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 virtual machine 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 quicker development and testing.

GOALS

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

Enabling Virtualization

Xenial is a 64-bit Ubuntu box. The CPU on some machines must be configured through the BIOS to enable virtualization of a 64-bit virtual machine. To check or enable the virtual extension for AMD -V or Intel VT-X, check out the following website, or Google AMD-V, or Intel VT-X

https://docs.fedoraproject.org/en-US/Fedora/13/html/Virtualization_Guide/sect-Virtualization-Troubleshooting-Enabling_Intel_VT_and_AMD_V_virtualization_hardware_extensions_in_BIOS.html

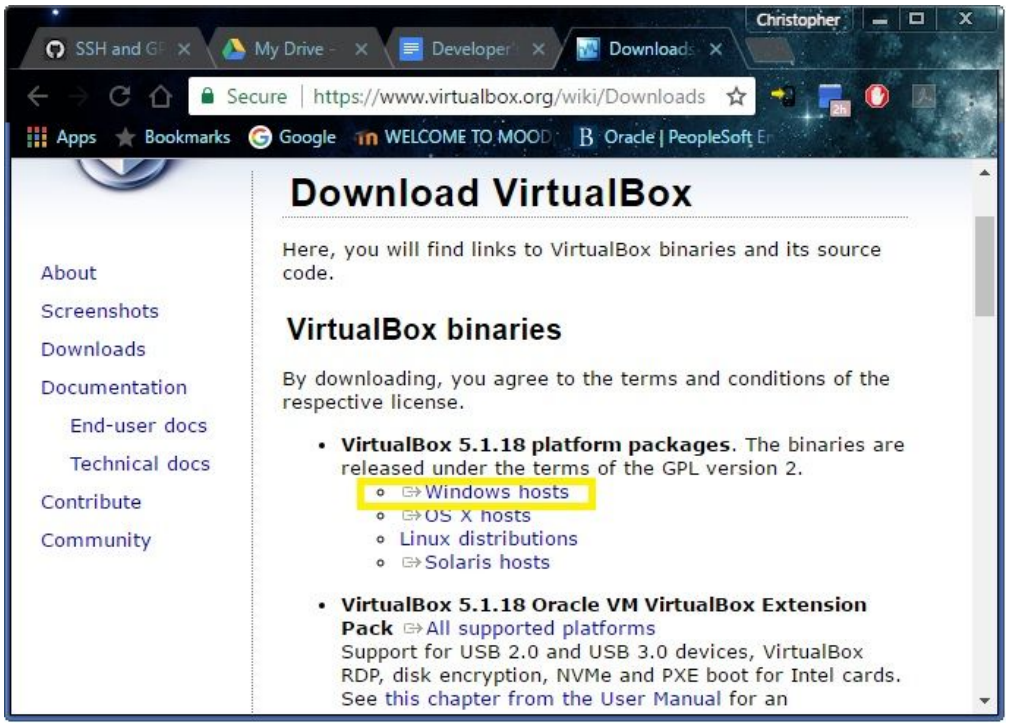
Create Ubuntu VM

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, used to power up the Ubuntu server. A Vagrantfile is placed in a directory on the C drive, and Putty is used to SSH to the Ubuntu server, providing a shell to easily use the Linux OS, .

