COSC349 – Assignment 1

William Duggan – 3409853

https://github.com/wduggan/cosc349_asgn1

Introduction:

My application is a very simple online store and administration application which utilises three different virtual machines that interact with one another. One virtual machine (VM) is used to run the customer facing webserver, another to run the administration webserver, and another to run the database server.

Application Design/VM Interaction:

My application has three VM's, two web servers and a database server - each VM is separated to perform an individual task. The communication between the web servers and the database server is done using a private network, which is setup within the Vagrant file.

I will start by explaining the third VM that is configured within the Vagrant file – the database server. The database server is configured to run on IP 192.168.2.13 in the private network. The database server has MySQL installed on it and holds two different tables within the database – the 'products' and 'users' tables. The users table holds the username and password of the admin for the admin website, and the products table holds the productid, name, description, price, and quantity for different products in the online store.

The first VM is the customer-webserver which can be accessed on the host machine in any web browser from the localhost IP 127.0.0.1 on port 8080 (127.0.0.1:8080) or from the configured IP on the private network at 192.168.2.11. The VM has Apache installed on it for the http web server, and PHP installed for the functionality to communicate with the database server. The customer page opens at a welcome screen, which can then be navigated to a "View Products" page, where the data from the 'products' table in the database is displayed – using PHP to access the database via the IP on the private network. The purpose of this VM webserver is for customers to view the products in the database and then 'buy' a product and have it be removed from the database.

The second VM is the admin-webserver which can be accessed on the host machine in any web browser from the localhost IP 127.0.0.1 on port 8081 (127.0.0.1:8081) or from the configured IP on the private network at 192.168.2.12. The admin website also has Apache and PHP installed on it. The admin webserver opens to a login screen, at which the correct username and password (admin, admin) must be entered in order to access the site's content. The webserver VM sends the forms input to the database server VM and checks if the values entered matches the content within the 'users' table in the database - if so, the

admin is permitted entry. The website displays the product data on the site, but also allows for an admin to add new data to the database - providing a simple form of database management functionality from the webserver. Changes made on the admin site can be seen on the customer site.

By splitting up these different aspects of the application and delegating them to separate VM's, each VM can focus on one individual task and complete it as well as possible. Whereas if we had everything within one VM, it may slow down the speed of the machine and the processes within. It also keeps different aspects of the application separate from each other which offers greater simplicity when needing to alter only one part of the application.

Software Components and Download Volumes:

- This application uses the Ubuntu/Xenial 64-bit VM for the VM box. The first time that 'vagrant up' is run, the Vagrant box will need to download Ubuntu/Xenial, which has a file size of approximately 278MB. This only needs to be downloaded once as the file is stored locally, so any other time 'vagrant up' is used, it won't have to be re-downloaded.
- At the start of the provisioning for each individual VM, the command 'apt-get update && apt-get upgrade -y' is run. This gets all available updates for the Ubuntu packages from the Ubuntu website, and then upgrades them for that VM. After running the command for each VM, 19.5MB of content is installed on each VM.
- For the two web servers, an additional 20.8MB of disk space is required for each one after running the 'apt-get install -y apache2 php libapache2-mod-php php-mysql' command. This command installs the Apache and PHP packages on the webservers.
- For the database server, an additional 158MB of disk space is required for each one after running the 'apt-get -y install mysql-server' command. This command installs the MySQL packages on the database server.
- This entire 'vagrant up' process takes approximately 4 minutes and is all done automatically and able to be left unattended.

How to use the application:

In order to run the virtual machines for the application, VirtualBox will need to be installed from:

- https://www.virtualbox.org/

In order to run the provisioning of the virtual machines for the application, Vagrant will need to be installed from:

- https://www.vagrantup.com/

Then the repository containing the application will need to be cloned from GitHub. In the terminal, navigate to the desired directory you want to clone the repository into and run the command:

- git clone https://github.com/wduggan/cosc349_asgn1

Once cloned, use the terminal to navigate into the directory of the repository where the Vagrantfile is contained (this can be checked with the 'ls' command). Then to build and start the virtual machines, run the command:

- vagrant up

Once the process is complete in the terminal, you can open the VirtualBox application to see that all three VM's should be there on display (optional). Now to interact with the application. To see the customer website, enter the following into any web browser:

- http://127.0.0.1:8080 or http://192.168.2.11

The customer page opens at a welcome screen, which can then be navigated to a "View Products" page, where some pre-loaded products/data from the database is displayed to the user – as well as any new data added by the admin.

