R, Databases and Docker

M. Edward (Ed) Borasky, editor

2018-09-04

Contents

1 Prerequisites **5**

2 Docker Hosting for Windows **7**
2.1 Hardware requirements 7
2.2 Software requirements 7
2.3 Docker for Windows settings 7
2.4 Git, GitHub and line endings 9
2.5 Installing the R `docker` package 10

3 Miniconda Integration **11**
3.1 Why do this? 11
3.2 Install the `installr` package. 11
3.3 Install `Miniconda3` 11
3.4 Install `reticulate` 16

Chapter 1

Prerequisites

- R, 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.

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!

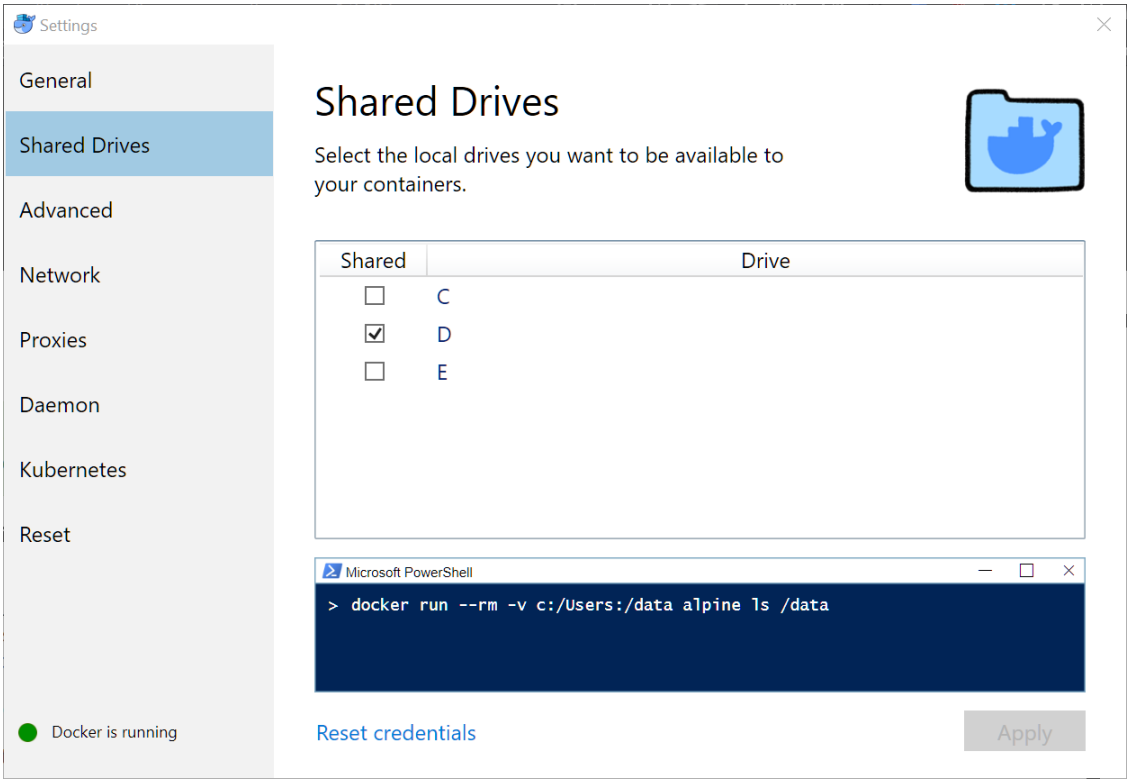
2.2.2 Windows 10 Pro

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.

