COSC349 Assignment 1 Report

Angus Fitzgerald-Kearns, 9467284, fitan094
Hannah O'Connor, 2262274, ocoha990

Git commit ID 532354f
Git Repository https://github.com/hroseoc/assignment1

Application Description

Our final application, MeTrade, consists of 3 communicating Virtual Machines: 2 web servers (clientserver and adminserver) and 1 database server (dbserver). It is a very primitive system for a rental company.

The clientserver hosts the webpage for a user to submit a form to rent a pre-loaded item. The dbserver hosts the mySQL database to store the booking information from the client server. The adminserver hosts a webpage for the business side - as it displays who has rented what item (along with their contact details).

Application Instructions

Prerequisites: virtual box and vagrant downloaded.

1. Create a new directory and transport into it.
2. Run `$git init`
3. Run `$git clone https://github.com/hroseoc/assignment1`
4. Transport into the 'assignment1' directory.
5. Run `$vagrant up`
6. Go to:
    a. http://192.168.2.11/client.php ~ to view the client interface of MeTrade.
    b. http://192.168.2.13/admin.php ~ to view the admin interface of MeTrade.

At the client site, fill out the form to rent an item from the dropdown box and hit submit.

At the admin site, view the rented items and the renter's personal details.

Rationale for utilising Virtualisation with 3 VMs

Some benefits of using 3 VMs for our application development include:

- *Simplified task management* - isolating tasks such that the roles for each server were divided up it was easier for us to optimise and debug during development. We could (and still can) make changes to one server without making changes to the other.

- *Elasticity* (ability to rapidly scale up resources to match demands) - if website traffic to the client server was to increase it is valuable to have it running on its own server because the load can be handled without having to be concerned about the admin or database aspects. If one of the VMs is corrupt, then we don't have to worry about the others making it easier to identify and getting it up and running again.

Build process

When running `vagrant up` the total download volumes for ubuntu/xenial 64-bit are roughly 250MBs.

- The initial download volumes for each of the 3 servers are about 19.5MB. The dbserver requires an additional 158MBs.

Subsequent builds require the same storage but not any extra MBs as it's already been downloaded onto the local computer.

Future Development Changes

- Detail 2 different types of change (along with a specific example of each) that a developer might make to the code of your repository, and how they can subsequently rebuild and rerun the application after they have made such changes.

Change 1: Create a login system for clients.
Developers could edit the `client.php` file, enabling the clients to create accounts. The accounts could be stored in the sql database. Then they could edit the `admin.php` file to enable admin users to view the accounts created. To test and re-run the developers would be required to destroy the servers (`vagrant destroy`) as the database would've been altered.

This would help to improve security and create a smoother, more personalised user experience.

Change 2: Add a shopping cart in the client site.
Developers could edit the client.php file to improve the functionality by displaying items selected to the client in the form of a shopping cart. This would only require them to refresh the client site webpage to view and test the changes. Additionally you could implement a refresh upon adding something to the cart.

Our Process and Challenges

When we initially began this assignment at the beginning of lockdown, we quickly discovered that we were unable to use Vagrant on our new macbooks with the M1 chip. This led us down the Docker route, and we commenced to learn about and implement an application using docker and it's tools (specifically, with docker-compose to manage the containers). One example of this can be viewed here:

- https://github.com/hroseoc/Docker-Client-Server-1

This alternative pathway encompassed difficulties, e.g., finding the relevant tutorials and information was time-intensive. As soon as we could get back into the labs on campus together, we decided to return to Vagrant and use the labs that we completed as a starting point to build our application. Although we did not get to showcase a finished application with docker (we only managed to get 3 servers running without communicating or only 2 servers running, but communicating), it was beneficial for us to learn about more light-weight virtualisation.

Once we were able to get back into the labs, we began constructing the servers using Vagrant. The interruptions due to Covid meant that our development time was significantly shorter than we had anticipated. However, throughout the development stage we maintained good practices of systematically constructing the servers with regular testing. Initially, our application was very basic as we wanted to ensure that all three servers connected and worked correctly. Once we had established that the three servers were working, we began developing more user input parameters and building upon our original database. Beyond the functionality of the site, we added a basic .css file to provide some styling to the site, creating a nicer experience for both the user and administrator.

In the planning stages of our assignment, we had discussed implementing more features such as a login page and a shopping cart for the user. However, due to the delays imposed by Covid, we found that we did not have enough time to get these features working correctly. As stated, future developers could implement these features for future versions of the application.

Collection of external resources:

- Solution from Lab 6 by David Eyers: https://altitude.otago.ac.nz/cosc349/vagrant-multivm
- https://www.w3schools.com/php/php_mysql_connect.asp
- https://www.php.net/manual/en/mysqli.real-escape-string.php
- https://www.w3schools.com/php/func_mysqli_query.asp
- https://www.w3schools.com/php/php_forms.asp?fbclid=IwAR1fFVYBlVvTBeufJkODL7Actd8Vl4P-SAwjzT7A81xHe4370VqyvqYIiqU
- style.css is a solution extracted from Cosc212 assignment submitted by Angus Fitzgerald (link isn't on github, but it can be sent through upon request)