Report on Data Analysis and Machine Learning Project

Zhi Qi

11/20/18

Table of Contents

Abstract

3

Background Information

4

Project Implementation

6

Project Demonstration

12

Conclusion

15

References

15

**Abstract**

This project is a big data analysis project that utilizes Tensorflow, Cassandra, and Docker. The goal of this project is to have a web service that allows user to submit a picture of a handwritten digit then return the predicted number and add the log information to the online database. The web service is created using Flask, and the prediction model is the Softmax model from Tensorflow tutorial. The prediction model is exported using tensorflow.builder, and loaded when building Docker Container under app.py. Next we have the online database. Online database is achieved using Cassandra through Docker. First we launch as Cassandra image, configure it using cqlsh, then link the container we built earlier with this Cassandra image.

Keywords: MNIST, Docker, Cassandra, Python, Tensorflow, Flask

**1. Background Information**

**1.1 Data Analysis**

Data analysis is the method or methods that can be used to analyze data and the process of analyzing it. There are many different forms of data, but people usually think of quantitative data first. These are data such as census data or survey data. There is also scientific data, such as data physicists collect about the cosmos. These days, most people are interested in analyzing the vast amounts of data collected through transactions with a variety of companies and/or websites. Often, though, people forget about qualitative data, which is data that has not yet had any form imposed on it by humans.   
Big Data is a term for data sets so large and/or complex that they are difficult to process using traditional data processing software. Difficulties include analysis, capture, metadata, storage, search, sharing, transfer, visualization, and privacy. Hadoop is an application that helps manage big data.

**1.2 Machine Learning**

Machine learning (ML) is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) from data, without being explicitly programmed.

The name machine learning was coined in 1959 by Arthur Samuel. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

**1.3 Docker**

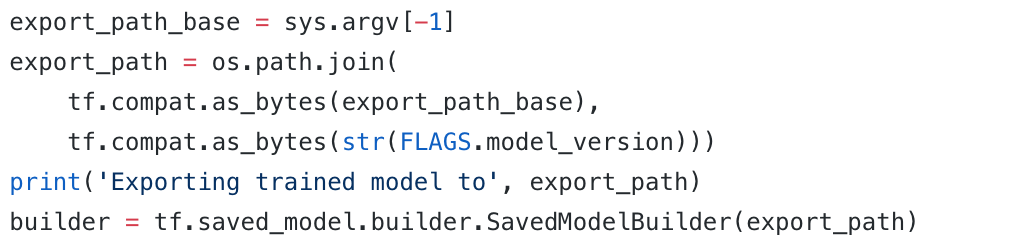
Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

# 2. Project Implementation

**2.1 Exporting Tensorflow Model**

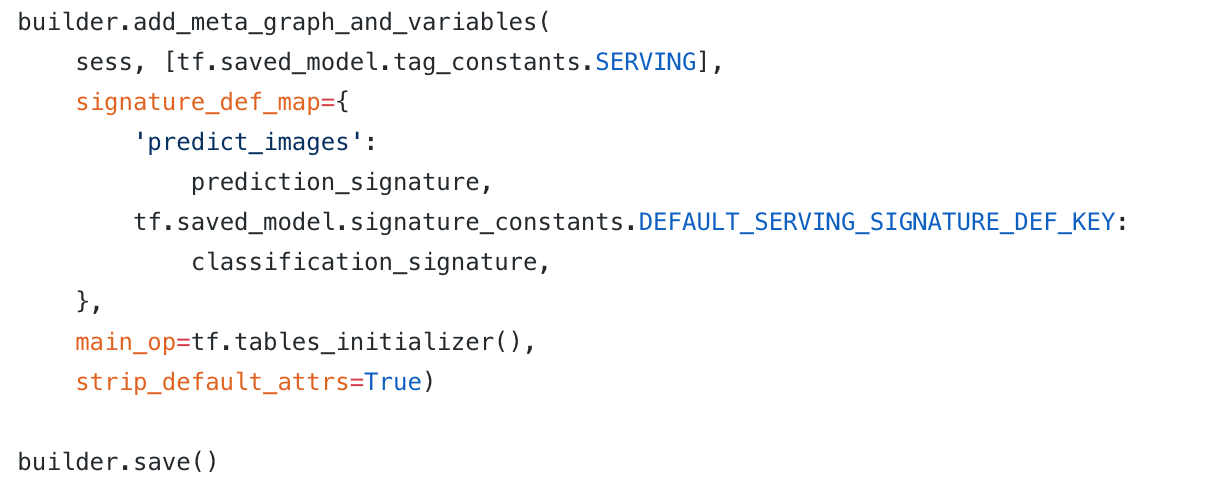
Since Tensorflow is open sourced, for the simplicity of this project, the model that I used in this project is the MNIST softmax model available on Tensorflow tutorial. The code for exporting the model is as follows[[1]](#footnote-1):