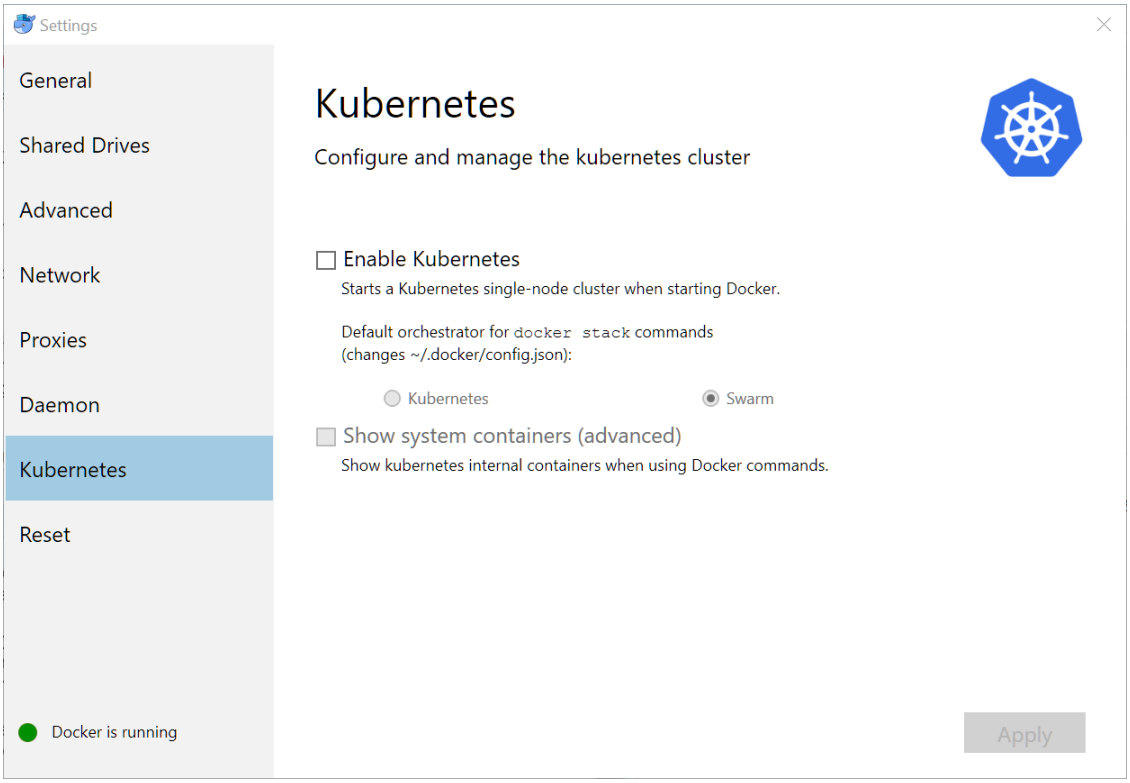


2.3.2 Kubernetes

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`.

2.5 Installing the R `docker` package

First, if you haven't already, install Miniconda3 and `reticulate`. The instructions are at https://github.com/smithjd/sql-pet/blob/master/Howtos/miniconda_integration.md.

Now, install `docker`:

```
if (!require(docker)) install.packages("docker")
library(docker)
```

Create a `conda` virtual environment:

```
library(reticulate)
conda_remove(envname = "docker")
conda_create(envname = "docker")
conda_install(envname = "docker", packages = "docker", pip = TRUE)
use_condaenv("docker")
```

Did it work?

```
library(docker)
client <- docker$from_env()
s <- client$containers$run("alpine", 'echo -n "Hello World!"', remove=TRUE)
print(s$decode("UTF-8"))
```

Chapter 3

Miniconda Integration

3.1 Why do this?

A number of R deep learning packages use Python under the hood. RStudio's **keras** (Allaire and Chollet, 2018) package, for example, works this way. Also, the R **docker** (Karambelkar, 2017) package works by calling a Python Docker API library from R via **reticulate** (Allaire et al., 2018). And, of course, you'll probably end up receiving a Jupyter notebook or two even if you're a die-hard RStudio user.

`Miniconda` is a bare-bones minimalist version of the rather large Anaconda environment. If you're doing Python data science, you probably have the full Anaconda installed already. But for R programmers, we only want enough Python for the R packages that use Python libraries to work. So … here we go!

3.2 Install the `installr` package.

There's an R package called **installr** (Galili, 2018) that can run a Windows installer.

```
if (!require(installr)) install.packages("installr")
library(installr)
```