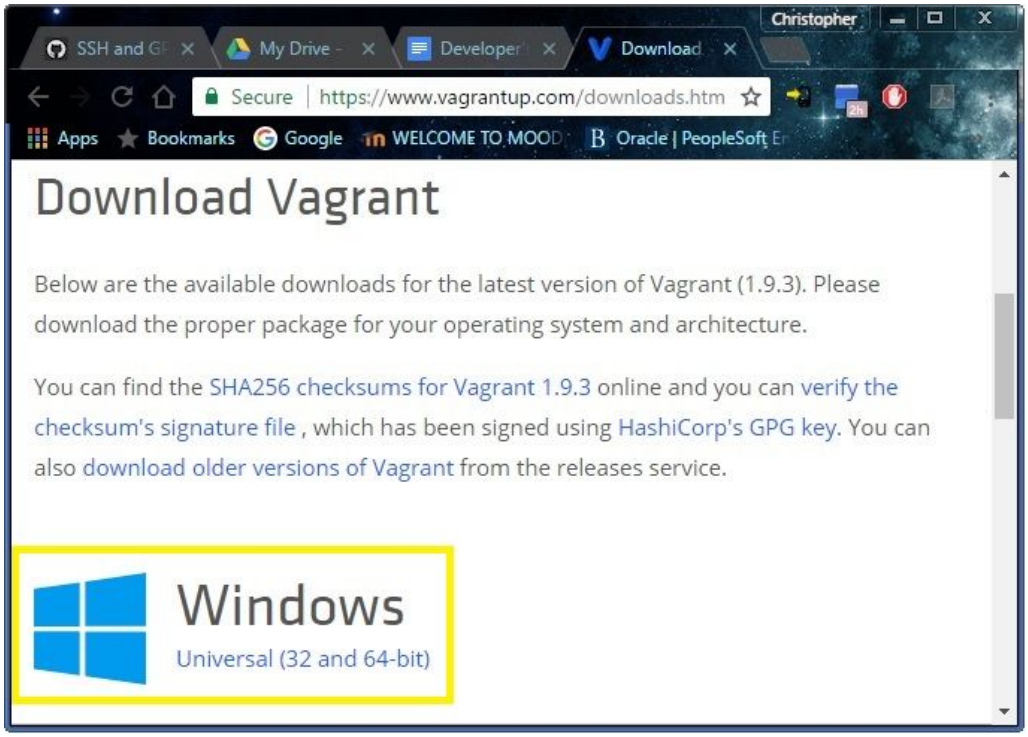
Download & Install Virtualbox, Vagrant

The downloads for Vagrant and Virtualbox can be downloaded online from the links below. More information about Virtualbox and Vagrant are also available from the links below.

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



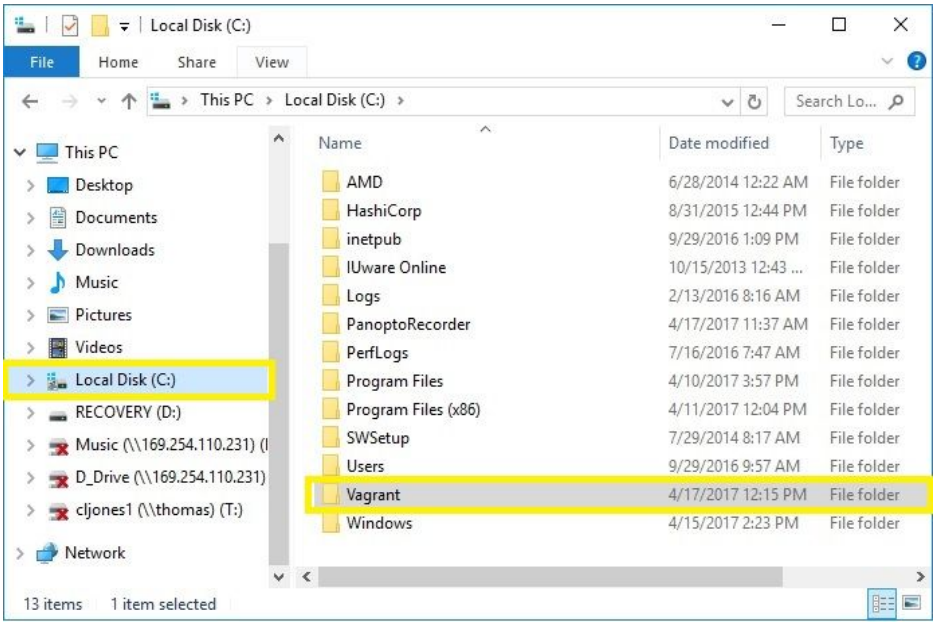
Install Vagrant from https://www.vagrantup.com/



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. Begin by downloading the Vagrant compressed zip folder from the Spring 2017 Ronald McDonald House EPICS website. (May need to login). The folder also has PDF's to help with setting up Vagrant, VirtualBox, and GitHub help. 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

