Ronald McDonald House Service Project

Developer's Volunteer Calendar: Windows OS

EPICS Spring 2017

OVERVIEW

This document includes instructables for running and developing the Ronald McDonald service calendar on a Microsoft Windows host environment. The calendar and web server is tested via the command line. The instructions herein are used to download and install a guest Ubuntu VM on a host Windows machine. Virtualbox, a virtual machine manager, is installed and uses a Vagrantfile to quickly and easily get the volunteer calendar up and running for rapid development and testing.

GOALS

1. Create Ubuntu VM using Virtualbox and a Vagrantfile
2. Download & configure PuTTY with Ubuntu VM
3. Logging into Ubuntu VM
4. Setup GitHub repository for Ronald McDonald service calendar development files
5. Installing Node.js & NPM
6. Using NPM to install Grunt & Bower
7. View the Calendar in Internet Browser

Create Ubuntu VM

Ubuntu is a complete Linux OS. Running the project through an Ubuntu server on your host WIndows machine makes running the command prompts for this project easier. Virtualbox is a virtual machine (VM) manager and is used to power up the Ubuntu server. Once Virtualbox is downloaded and a Vagrantfile is placed in a directory, Putty is used to SSH to the Ubuntu server, providing a shell to easily use the Linux OS, .

Download & Install Virtualbox, Vagrant

At the time of this writing, the Vagrant download provided on the Ronald McDonald House EPICS website works best with a Virtualbox version <5.1. The downloads for Vagrant and Virtualbox can

be downloaded from the EPICS website and installed after unzipping the compressed folder. Newer versions may be available online from the links below, however, the newer versions have not been tested. More information about Virtualbox and Vagrant are also available from the links below. Follow the installation prompts after unzipping compressed folder.

Install VirtualBox from https://www.virtualbox.org/

Install Vagrant from https://www.vagrantup.com/

Create Vagrant Directory

A Vagrant directory is created to house a Vagrantfile that is used to quickly and easily construct an Ubuntu development environment specifically for this project. The Vagrantfile can easily be modified to run recurring command line prompts so that the time to get the Ubuntu VM up and running is minimized. Begin by downloading the Vagrant compressed zip folder from the Ronald McDonald House EPICS website. Then use the following steps to place the Vagrant folder on your C drive

1. Open File Explorer
2. In left hand menu, left click Downloads
3. Right click the compressed, downloaded RonDon Vagrant folder
4. Select Extract All from the pop-up menu
5. Hit the Browse button and select Local Disk (C:) from the left hand menu in the file explorer
6. Click Extract

Create Ubuntu VM

Run the Virtualbox application through the WIndows desktop or start menu. Notice the left hand side of the application. This is where all VM's are stored for easy startup. Right now, the menu should be empty. We must create the Ubuntu VM. Use the following steps to easily create the Ubuntu VM.

Through Windows startup menu, open a command prompt: powershell or cmd.exe

Change directories to where Vagrant directory (folder) was saved. If saved to Local Disk (C:), type

```
cd C:\Vagrant
```

To create Ubuntu VM, type

```
vagrant up
```

Now check the Virtualbox application. After a few seconds to a minute, a menu item should appear in the left-hand window that says something such as “Vagrant Default blah blah blah”. This is your Ubuntu VM.

On the upper right-hand side of the application is a Preview window. Words should begin scrolling down the preview. This is startup info about the VM. To open a bigger, interactive preview window, right click the “Vagrant Default blah blah blah” item in the left hand menu and click show.

