Assignment 1, Cloud Computing Architecture

Brodie Bosecke, 5471718
Meggie Morrison, 7777435

Github repository containing our files - https://github.com/bbosecke/Asgn1

Introduction
The application that we have created is a website and database that registers new users, and records their information and preference for either skiing or snowboarding, for BroMo mountain. The database has a variable, runs, which represents the number of runs the skier or snowboarder has completed throughout the season.

When a user registers on the website, a physical myPass card will be created with their username and full name. On the users first visit to the mountain, they will be issued the physical myPass card which allows them access to the lifts on the mountain, and can track the number of runs they make, for ‘hi-score’ purposes.

The application takes a username and full name from a registering new user, and this information is stored in a database for later use by the admin. If future developments are made, the administrator could be given the ability to update the users' run count or delete a user if they break the rules (sharing a physical myPass card with a friend).

3 VM explanation
The application that we have created uses three virtual machines (VM) to run. We have separated these to differentiate between user and database privileges, as well as to divide up the tasks handled by each server.

The first VM, webserver, hosts a web server which allows the user/client to access the webpage and input their information which is then sent to the second VM, the database.

The database (VM 2), dbserver, contains the information that is to be accessed by both VM 1 and VM 3. During the build process, dbserver creates a mysql user and password, and creates the database, contestants. All privileges are given to dbserver. After the installation of mysql-server, the setup-database.sql file script is run so that the table, myPass, is initialised, and then populated with values.

Adminserver (VM 3) displays the table, myPass. As new data is sent to the database from VM 1 (webserver), the adminserver webpage updates accordingly.

The initial build size for the ubuntu/xenial vagrant box requires approximately 280 MB. The webserver and adminserver use an additional 20.8 MB of space each, and the dbserver requires approximately 158MB of disk space. The subsequent builds should not require any extra MB as all the required installations for each machine have been installed already.

How to build (unattended)
In order to run our application, the following steps will guide you through how to get started. NOTE: the build process is completely unattended - there is no user interaction required when step 3 is executed.

1. Go to the github repository, https://github.com/bbosecke/Asgn1 .
2. Clone the repository into a new directory, “Asgn1”, on your desktop.
3. Once successfully cloned, using terminal, cd into “Asgn1” and run the command, “vagrant up”.
4. Once vagrant has initialized the three virtual machines, you can access any 3 of the machines by running the command, “vagrant ssh x” (x = machine name, e.g. dbserver).
    a. This allows you to access files or the database within the virtual machine.

If you wish to enter the database and query within the virtual machine dbserver, first you need to paste the following command into the terminal, “export MYSQL_PWD='insecure_mysqlroot_pw' “. Then you can login to mysql by using “mysql -u root”, which now gives you access to the database. The name of our database is “contestants” and the skier’s information is stored inside of the table, ’myPass’.

The initial download time from ‘vagrant up’ takes approximately five minutes and 20 seconds. Subsequent builds using ‘vagrant up --provision’ take approximately 30 seconds.

How should a user use the application?
Any user wishing to create an account for BroMo mountain should fill out the appropriate form on the webpage, ‘http://192.168.2.11/index.php’. The form requires you to create your own username, then enter your name and your preferred sport between skiing and snowboarding. Once the appropriate information is entered, the user hits “submit”. The data is then sent to the database and stored in the myPass table, inside of the contestants database. The user's information can now be accessed by the admin.

The webpage, ‘http://192.168.2.13/admin.php’ displays the information stored in the myPass table.

For the convenience of the marker for this assignment, we have included a link between the webserver website and the adminserver website to direct you between the two pages. These would not be displayed when our website is deployed.

There is also an error with the PHP form in the webserver website - whenever the page is reloaded, the form is submitted with blank values automatically. We figured this error was irrelevant to the assignment specifications and ignored it because we are not proficient in PHP. Because of this, there is a line in the database that is populated with no values, except a ‘0’ in the runs column.

Future modifications another developer could make

Our application has some modifications that would be useful when implemented. After making changes to files, depending on which files they are depends on the rebuild process.

- If the files edited are .php files, then no vagrant up is needed, just a refresh of the webpage 192.168.2.11/index.php or 192.168.2.13/admin.php.
- If any changes are made to the database, then a “vagrant destroy” command followed by a “vagrant up” will allow the database changes to be made.
- If any changes are made within the ‘provision’ section of the vagrant files, then the following command allows a quick reboot that implements the changes, “vagrant up --provision”.
- If any changes are made within the vagrant file elsewhere, then a “vagrant destroy” followed by a “vagrant up” must be run. This is unlikely however, as our 3 machines are already installed with the required installations for our website and database to be run, changed and accessed appropriately.

Modifications that another developer can make to our application are...

1. On the webserver website page, it would be helpful for users to be able to access the number of runs that they have completed. In the webserver web page, a search bar could be placed, so that you can type a username and a query would be sent to the database to display the number of runs that the user has completed. This would require a query that matches the username entered with the database, and if there is a match, to display the number of runs completed from the row that matched with the username.
2. Another change a developer could make is to give the administrator access to the database to update the run count of a skier/snowboarder manually, or to delete a user if they have broken the rules in any way.

References

David Eyers, COSC349 Lecturer
https://altitude.otago.ac.nz/cosc349/vagrant-multivm
Link for the basis of our PHP code:
https://www.tutorialrepublic.com/php-tutorial/php-mysql-insert-query.php?fbclid=IwAR0xFHfvNd7Ja4UJmoIQadsYfK7Ce2UpkOWjmrIPv-irMRkB7SjrgWo10FA