To see the administration website, enter the following into any web browser:

- http://127.0.0.1:8081 or http://192.168.2.12

The admin webserver opens to a login screen, at which the correct username and password (admin, admin) must be entered in order to access the site's content. The website then displays the product data on the site (the same as the customer website), but also allows for an admin to add new products to the database and to the page. This is done from entering information and submitting the form on the 'Add Products' page. Changes made on the admin site can be seen on the customer site.

Feature failed to be implemented:

As explained previously, the purpose of the customer webserver VM was for customers to view the products in the database and then 'buy' a product and have it be removed from the database. However, I was unable to achieve the 'buying' aspect of the application due to a lack of time. I did try doing some research and looking for ways to do this online, and I attempted to do some of these methods but could not get them to function before the due date of the assignment. Given more time to research and test I believe this would not be too difficult to implement.

Possible Changes and Extensions of the Application:

One helpful modification that would be good for the application that a developer could make, would be to fix the feature I failed to implement as mentioned above – the 'buy'

button on the products in the customer website, and potentially add a 'shopping cart'. This would allow the user to add products to a shopping cart, then click 'buy' within the shopping cart and have the relevant items and their quantities be removed from the database and displayed to the user.

Another helpful extension that could be made to the application would be to implement a logout function for the admin website. As currently the admin can login, but cannot logout. On top of this, some sort of session storage should be implemented to keep track of the admin. This is because right now for example, if you sign in as admin, nothing is actually saved to the session saying you are the admin; the application just checks the username and password against the database and redirects you to the correct page if the info is correct. This means that after signing in to the website, if the user presses the 'go back' button in their web browser (taking them back to the login) and then presses the 'go forward' button, they will reappear inside the admin website which is not very secure at all. Adding this addition as well as a proper form of logging out as admin would provide a layer of security to the application and validate who is accessing the admin pages.

Another addition that could be added would be the ability to delete products from the database from the admin website. Currently the admin is able to create products and have them be added to the database, but it would probably be good to also be able to remove products.

Reasons for handing in late:

The reason I am handing in late is because I believe that being in lockdown had a big impact on my ability to learn and do work. Since I always do my work in the computer labs, it was a bit difficult getting everything setup on my own personal computer to do the assignment as well as other lab work, and getting familiar with the environment – especially since I am running on windows in which the terminal commands are different to the CS labs. I was also undertaking this assignment by myself rather than in a pair with another student. This is because I only know a few people in the class by face and not by name, so when lockdown came along, I didn't know their names to be able to message them on social media and try to link up to do work together. This meant I had no other people to communicate with besides David to ask for help from, and although David was helpful, it isn't quite the same as being on campus and being able to ask for help. Whereas if we were on campus, I would have been able to discuss ideas and concepts with other people, which would have helped my understanding of things required for the assignment. I did greatly struggle with understanding some aspects of the content and so I relied upon researching online to help with this – and sometimes I had problems which I couldn't find a solution to online. And on another note, I have spent the lockdown period in Twizel with my family, where the internet connection is not very good, so sometimes I was not able to access the internet for help or message David as there was no internet connection sometimes.

Despite these reasons, I have been grinding away at the assignment for long periods of time to try and get it done in time for the original deadline, and just haven't quite made it due to

struggling with certain problems that I didn’t know how to solve until David offered me some help. And currently I am only handing the assignment in a day late, and I could have continued to go through the rest of the 10-day extension period to potentially finish adding that ‘buy a product’ feature I mentioned I failed to add above, but I wanted to honour the original due date and try get as close to that as I could.