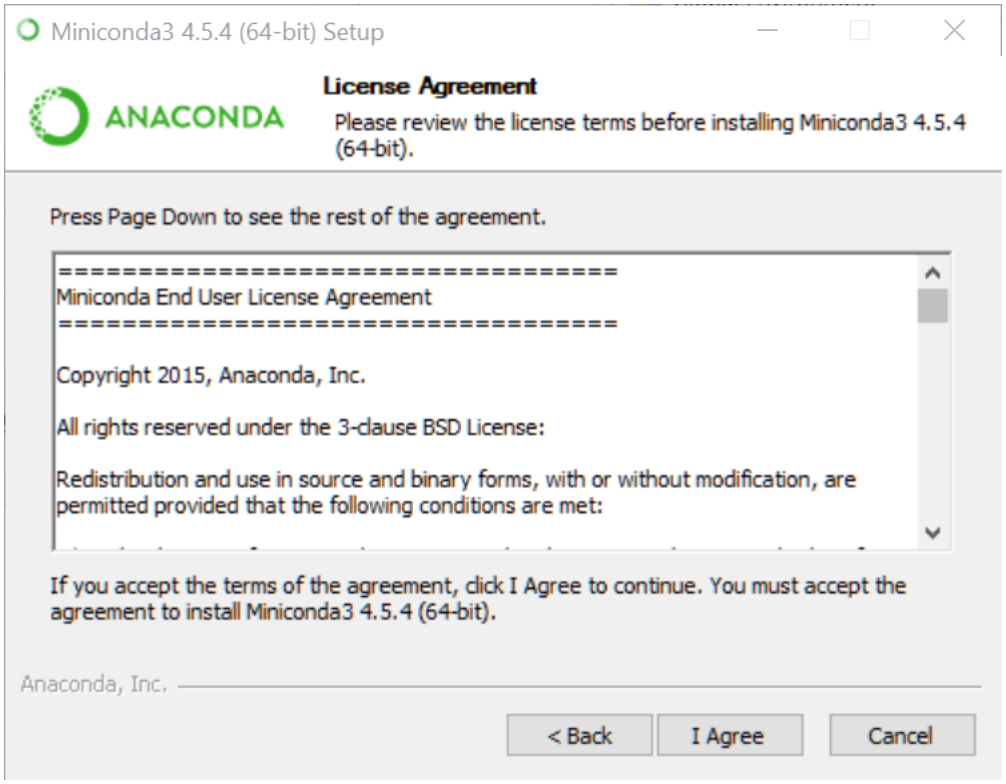
3.3 Install `Miniconda3`

The following R code chunk will install 'Miniconda3.

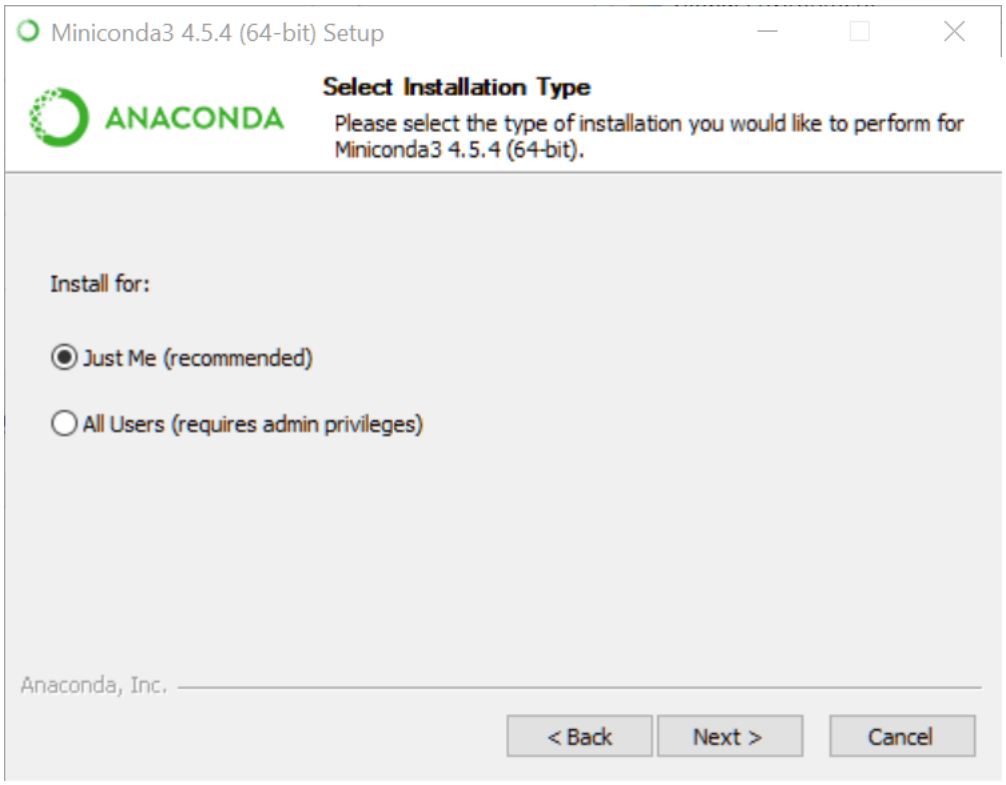
```
install.URL("https://repo.continuum.io/miniconda/Miniconda3-latest-Windows-x86_64.exe")
```