First we declare the export path: 

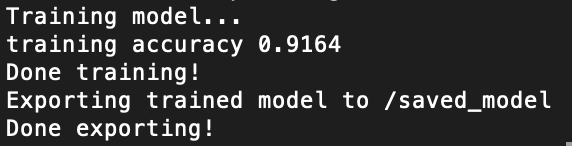
Then we build the signature of the map:



Lastly, we save the model using the standard builder feature:



Now if we run *mnist\_saved\_model.py,* we have:





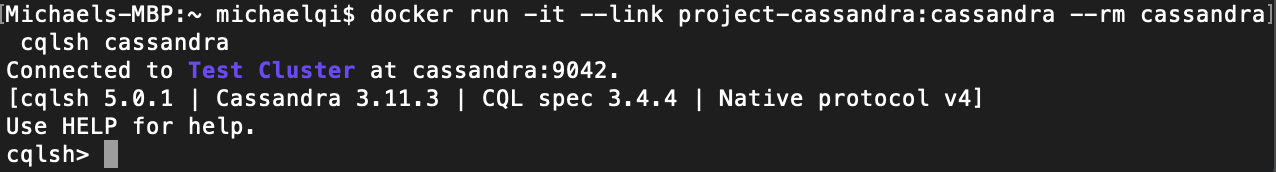
**2.2 Connecting to Cassandra**

This is part of the prerequisite work that needed to be done before building the Docker container. In this project we use a Cassandra instance from Docker.

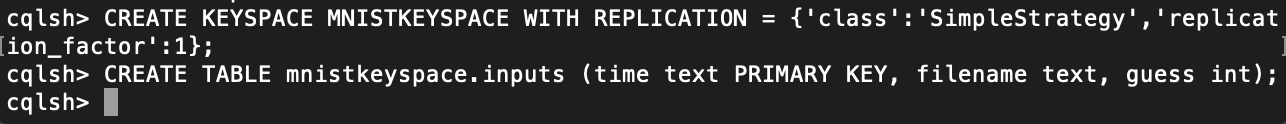
First start a Cassandra instance, name it as project-cassandra:



Then connect to it from cqlsh:



Using cqlsh, we can create a KEYSPACE first then create a table. Here we name keyspace as *mnistkeyspace* and table as *inputs.*



**2.3 Building a Docker Container**

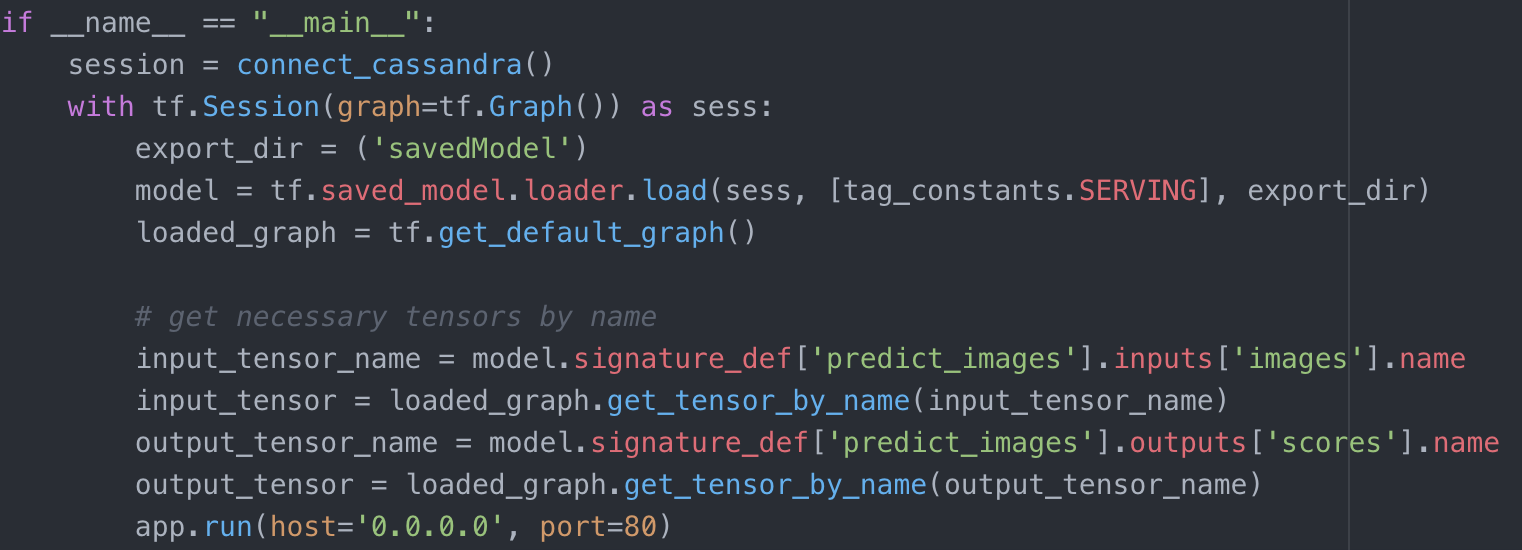
Since we have to serve our service using flask, and that Cassandra will be required as online Database, it makes most sense to build our flask service with our model into a Docker container. To build a Docker container, we need three files: *app.py, requirements.txt*, and *Dockerfile*[[2]](#footnote-2).

