Deliverable 3: MySQL and Node.js in Docker

Colleen Baksi, Claudia Gajes, Sidney Mantwill
Computer Science Undergraduate Program
West Chester University
https://github.com/sidneymantwill/SCCProject_CSC496

Abstract

The objective of the project was to install and deploy a multi-node computing service on CloudLab. Other criteria that needed to be met: the submission of both a technical report and presentation on the topic, and the submission of the Github repository containing a fully automated and deployable profile. To meet the needs of this project we used a MySQL database accompanied by a Node.js webserver. The MySQL database would contain a list of Pennsylvania area codes, with each code mapped to its corresponding area. In theory, the basic concept and functionality of the project would allow a user to enter a Pennsylvania area code into the console. A search would then be queried onto the database, and if the input code was found the corresponding area would be output for the user to view. The project should additionally be able to function autonomously through the browser with an IP address, but due to certain technical issues it is only able to work locally though the Windows desktop Docker terminal.

1. Introduction

What is the problem

The goal of this project was to provide a computing service with two service nodes through CloudLab's Openstack Infrastructure. Initially there was some confusion about how to begin, leading to a number of struggles to fully understand the project guidelines and requirements. After heavily researching potential project ideas, it was decided that a Node.js web server with a MySQL database as a backend interested us the most. It was planned to have them interact through Docker containers. The specific vision for the assignment was to create a web server that would load a list of area code options, and return a corresponding Pennsylvania city based on the area code a user would input. The input code would be queried in the database, and if it was found the area associated with that code would be returned. Conceptually, the deployment of this project did not seem too difficult, but during the preparation and implementation of the second and third deliverable, a few issues were encountered. Though the idea at its core is simple, the actual execution was challenging and very time consuming. Setting up the server and database on their own were fairly simple tasks, but getting them to function together within Docker containers was a complicated and difficult task. A lot of effort, research, and teamwork was required to get the project to the presentation phase of the second deliverable. The technical presentation for this deliverable went fairly smoothly, but following that there were still the matters of trying to figure out how to get the MySQL database up and working on Docker, and fully automating the profile.

2. Methods

How the problem was solved

As it was previously stated, the objective for this project was to have a site with a list of Pennsylvania area code options that a user can choose to search from. The data for the listed area codes and their corresponding cities is stored in a MySQL database table. Currently, the user is only able to search by using the command prompt of a local machine, and not through the actual webpage. The user must access the webpage before they are prompted to search for a code. The Node.js server connects to the MySQL database to access the table, and then the server runs a query to check if the user's input area code matches an area code in the table. If a match is found, the corresponding city is output for the user to view. If the input area code does not exist in the table, an empty array is then returned. There is currently no exception handling for non-numeric characters, and the server crashes when anything other than numbers is added by the user. There is also nothing implemented to control the length of the number input. All area codes are only three digits long, but currently the user is able to input a number of any length.

The project is currently able to demonstrate a successful proof of concept through the terminal of a local machine. When running the server, one must connect to the localhost and the port number (8080), so the server can then connect to the MySQL database, prompt the user for input, and run the query.

The project can currently be partially demonstrated on the Docker desktop terminal and on CloudLab. The Node.js web server can be successfully containerized on Docker once the Github repository is cloned on either the Docker terminal or CloudLab. The Github repository contains our JavaScript file, the HTML file, and other file dependencies. When the project is launched in the Docker terminal on a Windows desktop, the server successfully runs and connects to the network. This is done by using the default IP address that Docker is configured to along with the port number (8080). The server can be similarly run on CloudLab using the web address found via CloudLab.

Why the problem is not already solved or other solutions

Despite extensive research on this matter, there were struggles with having to containerize the MySQL server. Unfortunately since it is not successfully containerized, the web server throws an error when it attempts to connect to the database. To work around this issue, during the testing phases of the web server, the code that defines and connects to the database must be commented out temporarily. This allows the HTML file to load in the browser and it allows the Node.js server to run just long enough to prompt the user for input. The user must input an area code choice into the command prompt of the local machine, but still a database error is thrown. Since the database does not yet exist, the server then crashes as a result.