Here are the screenshots you'll see:

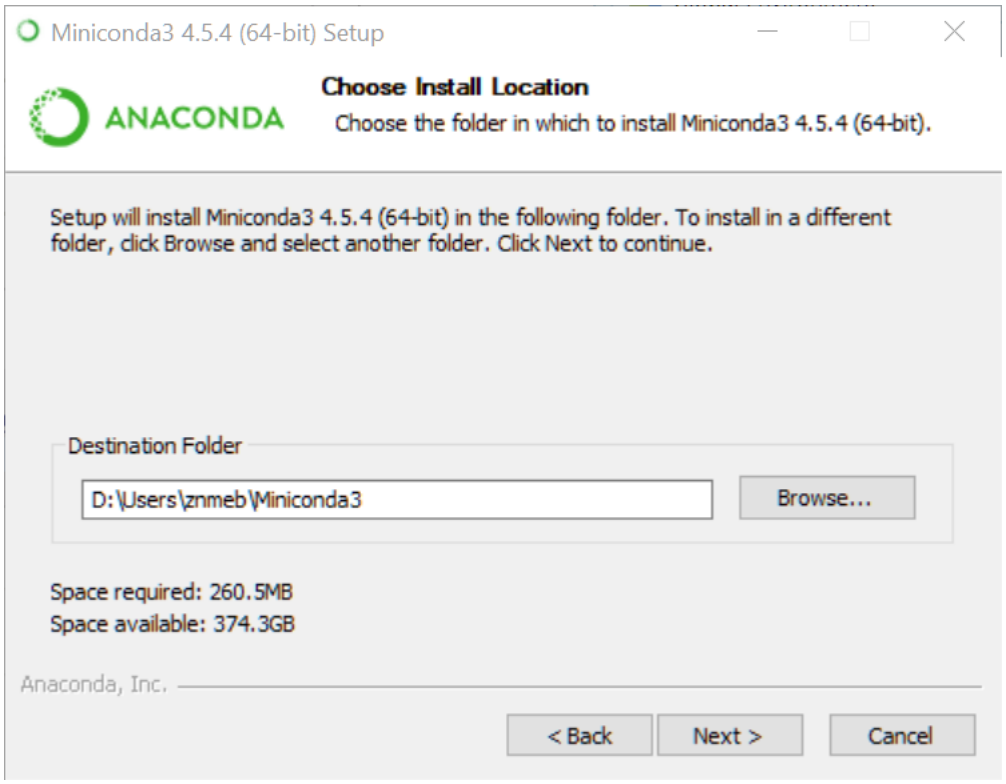
Click `Next`.