*App.py:*

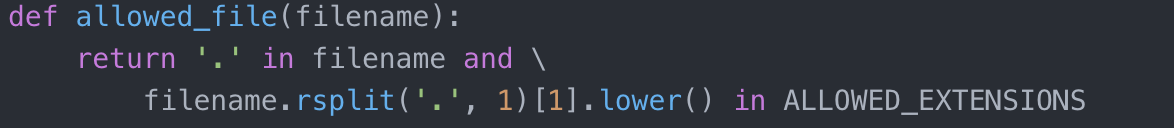
First connect to Cassandra instance that was configured earlier:

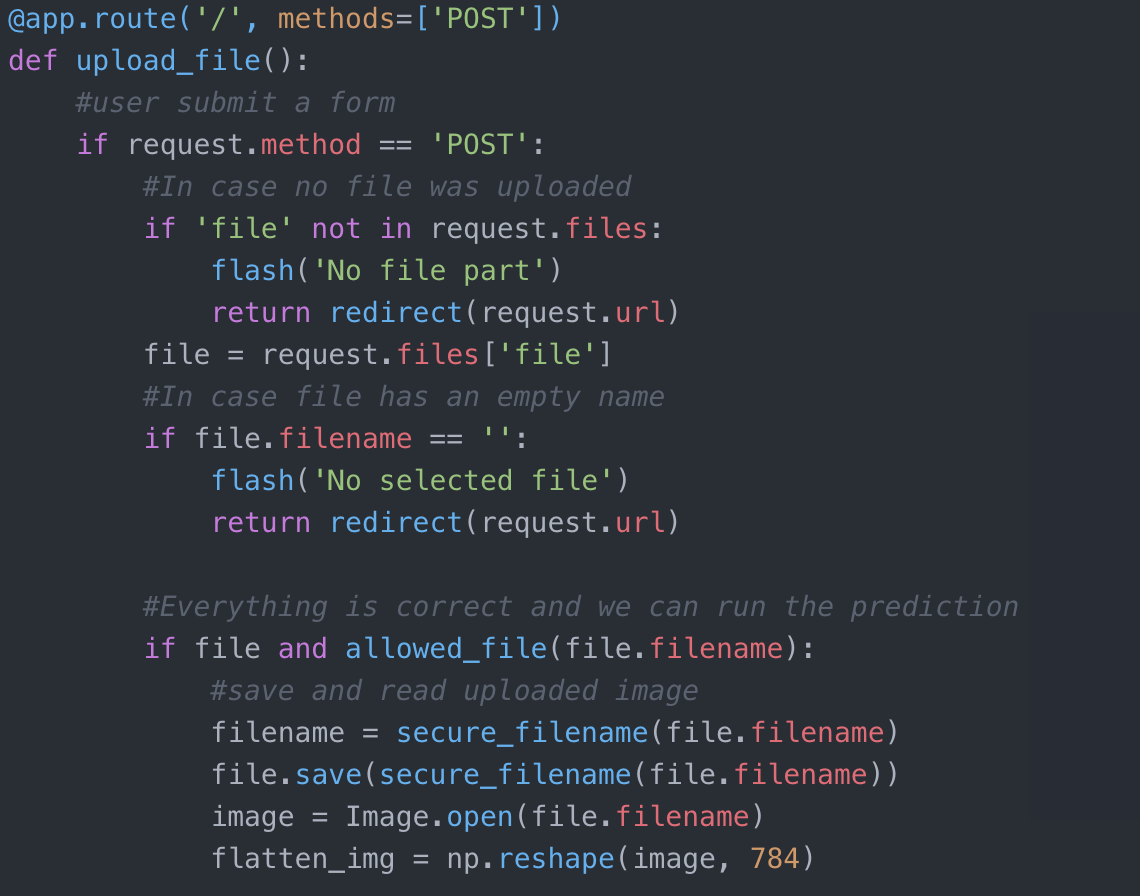


Next we have to make sure to load the model exported above into this program before running.



