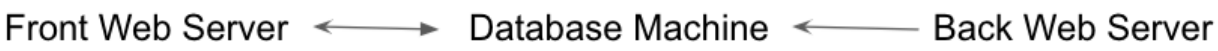
COSC349 Assignment 1

Alex Lake-Smith 5400306
https://github.com/AlexLakeSmith/349-Assignment-1

The aim of this assessment is to understand the fundamentals of building three separate but interconnected virtual machines that provide an online application that a user can access. My idea was to build a basic front facing web server which users can view a list of songs that have been added to my Birthday playlist, a back facing web server which admin users can add songs to the playlist and finally a database machine that stores the entries.



As seen from the image above, the front web server interacts with the database machine to access it's contents and display them in a table. The back facing web server interacts with the database machine when users correctly input a song into the form provided. This will then update the database and can be seen on the front facing webpage.

These machines have been established on a private network which is what allows them to interact with each other and is done in my VagrantFile configuration. I have also included my provisioning scripts within my VagrantFile to make my program more compact and more readable, also allowing for an inexperienced administrator to rebuild my virtual machine's setup easily without having to create multiple shell files.

Installation/Setup Instructions

To run my three VM's you will need to install the latest version of Vagrant and VirtualBox which are both linked to below.

- Vagrant: https://www.vagrantup.com/downloads
- VirtualBox: https://www.virtualbox.org/wiki/Downloads

You will then need to make a clone of my git repository which can be done by using the terminal or by downloading an archived zip folder of all of its contents. Instructions are shown below for both these methods.

1. Terminal Method
Type into your terminal window this command:
git clone https://github.com/AlexLakeSmith/349-Assignment-1.git

2. Download contents as a .zip
Type the following into your browser window:
https://github.com/AlexLakeSmith/349-Assignment-1/archive/main.zip
Once this has been done you then navigate to the directory the repository has been cloned/downloaded to and enter the command: vagrant up

You can then access the front facing website which can be used to query current songs in the playlist by searching the following address in your browser:
http://127.0.0.1:8080/

You can alternatively access the back facing website which can be used to add new songs to the music database at the following address:
http://127.0.0.1:8081/

In your report, detail two 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.

Future Modifications

There are a few ideas that come to mind when thinking about ways a cloud developer could take my repository and build it to their liking or expand on how I have currently designed it. When building my initial configuration I referenced Lab 6 quite regularly and due to my lack of experience using html, css or javascript to create web applications I wanted to keep my idea relatively simple. This way it would allow me to create an application where I learnt some basic skills using these languages but also would allow for somebody who may have similar skill levels or understanding as myself to be able to take my idea and expand or replicate it in whatever way they choose.

One way that I believe my application could be extended is more control over the database through the back facing web server. This could be done by incorporating more submission forms which could allow the admin more privileges. Currently my back facing web machine only allows for songs to be added to the database but perhaps with more forms you could allow for songs to be removed, certain artists or albums could be added or perhaps you could add a certain genre all at once.

Currently only the admin user has access to adding songs to the playlist through the back facing website so I thought another extension could be to include forms that public users could submit on the front facing website to request for songs to be added to the playlist. New SQL tables could be created and submission forms could be used to insert information into my database file for the admin to view, the admin can then decide to add these songs based on the requests that they have been provided.

Another possible way a developer could also extend my front facing web server is by implementing a Spotify music player widget into the html code. If a playlist was created to correspond with the database it would allow for users to listen to the songs currently in the playlist directly on the website.

Once these changes have been made to the subsequent files the developer would then need to either reload the machine where changes had been made or completely destroy the current

machine's state, vagrant up and then wait for the machine to be rebuilt. To do this the developer would need to type the following commands:

- *vagrant status - Checks if any virtual machines are currently built with vagrant*
- *vagrant destroy - Will destroy any machines currently built in that directory prompting either y/n*
- *vagrant up - This command will then build the machine from the VagrantFile in the current directory.*

As mentioned above Vagrant reload can be used to reboot VM's and push changes however sometimes if changes are made to the VagrantFile or IP allocation then it is best to destroy the machine and then rebuild it, this ensures the changes are up to date.

