COSC349 Assignment 1

Vincent Lee 5687923

This application uses Vagrant to set up three virtual machines, each with its distinct purpose. Among these, there are two web server machines and one dedicated database machine. The first web server, known as webserver, functions as the public-facing website that customers interact with when visiting the online store. Meanwhile, the second web server, known as the adminserver, serves as a platform for administrators to manage products and perform actions like adding or removing products. The database VM is responsible for housing the MySQL database, which stores essential product data, user information, and admin accounts for the online store. The web servers interact with the database to retrieve and update information, ensuring that the website functions smoothly and securely.

There are many benefits of using separate VM's rather than aggregating functionality into a single server.

1) Isolation: Since each VM operates in an isolated environment, this can help contain security breaches. If one of the VM's is compromised, it is less likely to affect others. By dividing up the web servers and the database in to distinct VM's, it helps protect sensitive data by reducing the risk of unauthorized access to it.
2) Resource allocation: Each VM has their own dedicated resources, making it easier to allocate resources appropriately when needed and prevents conflict over a shared resource.
3) Consistency: Having three VM's for three separate functions allows for a more consistent and organized application. Firstly, it ensures that each virtual machine is designed and configured precisely for its intended purpose. Consistency also helps with maintenance and updates. Since each VM has a designated role, any modifications required can be implemented with a low level of disruption to the overall application. This means that updates that are specific to one VM won't affect the functionality of the other VM's. This makes the entire application more stable and reliable. This also helps with debugging and troubleshooting. Since each VM has a distinct purpose, it is easier to pinpoint issues and track performances for each function. If there is a problem, you can easily identify where it will be due to the isolation of the functions.

Software components the build process will download:

Build-adminserver-vm.sh & build-webserver-vm.sh:

This provisioning script installs three packages: Apache2 (a web server), PHP (a server-side scripting language), and PHP modules for MySQL database interaction. The total size of the packages is approximately 162.6 MB.

Build-dbserver-vm.sh:

This provisioning script installs mySQL server. The total size of the packages is approximately 200 MB.

These numbers were calculated by using the command: dpkg-query -W -f='${Installed-Size;8} ${Package}\n' | sort -n. This shows a package list sorted by size. This command was run for each provisioning script.

For subsequent redeployment, the package manager dependencies are cached. This means that if the specific version of a dependency is downloaded during the first build, it won't need to be re-downloaded unless there are updates or changes to the dependency.

Each of the Vagrant Boxes take up approximately 2.5GB of storage on the disk.

Changes:

The first change I would make is to create a more secure database by enforcing different levels of access control, such as read-only access for users and full access for administrators. Additionally, separating user and admin information into a separate database can add another later of security and organization. To create another database, a new vagrant box will have to be added to the vagrant file. Along with this, a new sql file will have to be created that creates the new database and tables. Another provisioning script will have to be created to create the database server and other required variables such as users/databases. After this, a simple "vagrant up" should initialize and create the new database server. Within the .php files, changes will have to be made to ensure the right information is gathered from the right databases. To enforce different levels of access control, within the build-dbserver-vm.sh file, a new user would have to be created with the appropriate level of access control.

Another change would be to enhance the website's functionality. Currently, the website is a test template to demonstrate the synergy between the three VMs. To enhance it, you would need to incorporate additional files into the /www directory. These files will enable actions like adding items to a cart, browsing, and categorizing products. These modifications should automatically update if a current file is being modified. If a new file is added, a simple "vagrant up --provision" command should suffice to showcase the updates.

References:

David Eyers - https://altitude.otago.ac.nz/cosc349/vagrant-multivm

ChatGPT was used to create a basic style to the website.