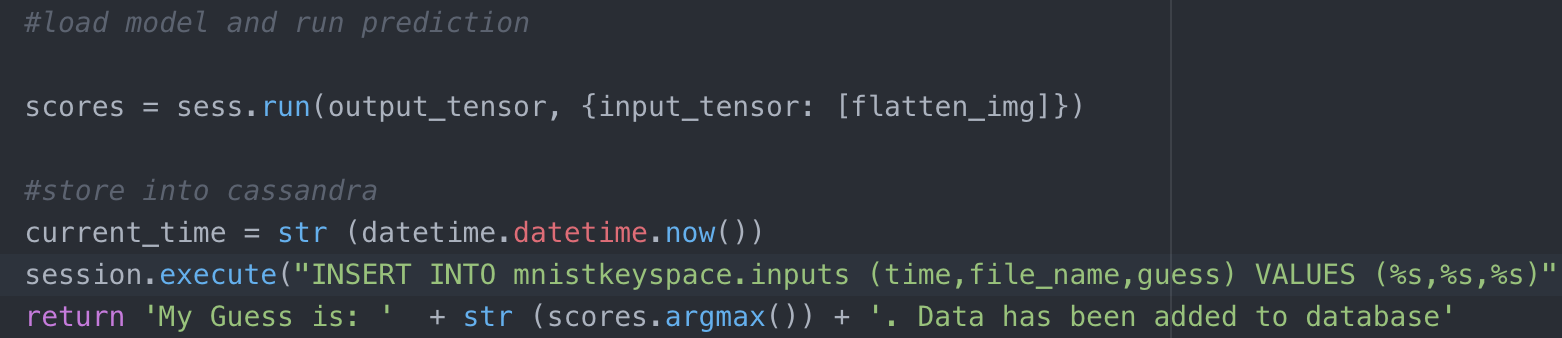
Then we define our app so that it takes a user submitted form (which should be a 28x28 image):





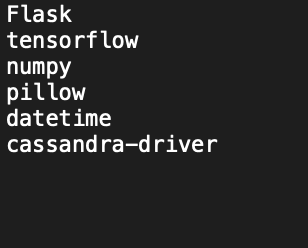
\*Note that we limit the allowed file extension to only .*png*, this is out of the concern that user might submit a php file and mess up the entire program.

Next we take the user submitted image and run it through the model, then return the result as text and store current time, image name, and guessed result to our Cassandra database.



*Requirements.txt:*

This is a very simple file, which lists the required packages for app.py to run.



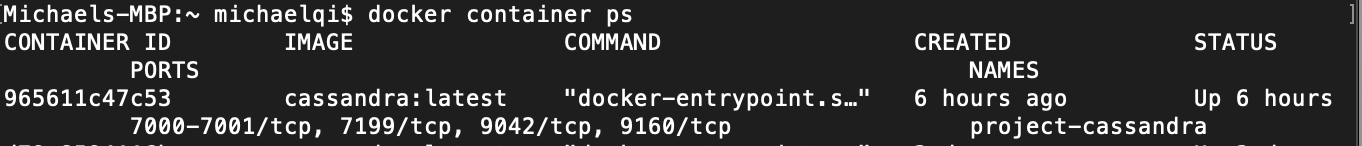
*Dockerfile*:

Dockerfile defines what goes on in the environment inside your container. Access to resources like networking interfaces and disk drives is virtualized inside this environment, which is isolated from the rest of your system, so you need to map ports to the outside world, and be specific about what files you want to “copy in” to that environment. However, after doing that, you can expect that the build of your app defined in this Dockerfile behaves exactly the same wherever it runs.