Troubles Faced

Apache2 Default Page/Missing .Cnf File

One of the earlier issues I faced when trying to set up my web server machines was to get my own index.php files to show up as my webpage instead of the default Apache2 page which is displayed when you initially setup an apache web server. I used online resources such as Stack Overflow, Youtube and Google to help try and establish what was causing this problem and also contacted David for advice on how to incrementally step through and resolve it. He advised that I look into my apache configuration on my webserver machines and through this I found that my test-website.conf/back-website.conf files were not established correctly. Once I fixed this I could then see my own html files that I wanted to use.

Database Machine Issues

```
==> dbserver: Running provisioner: shell...
    dbserver: Running: inline script
    dbserver: Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
    dbserver: Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
    dbserver: Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
    dbserver: Hit:4 https://esm.ubuntu.com/infra/ubuntu xenial-infra-security InRelease
    dbserver: Hit:5 https://esm.ubuntu.com/infra/ubuntu xenial-infra-updates InRelease
    dbserver: Get:6 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
    dbserver: Fetched 325 kB in 2s (139 kB/s)
    dbserver: Reading package lists...
    dbserver: Reading package lists...
    dbserver: Building dependency tree...
    dbserver: Reading state information...
    dbserver: mysql-server is already the newest version (5.7.33-0ubuntu0.16.04.1).
    dbserver: 0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
    dbserver: ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
    dbserver: ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
    dbserver: ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
    dbserver: ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
    dbserver: ERROR 1050 (42S01) at line 1: Table 'music' already exists
```

When I was trying to initialise my database and get my table results to load on my front facing web server's page I struggled with a few issues. One of the issues that I spoke to David about were the error messages that were stated above. The problem that was causing this was the fact that I was trying to re-provision a machine that was not built for such a function. David suggested that if I was to make changes to my machine that I should vagrant destroy and then build again. For example the error above saying "Table 'music' already exists" was because I was trying to recreate a table which was already established in my machine. By rebuilding my machine I was able to resolve all the issues above. This has now become common practice

when making significant changes to my machine. I also learnt that it is important to keep copies of my vagrant machine states so that if something breaks I can work backwards to find out the root cause of what went wrong. I also struggled to get my database file's entries to load onto my webpage but I then realised later that I had used the wrong $db_name on my index.php file compared to what it was on my VagrantFile initialization (silly error but an easy fix).

Form/Database Updating

This was the most challenging issue to fix and took me over a week to resolve. I used online resources to try and find examples on how to use html forms to insert data into my SQL table and based my initial back facing web server code off of this. I brought my issue to Pradeesh and he gave me some tips on how to debug my php code incrementally using sudo tail -F /var/log/apache2/error.log. My initial form action was left blank so my php code didn't have any idea on what to do with the information it was being passed. This caused an internal server error 500 message. This is why I created a separate php file called insert.php which was just used for handling the information that was passed to it through form submissions. I was initially using PDO to try and enter my data into my database but struggled to get it to work so I moved to mysql functions. These also didn't work as they had been deprecated in an earlier version of php. I then switched to using mysqli functions which I managed to get to work for insertion. Other errors I ran into along the way were debugged using the apache error logs and included calling functions incorrectly, not creating variables which linked to my post data to insert into my SQL table etc.

COVID Lockdown

I thought I would also add that I faced a few issues through the snap COVID lockdown. Initially I struggled with working through both the lab work and assignment when trying to install multiple VM's as I didn't have enough space on my laptop hard drive. I had to install some cleaning software and wipe unnecessary files which seemed to help. I also found that destroying/removing VM's that I wasn't using or didn't need helped free up space. I struggled with not having a study space such as the labs to try and get work done but made the best out of the situation we were all in. I found that regularly asking questions in the pop up Zoom tutorial times or lab times were helpful if I got stuck but I feel I learn better through face to face interaction where I can ask questions and be physically shown what I'm doing wrong or what ways I could go about fixing these issues through debugging. Both David and Pradeesh were both very informative whenever I had questions and helped clear up some of my issues which have been spoken about above.