Another issue faced included getting the project to work through the Docker branch on CloudLab. The GitHub repository can be successfully cloned and run, but due to a misunderstanding with the web addresses, in initial tests the webpage would load infinitely and each connection test in the environment is unsuccessful. As a result, a new node-head window had to be opened so the container can be manually and forcefully stopped. Fortunately, this issue has since been solved and the proper web address that must be used to connect to the server has been found. Since then, CloudLab tests have been successfully run with similar results as the Docker terminal tests.

The process of automation has been successfully implemented. The user is required to manually clone the GitHub repository to begin. Then, the user must simply run the automation.sh file. This file automatically builds the image for the project, and then runs the container. If the user needs to rebuild the image and rerun the container, after stopping the current one they only need to run the automation.sh file again. This currently only includes building and running the web server image and container, due to the continued struggles with containerizing the MySQL database.

3. Results

What was discovered

While attempting to conduct research on this process, it was found that there were not many resources present on Docker and CloudLab specifically. This made it very difficult to perfect the server and to troubleshoot the problems that were faced. Many solutions were found by using class resources and trial and error. Resources involving MySQL being used with Docker in particular were lacking, contributing to the troubles faced when trying to properly containerize the database. In fact, recommendations found both online and through peers suggested that the best course of action would be to not use MySQL at all, and instead switch to another SQL database management system. While this could have solved many of the problems faced in this regard, there was not enough time to examine all the other potential systems before we could switch.

In addition, the group faced some issues when it came to testing the server. One group member was unable to access the Docker branch on CloudLab, and thus was not able to conduct CloudLab tests. This slowed down the testing process slightly, and made it more difficult to explore issues faced on CloudLab.

The entire project itself was also very time consuming, due to its complicated nature and the many steps needed to make even the most simple process work in containers. Given more time, the group would have had a better availability and chance to look more deeply at the MySQL issue and even start using a different management system for the database.

3. Discussion and Acknowledgements

Self-Assessment on the final deliverable

There were problems when attempting to meet the requirements for the first deliverable. Unfortunately, there were misunderstandings among the group during this phase regarding what would be required in the future of the project. However, once this was pointed out, with time and effort the project goal was reexamined and changed to a more appropriate idea.

The project successfully met the requirements for the second deliverable and the technical presentation possessed the necessary criteria and followed the listed guidelines that were presented on D2L. Though a technical difficulty was experienced when presenting, necessary screenshots of both the web server deployment and the successful demonstration of the Docker desktop component came in handy as a much needed back-up plan for displaying the accomplishments to the audience and to show its full functionality. The presentation also demonstrated both the successful creation and execution of Docker images and containers and the execution of a container. The web server was successfully run in the browser via the default Docker IP address, and despite lacking a database to connect to, the demonstration allowed the user input as intended. This shows the utilization and understanding of cloud computing infrastructures when using Docker. The presentation also acknowledged technical errors, other problems faced, and future plans. Potential solutions were laid out to the audience in order to give an idea of the steps that would be taken to complete the project fully.

While the project did not fully meet the deliverable three goals laid out as part of deliverable two, it successfully meets the automation requirement. Besides automating the profile, the main goal described was successfully containerizing the MySQL database and then test the full implementation of the project. However, problems arose when attempting to implement the MySQL database. This was largely due to the lack of helpful resources on using Docker and MySQL together. When researching the issue and attempting to find methods of containerizing the database, almost nothing of actual use was found. There also seems to be a base incompatibility between Docker and MySQL, given all the suggestions to convert to a different database manager system.

A great deal of effort was put into this project. There was a little bit of a struggle in the beginning while trying to fully understand the project guideline and its requirements. A lot of effort, research, and teamwork was required to get the project to the stage it is in now. One barrier in the way is the lack of online resources and help with using MySQL and Docker together. This forced the group to rely on trial and error which unfortunately is extremely time consuming. In the end, the group was able to largely persevere and make huge strides in the project's progress. While the project is not fully functional on Docker and CloudLab, the base foundation is there and the understanding of Docker processes has been clearly demonstrated.