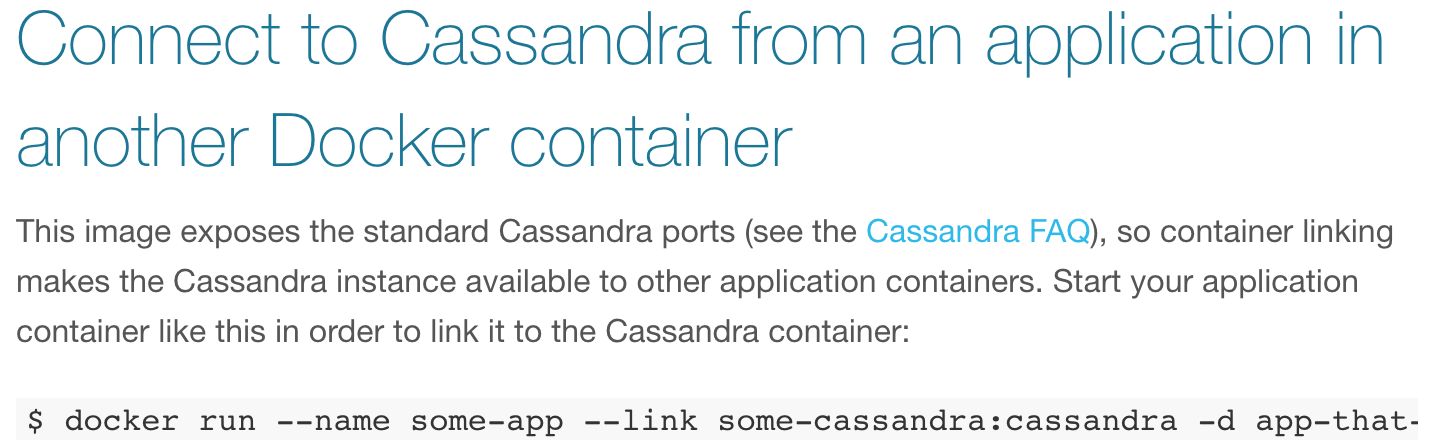
This Dockerfile is written according to tutorial with slight modification\*

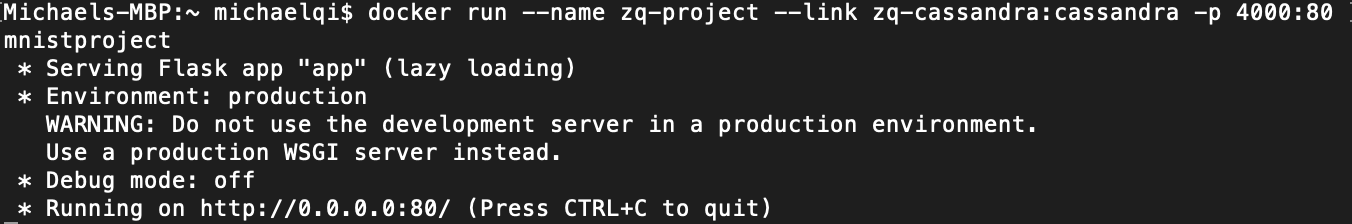
**3. Project Demonstration**

To run this program, first have the Cassandra container running in the background.