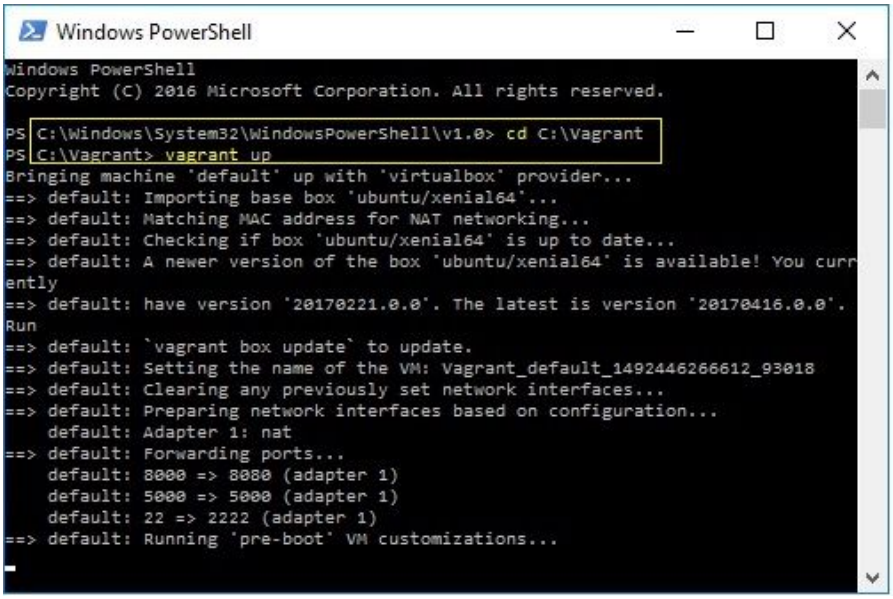
Run the Virtualbox application through the Windows desktop or start menu. Notice the left hand side of the application. This is where all VM's are stored for easy startup. Right now, the menu should be empty. We must create the Ubuntu VM. Use the following steps to easily create the Ubuntu VM.

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

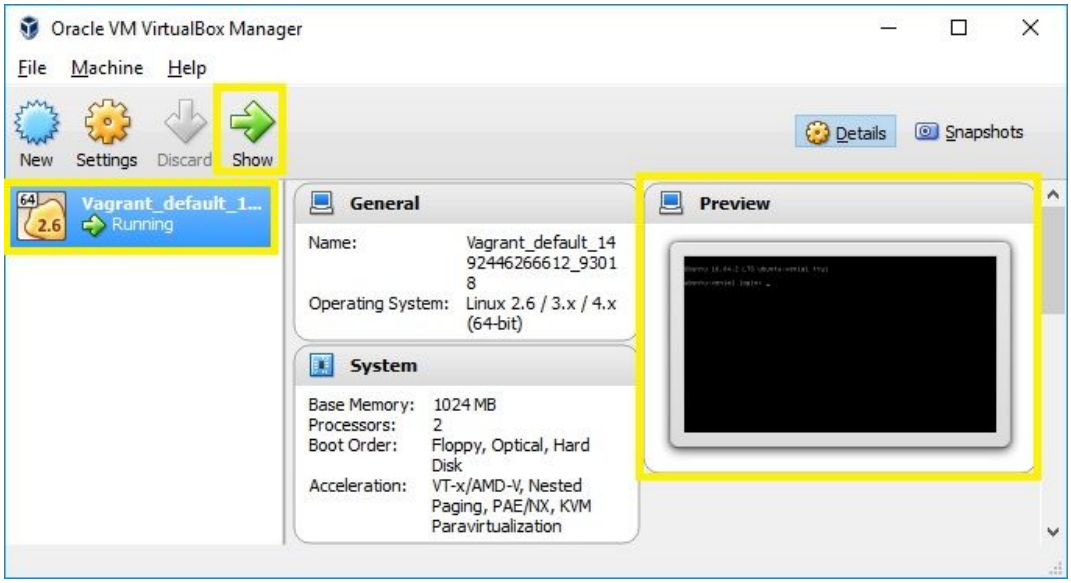
```
cd C:\Vagrant
```

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, left 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 Run