Subsequent Ubuntu VM Startups & Shutdowns

Once initially installed, the Ubuntu VM can be started quickly by simply

1. Opening the Virtualbox application
2. Left click the “Vagrant Default blah blah blah” item in the left hand menu
3. Click

To shutdown the Ubuntu VM

1. Opening the Virtualbox application
2. Right click the “Vagrant Default blah blah blah” item in the left hand menu
3. Select Close → Power Off from the pop up menu

Download & Configure PuTTY

Once the Ubuntu VM is running in Virtualbox, PuTTY is used as a client-server interface in order to provide greater functionality, such as cut and paste. PuTTY uses SSH, a client-server networking protocol, to connect to the Ubuntu VM in order to provide a (slightly) better developing experience. Follow these simple steps to connect to Ubuntu via PuTTY

Download & Install PuTTY for Windows (32- or 64-bit) from the link below
http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Ensure Ubuntu VM is running via Virtualbox

Open PuTTY via startup menu or desktop. In the fields provided type

Host Name: **localhost**

Port: **2222**

Since this is your development environment, you will probably be logging in frequently. So, you can save the login. Under Load, save or delete a stored session

Saved Sessions: **Ubuntu VM**

Click: **Save**

In the future, to connect to Ubuntu VM (server) simply click Ubuntu VM in the menu box and click Open, which opens a secure shell (SSH) similar to Thomas.

Logging into Ubuntu VM

Logging into the Ubuntu VM can be done by using the following information

Username: **`ubuntu`**

Password: **`96a418904595405b0e052e6d`**

Pasting is done in the shell by right clicking. When pasting the password you will not be able to see it but it's there. Simply right click and press enter. Once logged in the password can be changed to a more memorable one from the command line using

```
(sudo) passwd
```

Setup GitHub Repository

The Ubuntu VM is unable to access files on your host machines (Windows) hard drive. For this reason, to easily access, edit and save the Ron Don Calendar development files, a GitHub repository is created. Ron Don development files can be saved within the Ubuntu VM, however, should the VM crash for whatever reason, say during updating, files can quickly be downloaded from GitHub and reinstated after recreating the Ubuntu VM. Every Butler student has a GitHub account and can easily login using university credentials.

Talk to Prof. Linos to have yourself added to DaRonDon repo on GitHub as a contributor. This enables code for the project to be saved to one location. Use the following steps via the command line in the VM

Install GitHub in Ubuntu VM

```
sudo apt-get update && apt-get install -y git
```

Clone repo in your Ubuntu VM to bring files from GitHub into VM

```
git clone git@github.butler.edu:cljones1/DaRonDon.git
```

Check to make sure new folder with reponame was created

`ls`

You should see DaRonDon directory listed. Change directories to access DaRonDon calendar files

`cd DaRonDon`

Now the development files are in your VM. As you make changes to the files, after each session, be sure to back them up to GitHub in case, heaven forbid, WIndows or Ubuntu crashes unexpectedly.

Prepare any changed files

`git commit -a -m "update description"`

Send files to GitHub

`git push origin master`

The **`git commit`** command automatically knows which files have been changed that belong to the Ron Don repo. Be sure to add an **`"update description"`** else **`git commit`** will throw an exception.

Installing Node.js, NPM, & MongoDB

Node.js is an open source JavaScript (JS) runtime environment used for creating web applications. Node Package Manager (NPM) makes it easy for Javascript developers to share the code that they've created to solve particular problems, and for other developers to reuse that code in their own applications. Both of these need downloaded in order to properly run the volunteer service calendar.

MongoDB is a NoSQL data warehousing program that is used to store the data gathered from the front-end web calendar application. Mainly, who is going to volunteer and when. These programs can be downloaded in one command line.

`sudo apt install mongodb nodejs npm`

You can check to see if the installation was successful by checking the version, for example,

`npm --version`

`node --version`

A directory must be made for MongoDB to save the data

```
mkdir -p mongodb
```

```
cp -R -n mongodb-osx-x86_64-3.4.2/ mongodb
```

Now the database is installed and it can be opened to access the command prompt for mongo to see databases, collections and query the database, search and delete. The mongo process can be started by

```
sudo service mongodb start
```

```
mongo
```

You can always check to see if the mongodb (or any) process is running by using the following commands

```
ps -eaf | grep mongod
```

```
sudo service mongodb status
```

If for any reason the database must be restarted

```
sudo service mongodb restart
```