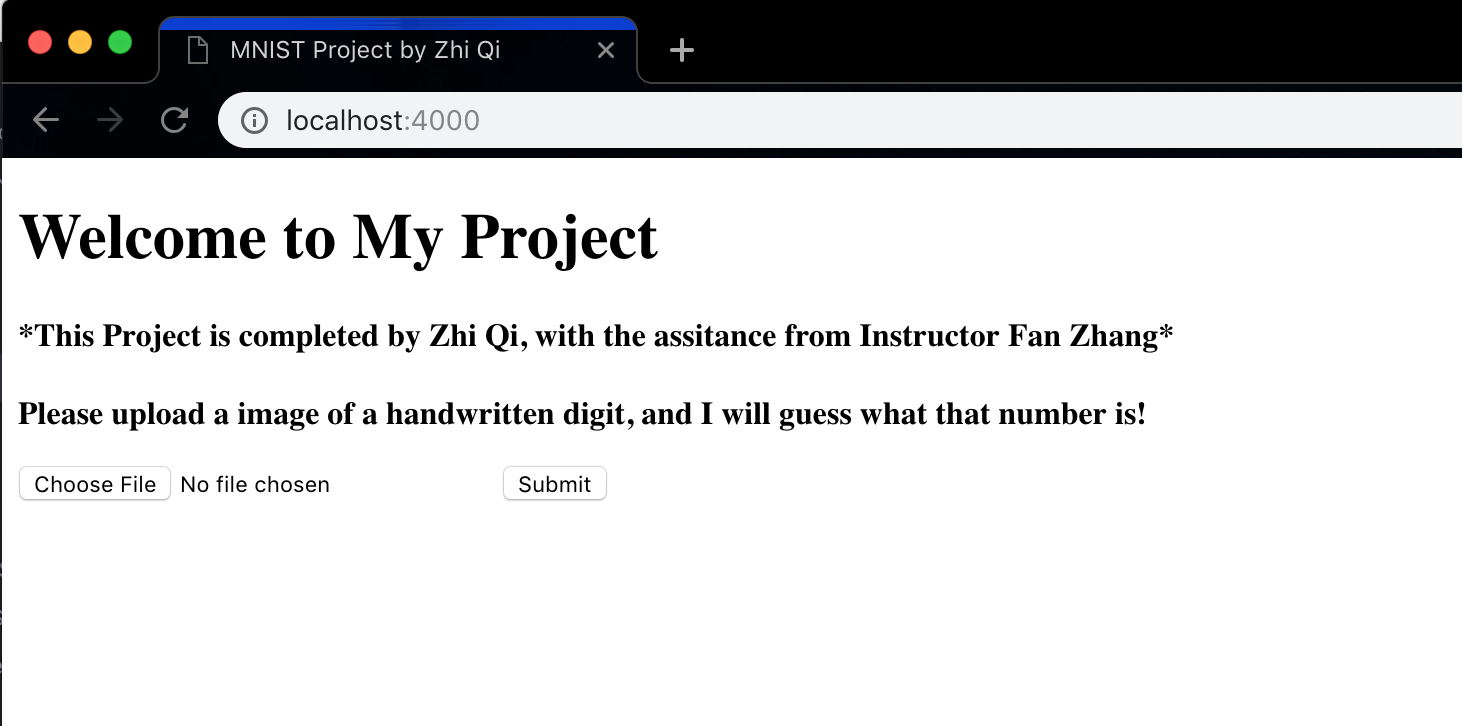
Then run the built container and connecting it to the Cassandra instance according to the tutorial.