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

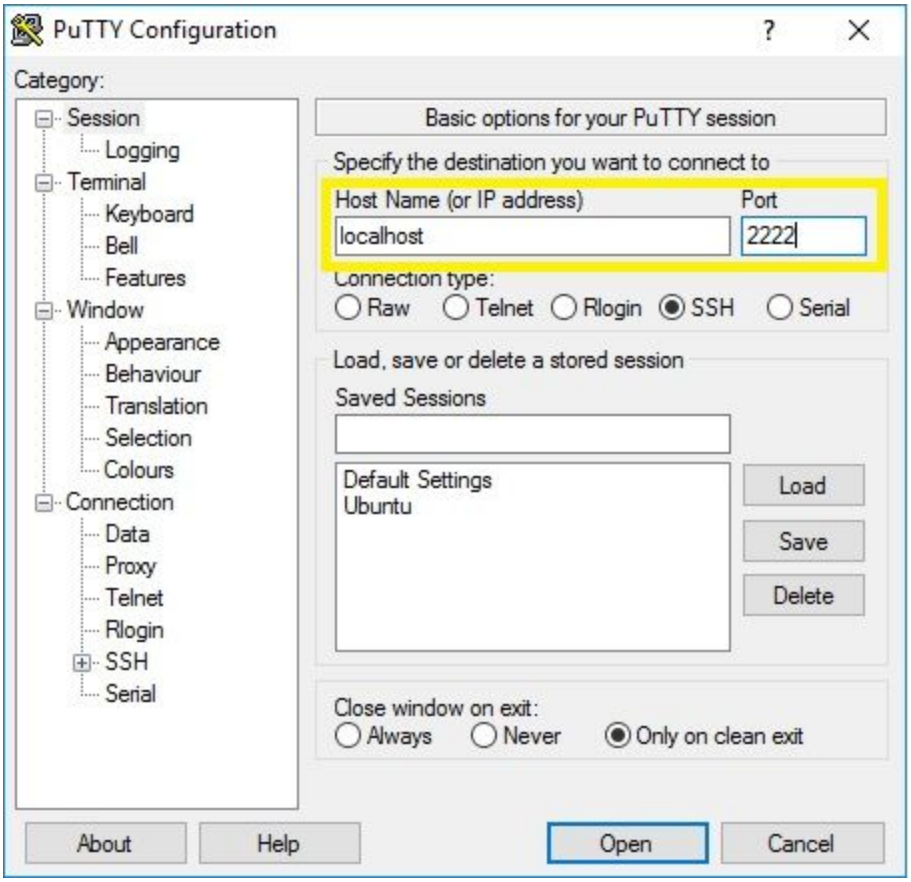
Once the Ubuntu VM is running in Virtualbox, PuTTY is used as a client-server interface in order to provide greater functionality, such as cut and paste. PuTTY uses SSH, a client-server networking protocol, to connect to the Ubuntu VM in order to provide a (slightly) better developing experience. Follow 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. If this password does not work for you, In

your home directory on windows you should find a folder named “.vagrant.d”. If you descend that so that you arrive at something like:

C:\users\NAME\.vagrant.d\boxes\ubunut-VAGRANTSLASH-xenial64\20170222.0\virtualbox

The path may be slightly different (in particular the box version that looks like a date) but you should only have one. In there you’ll find another Vagrantfile which contains the username and password.

Once logged in the password can be changed to a more memorable one from the command line using

```
(sudo) passwd
```

Installing Node.js, NPM

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.

```
sudo apt-get install -y python-software-properties

curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -

sudo apt-get install nodejs
```

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

```
npm --version

nodejs --version
```

Install MongoDB

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, along with some Python libraries. Via the command line

```
sudo apt-get install -y mongodb
```

