Assignment 1 report

Brit Duncan 6805461
David Epps 3017072

Application:

Our application was designed to be very simple. It converts your local time to a chosen timezone from a database of timezones. We chose to make our application as simple as possible to focus on demonstrating our aptitude at automated provisioning of VM's and running an application across multiple.

Division of Labour:

The application runs across three Vm's a Web Server, a Database server, and a Conversion server

The Web server acts as the front end to our application; it displays a list of available time zones to choose from and has a dropdown menu which allows you to select one. It pulls this list from the Database server. After a given instruction is chosen it passes that off to the Conversion server. The Web server then displays the current time and date in that time zone

The Database server stores all the available time zones and their offsets in a sql database

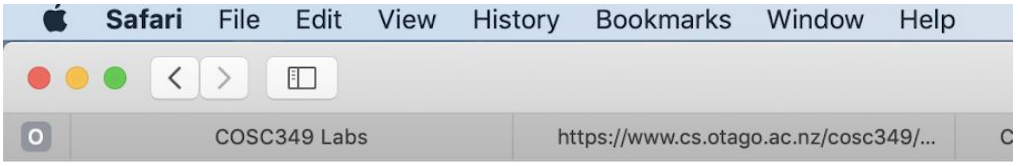
The Conversion server takes the System time (by default the same as the Host machines) and given a specific timezone by the webserver calculates and returns a new time and date for printing by the Web server.

Dividing the application thusly means that all conversion logic is maintained separately to the IO and the Data on timezones meaning each could be changed to suit specific needs and keeps logic discrete, it also encapsulates the Database server as not immediately accessible from the outside by putting a layer behind the Webserver, By putting it behind the web server with only specific queries allowed it protects from common attacks such as an sql injection attack

How to Run and Expected Downloads:

To Run simply clone the repository and from within that directory run
Vagrant up --provider virtualbox
Assuming you select virtual box as your provider and have a Ubuntu xenial64 box your total downloads should sit in the order of 20MB for each Vm For ~60MB total else you will need an additional ~ 270MB for a Ubuntu xenial64 box
The VM's should build completely unsupervised.You can then access the Application from the webserver hosted locally on 8080.You can navigate to it either through

http://192.168.2.11/index.php or http://localhost:8080



You can convert to the following timezones:

Timezone IDs	Timezone Offset
Pacific/Niue	-11.00
Pacific/Honolulu	-10.00
Pacific/Gambier	-9.00
Pacific/Pitcairn	-8.00
America/Phoenix	-7.00
Pacific/Galapagos	-6.00
America/Chicago	-5.00
America/New_York	-4.00
America/Argentina/Buenos_Aires	-3.00
Atlantic/South_Georgia	-2.00
Atlantic/Cape_Verde	-1.00
Iceland	0.00
Africa/Kinshasa	1.00
Africa/Johannesburg	2.00
Asia/Baghdad	3.00
Asia/Dubai	4.00
Indian/Maldives	5.00
Asia/Omsk	6.00
Asia/Krasnoyarsk	7.00
Australia/Perth	8.00
Asia/Tokyo	9.00
Pacific/Guam	10.00
Pacific/Noumea	11.00
Pacific/Auckland	12.00
Pacific/Apia	13.00
Pacific/Kiritimati	14.00

Choose a timezone to convert Pacific/Niue Convert

Development From Here:

The first obvious change would be to implement daylight savings time in each of the timezones offered where applicable there are a few ways to do this. I would suggest that the easiest way to do this would be the database have 2 databases 1 (the current one which is shown to users) and a second one with the daylight savings and their segments of the year. Then when the conserver does its conversion it can poll the database to find out if the date and zone it has converted to is in some kind of daylight savings and then modify its output accordingly before sending the output to the web server

A further improvement would be the inclusion of every timezone not just one timezone from every offset, obviously this is hundreds of timezones so representing this on the web server would be a challenge, we would have to implement a query bar that could autocomplete based upon rows in our database. Next we would have to import some external database of timezones to avoid hardcoding the many hundreds of timezones on the database server. Finally we would need to change the logic of the conserver such that switched correctly given the new wider database.

Our Development Process:

Presently our repository only has 2 commits: the initial generation and the one where we put up all our work (this will be the third). In Hindsight that wasn't a very transparent way to develop this application as it does not demonstrate a change over time so that will be outlined here. Because the application itself is very simple our development process consisted of a few hours in front of a machine working together across a couple of days wherein most of the work consisted of trying to get the different VM's to talk to each other, a result of which is that we didn't commit early drafts or work in progress to git as we went. Our first step was to create three separate VM's and dole out tasks to each as seemed appropriate. We used the Vagrant file in lab 6 as a jumping off point and added a third VM and began to do some small customisations for each. Hosting something small locally was an easy first step so we had a small php script print some text locally on the web server. The Database of timezones was a fairly small and simple edition and easy to host on the Database server so a small sql database was quickly whipped up. The conserver was next and some time was spent making a php combobox that took in some argument from the webserver (we would sort out how to get this across later) and spat out a date time string. We returned to the Webserver and had it knit the whole thing together. It now pulled the DB from the database server and could pass off an argument to the conserver. Once all of this was working neatly a back button was added to the conserver to make it more functional. Given the simplicity of the Application debugging consisted largely of trial and error trying to stick together each component rather than regimented testing which better suits more complex systems with more variables at play, rather than simply trying to make it work.