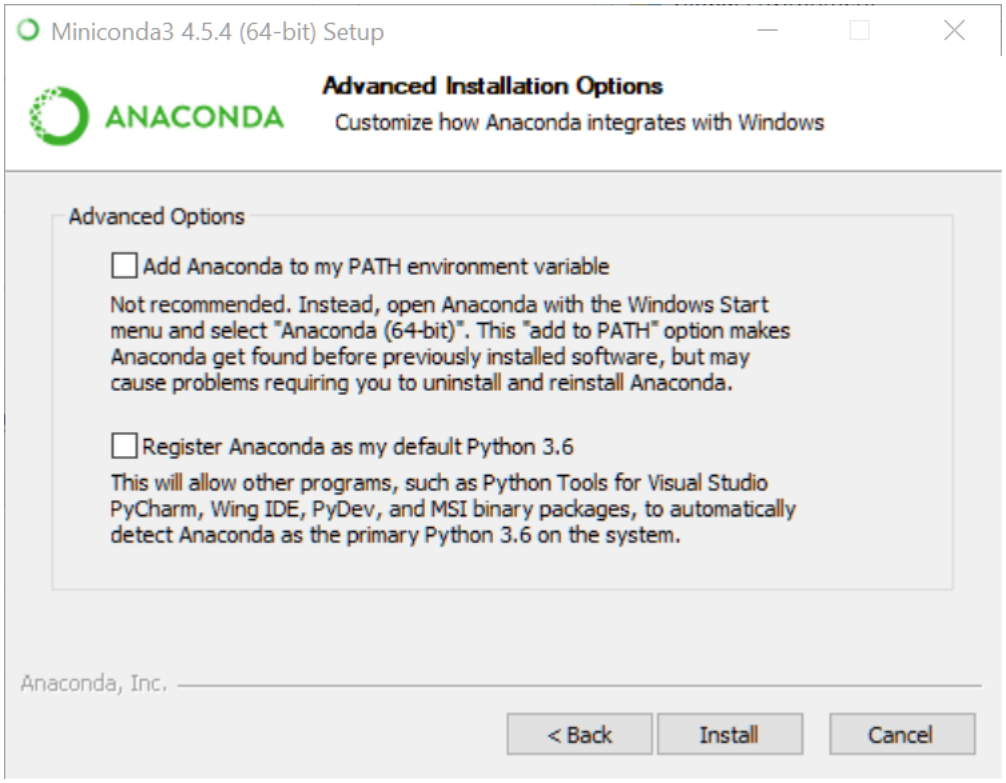
Click `I Agree`.


Miniconda3 4.5.4 (64-bit) Setup
ANACONDA
Select Installation Type
Please select the type of installation you would like to perform for Miniconda3 4.5.4 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel

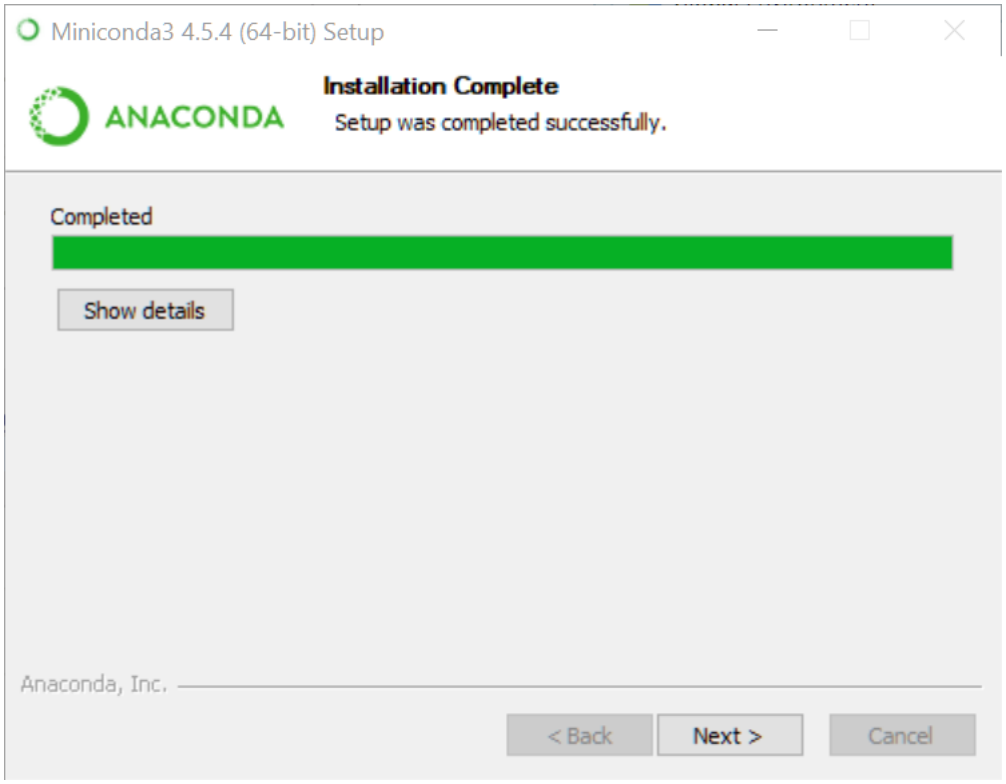
`Just Me`, `Next`.


