# COSC349 - Assignment 1

Mars Reservation Booking Application

Riya Alagh (8878519) & Mike Cui (9289732)

*Clone Link: https://github.com/alari453/booking-application*
*Commit ID: `30bc066`*

**INTRODUCTION:**

We have chosen to create an application booking system that utilizes three virtual machines that interact with one another. The first VM hosts a booking form for a user's input, the second displays the current bookings, and the third VM runs a database.

**APPLICATION DESIGN:**

Our application contains three VM's - a clientserver, an adminserver and a dbserver, which are separated within our application to perform each of their tasks as efficiently as possible. The first VM runs a web page in which users can input their personal details to make a booking to the Mars Reservation. The second virtual machine takes the information from the database and displays it on the 'Current Bookings' web page/server. The third VM is the database server that stores the inputs provided by the user on the initial web page/from the initial VM. The two servers for the web do not interact directly.

A more in depth description of our VM's is as follows:

The first VM, clientserver, is the first seen when running our application. Clientserver is a webserver that displays a booking form for a user to enter their booking details - fName, lName, email, and the user's preferredTime. The clientserver VM uses PHP to communicate with the database running on the third VM. Our second VM, adminserver, hosts an additional web page that displays the current bookings and those newly inputted by the user. We do this by querying the database through PHP and output the result to that web page.

The final VM, dbserver, hosts the database and stores the inputs from the initial clientserver VM. When the booking form is submitted, the data from that is stored in the dbserver. The information in the database can then be queried on by the second VM discussed earlier. The web servers communicate with the database using a private network that is set up in the Vagrant file.

There is significant importance in having separate virtual machines because it means we can make changes individually and incrementally, i.e., we can make changes within one of the VM's without affecting another VM it may have communication with. Additionally, a huge benefit of separate VM's is the elasticity that it allows for. We can deploy multiple machines when dealing with a high load with ease. For example, if one of the virtual machines is in a corrupt state, only one VM is down. To solve this, we can then vagrant up that individual machine again, which ultimately will be faster as it is only downloading a third of the information/data.

**DOWNLOAD VOLUMES:**

When you initially run vagrant up in the terminal, the expected download volume for the virtual machine box (ubuntu-xenial) is approximately 250mb. Additionally, when fetching clientserver, vagrant fetched 15mb in 9s and overall took 15 seconds to load and boot

clientserver. adminserver fetched 19.5mb in 9s to load and boot. Both VM's also get 5,163kB of archined. In regards to dbserver, it initially fetches 19.5 mb in 9 seconds, but then needs a further 18.0 MB of archives, which it collects in 9 seconds also. That operation used an additional 158mb of disk space. Overall, to vagrant up, it takes roughly 3-5 minutes to build the application. Because this is now stored locally after that first build, any subsequent builds or new virtual boxes simply require the same VM box with the same ubuntu/xenial box.

**HOW TO RUN THE APPLICATION:**

Assuming the user has the latest version of Vagrant and VirtualBox, running this application is fairly easy. First, the repository will need to be downloaded or cloned. To clone the repository in the terminal, use the command:

https://github.com/alari453/booking-application.git

Once this is complete, you can then navigate through the terminal into the directory (folder) that the repository/Vagrant file is in. In the terminal, you can then use the following command to start the virtual machines:

```
vagrant up --provider virtualbox
```

When the virtual machines are running, you can then open the web pages hosted by the appropriate VM's by typing in the following into the url bar:

- Clientserver can be accessed via localhost:8080/

The initial website, hosted by webserver, shows a booking form where you can input your specific information for your booking. Once done, there is a 'submit' button which after being clicked indicates whether or not your data has been added successfully. You're then given two link options to either make another booking or view the current bookings in the database.

- adminserver can be accessed via localhost:8080/

The second web page, hosted by adminserver, is a simple table that displays the current bookings in the database. It shows you the results of the inserted data, as well as some preloaded bookings that have been added to the database when the VM is started.

**SUGGESTED MODIFICATIONS:**

We discussed various modifications that further developers could make to our application, a few suggestions we have are:

- A potential logging in/out system that allows users to look at their bookings more specifically. This could involve a querying/searching system that allows users to find their booking more directly.

- Allow a user to update their booking information.
- Allow a user to choose a time slot that has not already been picked by another user.

A modification that would be helpful for this booking application would be some sort of registration and then login/logout feature. This could be done by running an additional web server that displays some variation of a registration page before the current first clientserver. Making this addition would be fairly easy for another developer to adapt, however, an important note would be to include password hashing for any account that is stored in the database.

Another modification that a developer could adapt to this code would be to create prepared statements that allow a user to retrieve and update the information that is currently stored in the database. Right now, the php is set up with queries that show what is made in the database. The other suggestion would allow time slots to not overlap. This would involve having a query that displays to the user only available time spaces.

If changes are made to the Vagrant file or any of the provisioning shell scripts, you would have to reload the virtual machines by running the following command in the terminal:

```
vagrant reload --provision
```

However, if neither of these changes involves updating the Vagrant file or the provisional shell scripts, changes could be seen as soon as they're made by simply refreshing the web browser the sites are being viewed on.

**APPLICATION DEVELOPMENT:**

We developed our booking application over the course of two weeks. Mainly, we developed through a master branch, however, we made use of an additional 'testing' branch for uncertain changes that were being made. Due to technical issues on Riya's laptop, commits were made rather frequently in order to avoid any minor errors or loss of code. Following this, commit messages were written with both developers in mind so that they were easy to follow and understand. We also completed a lot of our work while together on a zoom call in order to further avoid issues caused by Riya's old laptop. Unlike a typical web application, less attention was given to design and security due to the specific nature of our assignment.

**DELAYED HAND-IN:**

There was a delay in our hand-in for this assignment due to both technical and unpredicted interruptions. A large factor for needing an extension was because of the interruptions caused by COVID-19 and having to work separately in a pair while being in lockdown. We overcame this by using zoom to work together on certain parts and to discuss what needed to be done separately. However, we also had some technical

issues; Riya's 5-6 year old laptop crashed numerous times while running VirtualBox and any other applications which made it really hard to test and make sure things were running smoothly. This would have been solved by her purchase of a new laptop, however, the MacBook Pro's new M1 CPU apparently is not yet supported by VirtualBox. Unfortunately, this then caused further delays.

**REFERENCES:**

*David Eyers* - vagrant-multivm https://altitude.otago.ac.nz/cosc349/vagrant-multivm
*Hashicorp* - https://www.vagrantup.com/docs/cli