A directory must be made for MongoDB to save the data

mkdir -p 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

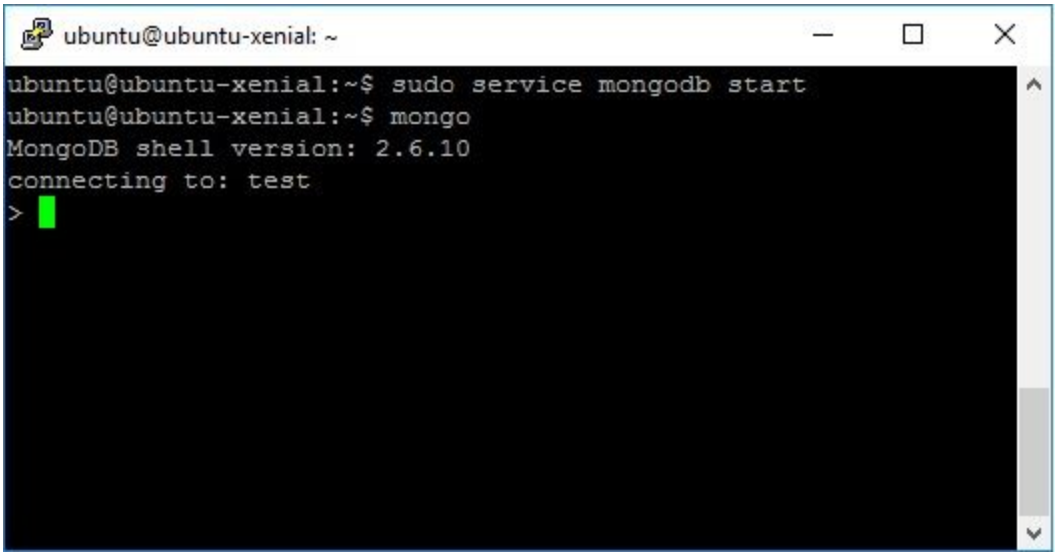
ps -eaf | grep mongod

sudo service mongodb status

If for any reason the database must be restarted

sudo service mongodb restart

The database can be closed at any time by using control C



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.

Download git via the command line

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

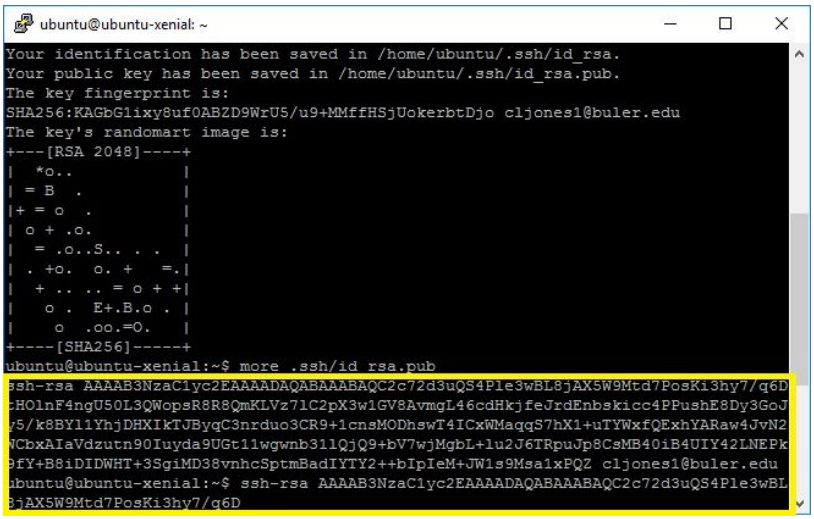
Talk to Prof. Linos to find the latest RonDon repository and code. It should be made available on the EPICS website. Log into GitHub using your moodle information. Upload this code to a new repository, and go to the settings tab. On the left hand side go to SSH & GPG keys. Click the add a new SSH key button. Give the key a name.

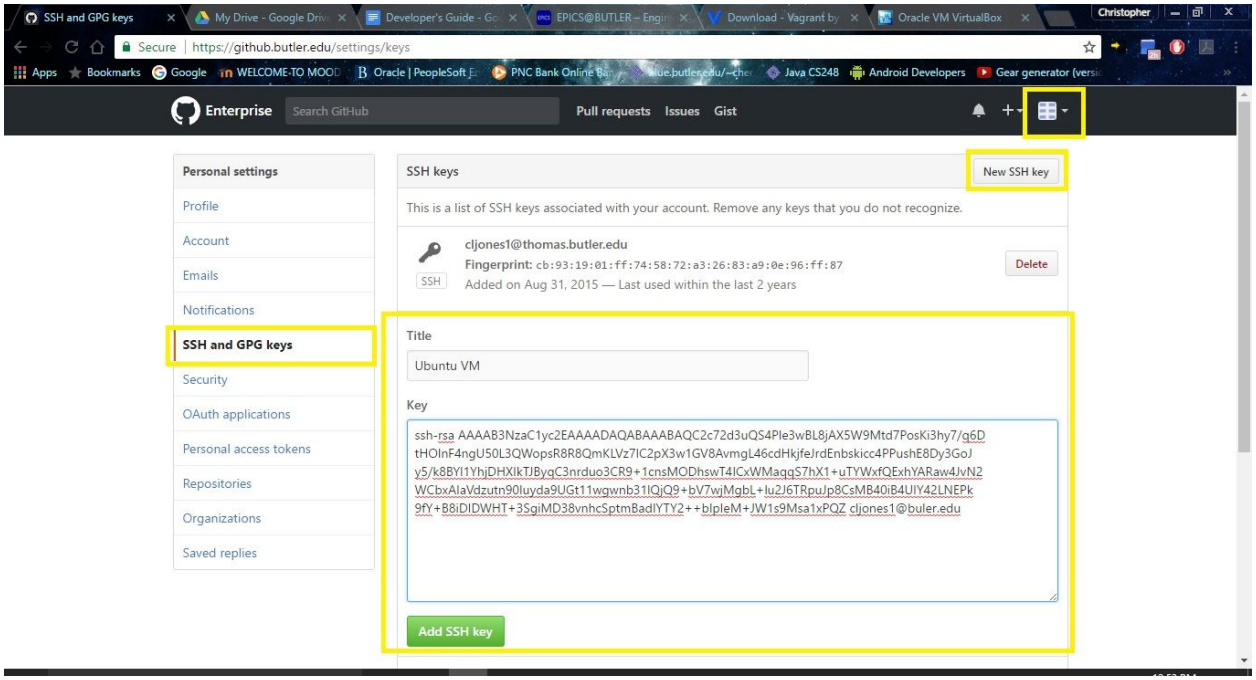
Back in PuTTY/Linux command line, generate an SSH key (hit enter 3x to accept defaults) and display the key

```
ssh-keygen -t rsa -C "you@butler.edu"
```

```
more .ssh/id_rsa.pub
```

