- Create the dvdrental database in Postgres and restore the data in the .tar file with:

  ./src/3_install_dvdrental-in-postgres-b.Rmd. See the results here: ./src/3_install_dvdrental-in-postgres-b.md

4.4 Verify that the dvdrental database is running and browse some tables

- Explore the dvdrental database:

  ./src/4_test_dvdrental-database-b.Rmd See the results here: ./src/4_test_dvdrental-database-b.md

Need to incorporate more of the ideas that Aaron Makubuya demonstrated at the Cascadia R Conf.

Chapter 5

Interacting with Postgres from R

5.1 Topics to cover

- keeping passwords secure
- differences between production and data warehouse environments
- overview investigation: do you understand your data
  - documentation and its limits
  - find out how the data is used by those who enter it and others who've used it before
  - what's *missing* from the database: (columns, records, cells)
  - why is there missing data?
- dplyr queries
- examining dplyr queries (show_query on the R side v EXPLAIN on the Postges side)
- performance considerations: get it to work, then optimize
- Tradeoffs between leaving the data in Postgres vs what's kept in R:
  - browsing the data
  - larger samples and complete tables
  - using what you know to write efficient queries that do most of the work on the server
- learning to keep your DBAs happy

5.2 More topics

- from Aaron Makubuya's workshop at the Cascadia R Conf.
  - SELECT * vs SELECT list of columns
  - controlling the number of rows returned with WHERE
  - Glue for constructing SQL statements vs dplyr
  - JOIN flavors
  - parameterizing SQL queries
  - show_query and EXPLAIN

Chapter 6

Other resources

6.1 Editing this book

- Here are instructions for editing this tutorial

6.2 Docker alternatives

- Chosing between Docker and Vagrant

6.3 Docker and R

- Noam Ross' talk on Docker for the UseR and his Slides give a lot of context and tips.
- Good Docker tutorials
  - An introductory Docker tutorial
  - A Docker curriculum
- Scott Came's materials about Docker and R on his website and at the 2018 UseR Conference focus on **R inside Docker**.
- It's worth studying the ROpensci Docker tutorial

6.4 Documentation Docker and Postgres

- The Postgres image documentation
- Dockerize PostgreSQL
- Postgres & Docker documentation
- Usage examples of Postgres with Docker

6.5 More Resources

- David Severski describes some key elements of connecting to databases with R for MacOS users
- This tutorial picks up ideas and tips from Ed Borasky's Data Science pet containers, which creates a framework based on that Hack Oregon example and explains why this repo is named pet-sql.