COSC349 Assignment 1: Effecting the portable building and deployment of software applications using virtualisation

Group:
- *Hugo Baird, 5109978*
- *Cedric Stephani, 8916960*

Job Listing Noticeboard - Application Overview:
This application aims to give users a platform to post available job positions using an administration form. This gets displayed on the public noticeboard where job seekers can find work and contact potential employers in a simple and reliable way. This is especially useful when it comes to students looking for one-off jobs for extra cash.

Virtual Machine Interactions - How:
The Job Listing Noticeboard application uses three virtual machines, two of which are web servers and one being a MySQL database server. The web servers do not interact with each other directly, but instead both directly interact with the database for read and write operations of job listings.

The first virtual machine configures a web server which hosts a PHP web form. This form provides potential employees the ability to input job listing details, which includes simple fields such as job description, location, and contact information. When submitted, the information gets POSTed to the MySQL database named 'joblistingdb' and stored in the table 'JOB_LISTING'. The MySQL database is directly connected to this web server using PHP configuration to outline the database IP address, MySQL username, database name etc. The data gets POSTed to this database using a prepared SQL INSERT statement typed in PHP which matches the input data to the correct job listing column accordingly.

The second virtual machine configures the MySQL database and inserts the mock data in the table. The MySQL database server is first installed and booted using the Vagrantfile with the MySQL username, password, and database name pre-configured automatically. Towards the end of configuration, an SQL script is called to create the 'JOB_LISTING' table which outlines all the fields/columns with their types accordingly. This is followed by several INSERT statements to fill the database with mock data on start-up which is autonomous.

The third virtual machine configures the second web server which hosts an HTML table containing all job listings available. This table contains all the details of the job listing, including contact information to get in touch with the employer. The index file fetches the job listing data directly from the MySQL database server. This is done using PHP configuration to again outline the database connectivity details such as IP address, MySQL username and database name which all must be accurate to connect to the database accordingly. This data is fetched using the SQL 'SELECT *' statement from the 'JOB_LISTING' table to retrieve all rows and fields to output to the user.

Virtual Machine Interactions - Why:
One reason why the web servers are hosted on separate virtual machines is due to web traffic. The Job Listing Noticeboard will get the bulk of the traffic as the ratio of job seekers to employers is quite high when it comes to looking for job opportunities. Thus, the job form and noticeboard being on their own web server allows for improved scalability and robustness to handle user load and surges of traffic. The job listing noticeboard will have significantly more traffic compared to the form due to the nature of job searching.

Another reason why the web servers are hosted separately is due to our mindset of a microservice architecture. This is very beneficial when it comes to services being down due to potential web server issues. This first of all makes it much easier to locate the issue and fix it quickly to get it back up and running. It also allows only one server (etc. the job listing form web server) to be down, while another server runs normally (etc. the job listing noticeboard). Thus, users can still look on the noticeboard for jobs while the form server is under maintenance. This would be quite crucial if the traffic and number of users scaled quite high, requiring consistent and reliable service access (where server/platform wide downtime should be avoided). It is best to minimize this by keeping at least one significant part of the platform operating whilst another is down.

Approximate build/download volumes:
The first time Vagrant is used, it must download the Ubuntu Xenial 64-Bit VM which gets stored locally and has a size of around 270MB. To test the build/download times, we ran the process of executing ‘vagrant up’ and then ‘vagrant destroy’. The initial build time for our application is approximately 4 minutes and 40 seconds and subsequent build times were similar; however, this will vary depending on computer and download speed. Additionally, when the virtual machines are stopped which is done by using ‘vagrant halt’ (to shutdown), and then ‘vagrant up’ again, it took approximately 1 minutes and 20 seconds to reboot and start running the service again.

How to run the application:
To run the application the user first needs to open a terminal window. From here they will need to change directories into where the application is being stored. From here first run the command “vagrant --version” to ensure that vagrant is installed on your device. It should give a response of “Vagrant version 2.2.7” as of the time of writing this. Next, we want to run the command “Vagrant up” which will run the startup scripts.

How to use the application:
Now that our application is running, open up your desired web browser. In the URL search bar type “http://127.0.0.1:8080/index.html”. This will bring you to the job listing form. From here enter all the information about your job listing into the relevant fields and then click submit; you will get a confirmation alert if the operation is successful. To view the job listing noticeboard click on the redirection link on the form webpage or go back into the URL search bar and go to this address: “http://127.0.0.1:8081/index.html” to view all the job listings.

Two modifications/extensions to make:
One useful modification that can be made by another developer is a create account and login feature. This would mean that only validated accounts could create a job listing and publish them onto the job noticeboard rather than any user that has access to the URL once the platform is deployed publicly. It would also allow each listing to be linked back to an account which could allow employers to edit their listings if previously posted or delete them later when the position has been filled. To do this we would create a user table in the MySQL database and add webpages for both sign in and create accounts. Once the user is logged in he/she would then be on a page which shows their current active listings and with clear options to modify them. Additionally, when logged in, users have access to the two pages to create a new job listing and view the job listing noticeboard.

Another modification another developer can potentially add is a filter/search feature on the job listing noticeboard page. This would allow a user to filter all the job listings by particular categories (for example: a certain field that a job seeker is searching in). The search feature would allow a user to enter keywords and have a result of listings matching the input. A developer must add PHP configuration in the web pages to make this work with the database by adding SQL statements for both the filter and search feature. This configuration must keep the format of prepared statements to ensure users will not be able to take advantage of SQL injection attacks, which otherwise could make the system vulnerable.

After these changes have been made by the developer, they must execute “vagrant destroy” to terminate the existing servers that are running, and then again execute “vagrant up” as this would start the application up again with the new features added to it this time, assuming no bugs occurred during the start-up process and implementation/configuration of the new features are correct.