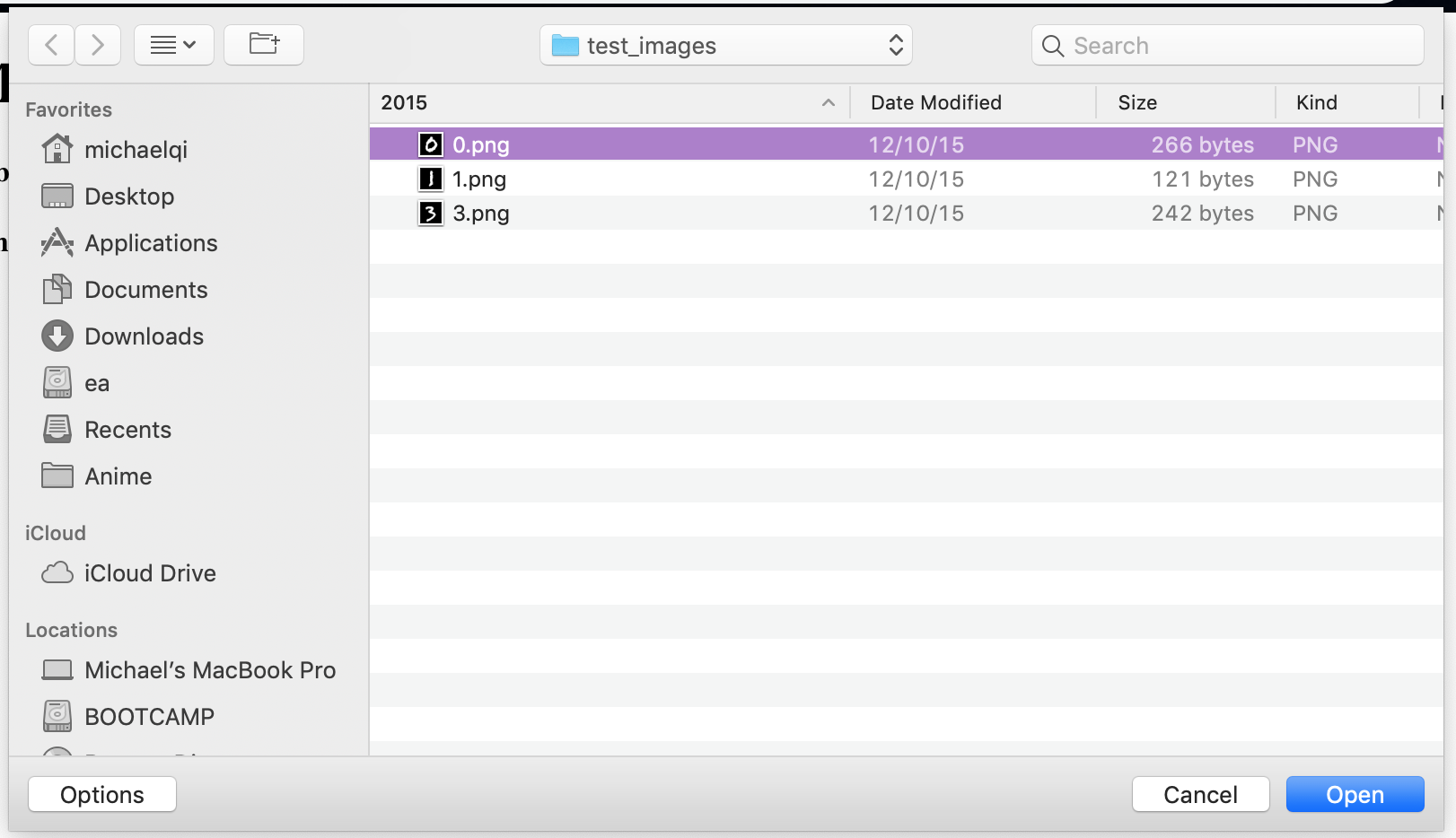


Note that it is necessary to map the port 80 with 4000.

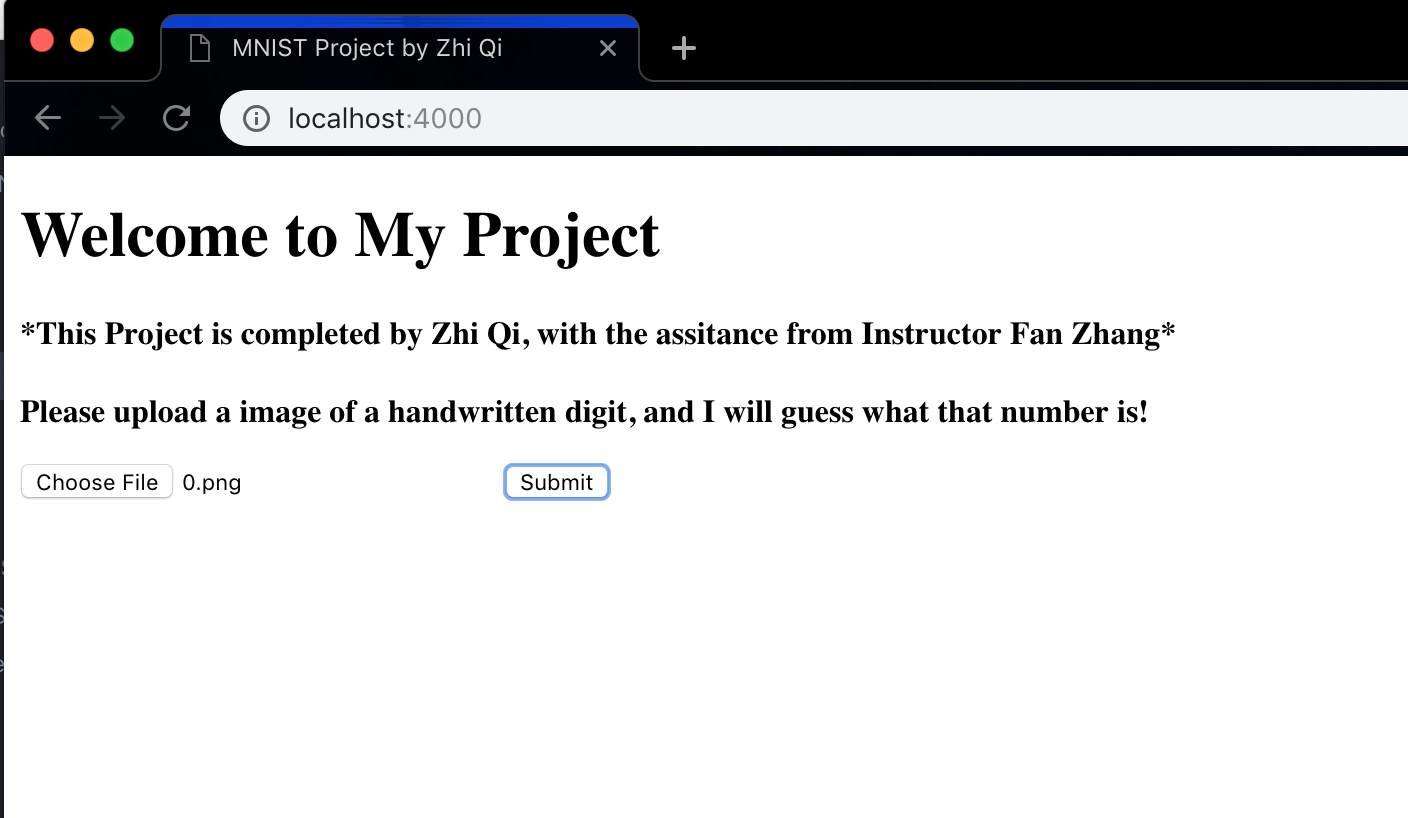
Now we can open up our program at *localhost:4000*