Miniconda3 4.5.4 (64-bit) Setup
ANACONDA
Choose Install Location
Choose the folder in which to install Miniconda3 4.5.4 (64-bit).
Setup will install Miniconda3 4.5.4 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
D:\Users\znmeb\Miniconda3
Browse...
Space required: 260.5MB
Space available: 374.3GB
Anaconda, Inc.
< Back
Next >
Cancel

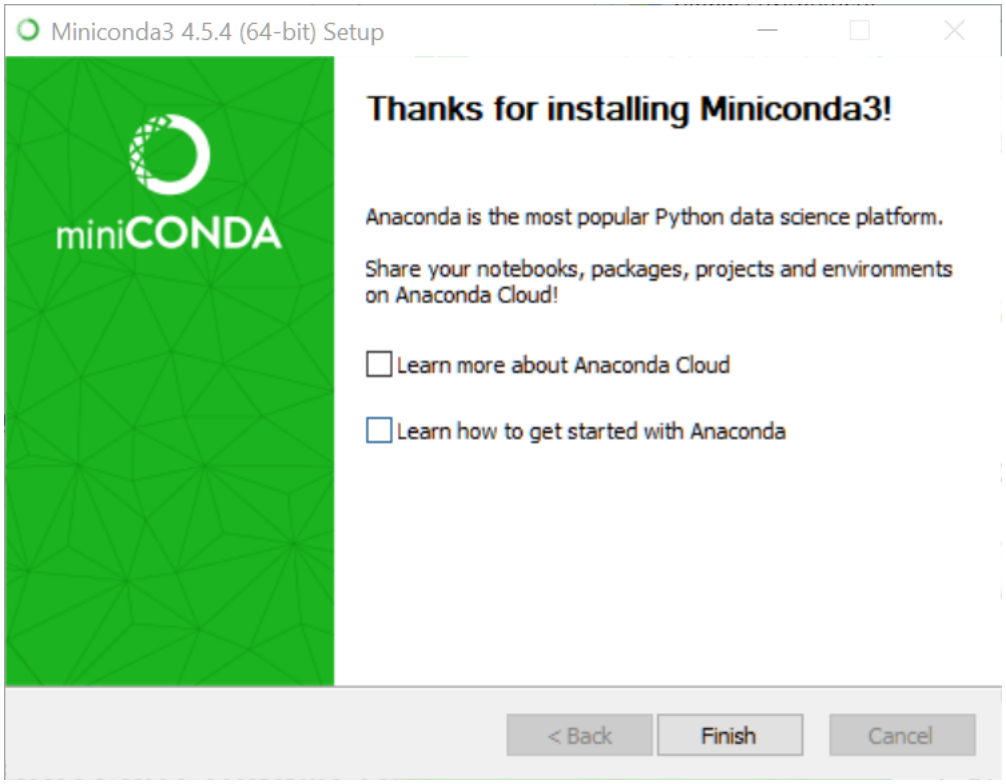
Choose the install location. The default is your home directory, which on my laptop is a small SSD. So I changed it to the `D` drive, which is a terabyte spinning disk. After you've set the install location, click `Next`.



Clear both check boxes and click `Install`.



Click `Next`.



Clear the check boxes and click `Finish`.

3.4 Install `reticulate`

```
if (!require(reticulate)) install.packages("reticulate")
library(reticulate)
```

Did it work?

```
py_discover_config()
```

```
## python:         /usr/bin/python
## libpython:      /usr/lib/libpython3.7m.so
## pythonhome:     /usr:/usr
## version:        3.7.0 (default, Jul 15 2018, 10:44:58)  [GCC 8.1.1 20180531]
## numpy:          /usr/lib/python3.7/site-packages/numpy
## numpy_version:  1.15.1
##
## python versions found:
##  /usr/bin/python
##  /usr/bin/python3
```

Bibliography

Allaire, J. and Chollet, F. (2018). *keras: R Interface to 'Keras'*. R package version 2.2.0.

Allaire, J., Ushey, K., and Tang, Y. (2018). *reticulate: Interface to 'Python'*. R package version 1.10.0.9001.

Galili, T. (2018). *installr: Using R to Install Stuff (Such As: R, 'Rtools', RStudio, 'Git', and More!)*. https://github.com/talgalili/installr/, http://www.r-statistics.com/tag/installr/.

Karambelkar, B. (2017). *docker: Wraps Docker Python SDK*. R package version 0.0.2.