Use NPM to Install Grunt & Bower

Grunt is defined as a task runner. Tasks in Grunt are analogous to plugins in Visual Studio; they unable additional functianlity. Grunt is used to run common tasks such as running and testing an application. The tasks run by Grunt are user defined within a dedicated Gunt file. As this is an existing project, the Grunt file is already setup and can be viewed from within DaRonDon directory

```
more gruntFile.js
```

The dependencies of the project are defined within the package.json file, along with additional project information. The contents of the file can be viewed via

```
more package.json
```

In order to use Grunt and get the project running on a local server, Grunt and the Grunt command line interface must be downloaded

```
sudo npm install -g grunt
```

```
sudo npm install -g grunt-cli
```

The Grunt file for this project defines four major tasks

1. JSHint : a linting task that enables debugging of JavaScript files http://jshint.com/
2. JSCS (JavaScript Standard Style) : a task that ensures JavaScript files conform to the JavaScript standard http://standardjs.com/
3. Uglify : a task that minifies and compresses JavaScript files https://www.npmjs.com/package/uglify-js
4. Karma : a testing task that enables the user to test an application on various web browsers https://docs.angularjs.org/guide/unit-testing

The Grunt file for the calendar app (gruntFile.js) has a standard function architecture and code structure. To learn more about Grunt, running tasks, and understanding the code, check out the following links

Building a JavaScript Library w/ Grunt.js : http://meri-stuff.blogspot.com/2013/06/building-javascript-library-with-gruntjs.html

Getting Started : https://gruntjs.com/getting-started

Creating Tasks : https://gruntjs.com/creating-tasks

Configuring Tasks : https://gruntjs.com/configuring-tasks

Bower is a web browser package manager that manages HTML, CSS, images, and API's such as JQuery. Thus, bower works more with the front end of a web application. Bower works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for. Bower is installed by using NPM globally

```
sudo npm install -g bower
```

To learn more about bower, check out the following links

https://bower.io/

https://code.tutsplus.com/tutorials/meet-bower-a-package-manager-for-the-web--net-27774

Run the Calendar in Web Browser

Once bower and grunt are installed, the only thing left to do is allow bower to download and manage all front end packages for the application.

With the virtual machine open, SSH into the Ubuntu server using PuTTY, and navigate to the git repository folder

```
cd reponame
```

Get the back end of the application up and running using NPM

```
npm install
```

Node.js is used to get the local server running

```
node .
```

You will know the local server is running once the following message is displayed in the shell

```
Ronald McDonald House Volunteer Service
```

The back end and front end portions of the application require separate shells running concurrently in order to get the volunteer calendar running in a browser. You can duplicate a shell by right clicking on the PuTTy cpu icon in the upper left corner of the shell and selecting Duplicate Session from the drop down menu. Sign into this server as well and navigate to the git repository folder again

```
cd reponame
```

Bower is used as the package manager for the front end of the application.

```
bower install
```

Grunt is then used to minify the .js files used for the app. Minifying simply removes unnecessary semicolons and so on from code; making underlying binary files smaller for more efficient running.

```
grunt minify
```

An error message may be thrown

```
Loading "grunt-karma.js" tasks...ERROR

>> TypeError: Cannot read property 'prototype' of undefined

Running "uglify:build" (uglify) task

>> Destination calendar.min.js not written because src files
were empty.
```

```
>> No files created.

Done, without errors.
```

We can forge ahead however. Lastly, Grunt is used to get the local server for the front end running using

```
grunt serve
```

Another error maybe thrown

```
Loading "grunt-karma.js" tasks...ERROR

>> TypeError: Cannot read property 'prototype' of undefined
```

Despite the error, the server should be up and running and can be verified

```
Running "connect:server" (connect) task

Waiting forever...

Started connect web server on http://localhost:8000
```

Now navigate to your favorite browser and type

```
localhost:8080
```

And voila! The calendar is up and running! Back in the shell, the front and back end code can be halted by pressing Ctrl C.