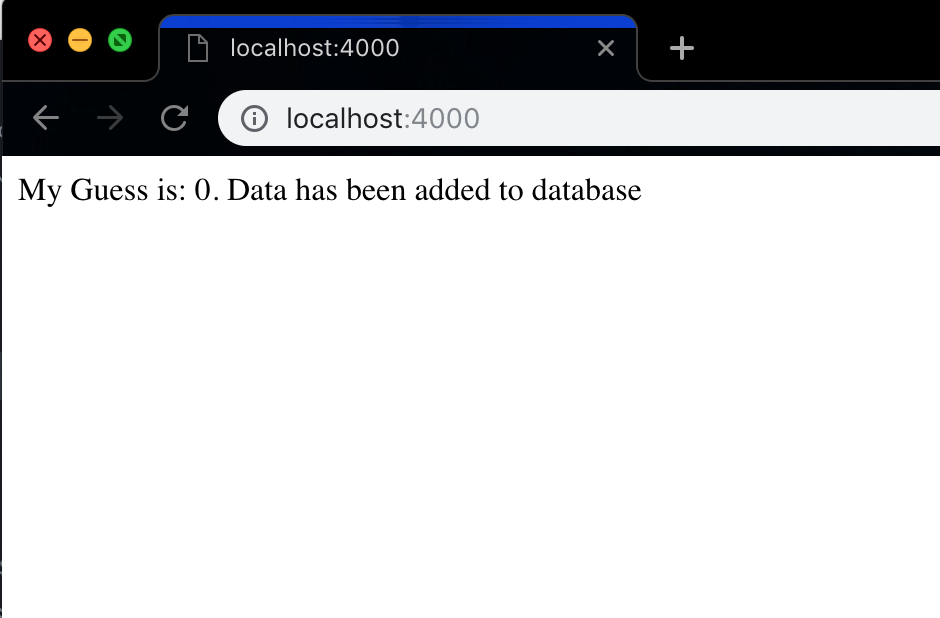
Select a file:



Then click the submit button. This way we submit a form with image as the content.



We have the result



**4. Conclusion**

**4.1 Reflection**

This project was both a challenge and a great learning experience. It is a challenge in a sense that I was new to most of those concepts, and the amount of problems that I encountered and spent hours on was countless. For example, exporting the Tensorflow model was actually a lot harder than I expected, due to the fact that I lack knowledge regarding the technical detail on how the model works. But on the bright side, I did benefit a lot from those hands on experience and it was quite a different experience from class projects. For example, there were no clear guidelines on how to implement this project, nor were the TAs present for all the problems I had. It was more like a solo mission. However, through this project, I was able to have the opportunity to experience what real-world coding will be like when I go out there in a couple for years. I’m am thankful.

4.2 Project Recapitulation

This project is a big data analysis project that utilizes Tensorflow, Cassandra, and Docker. The goal of this project is to have a web service that allows user to submit a picture of a handwritten digit then return the predicted number and add the log information to the online database.

There were three tasks to the implementation. First is the is to extract Tensorflow model, second is to configure Cassandra, and the third is to build the Flask app. The last task can be divided into several parts. First the app loads the model, then accepts the user submitted form which then outputs the result using the model, and in the end add log to the Cassandra instance.

References:

Dhar, V. (2013). "Data science and prediction". Communications of the ACM. 56 (12): 64. doi:10.1145/2500499.

Jeff Leek (2013-12-12). "The key word in "Data Science" is not Data, it is Science". Simply Statistics.

Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2

Samuel, Arthur (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. 3 (3): 210–229.

1. https://github.com/tensorflow/serving/blob/master/tensorflow\_serving/example/mnist\_saved\_model.py [↑](#footnote-ref-1)
2. https://docs.docker.com/get-started/part2/ [↑](#footnote-ref-2)