Copy (highlight in terminal) and paste the key into GitHub and save





GitHub is downloaded automatically via the VagrantFile the first time the Ubuntu VM is started. So, to **c**lone the repo in your Ubuntu VM to bring files from GitHub into VM

```
git clone git@github.butler.edu:username/REPONAME.git
```

Check to make sure new folder with reponame was created

```
ls
```

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

```
cd REPONAME
```

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.

Using Tree

Linux does not have a file explorer. You can see the contents of a folder by using the **`ls`** command. For this project it is helpful to see more than just the files in the immediate subdirectories within the GitHub repository. The project paths are divided into two parts: one for the front-end application (the webpage) and one for the backend application (monog database). To see the path structure, in the GitHub folder, in the command line type

```
cd REPONAME

ls

cd src/main

tree
```

You'll see the file structure printed on the screen as below. The bower.json, gruntFile.js, and package.json files in the initial project directory are files you should get familiar with; they are used by bower, grunt, and npm, respectively, to configure the application. The angularjs folder houses the front end directories and files, while the nodejs folder houses the backend directories and files.

For more information on tree, and various command line options, review the following website

https://lintut.com/use-tree-command-in-linux/

```
ubuntu@ubuntu-xenial: ~/DaRonDon/src/main
ubuntu@ubuntu-xenial:~/DaRonDon$ ls
bower.json     gruntFile.js  node_modules  README.md
Data (2).json  LICENSE       package.json  src
ubuntu@ubuntu-xenial:~/DaRonDon$ cd src/main
ubuntu@ubuntu-xenial:~/DaRonDon/src/main$ tree
.
├── angularjs
│   └── com.ronaldmcdonaldhouse.volunteerservice
│       └── static
│           ├── app.js
│           ├── css
│           │   ├── calendar.css
│           │   ├── contact.css
│           │   └── home.css
│           ├── index.html
│           ├── js
│           │   ├── controller
│           │   │   ├── CalendarController.js
│           │   │   ├── ContactController.js
│           │   │   └── HomeController.js
│           │   └── library
│           │       └── calendar.js
│           └── view
│               ├── calendar.html
│               ├── contact.html
│               └── home.html
└── nodejs
    └── com.ronaldmcdonaldhouse.volunteerservice
        ├── app.js
        ├── controller
        │   ├── CalendarController.js
        │   ├── EmailController.js
        │   └── ServiceController.js
        ├── model
        │   ├── Calendar.js
        │   └── Service.js
        ├── package.json
        └── server
            └── www.js

13 directories, 20 files
ubuntu@ubuntu-xenial:~/DaRonDon/src/main$
```

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 functionality. 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 GruntFile. As this is an existing project, the Grunt file is already setup and can be viewed from within the REPONAME directory

```
cd REPONAME
```

```
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 (CLI) must be downloaded globally from the /home directory

```
cd
```

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

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

I have also had to install Grunt inside the REPONAME directory as well in order to get the project up.

```
cd REPONAME
```

```
npm install grunt
```

```
npm install 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 GruntFile 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 globally in the /home directory by using NPM

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

I have also had to install Grunt inside the REPONAME directory as well in order to get the project up.

```
npm install 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
```

Check to make sure that you have permission to access a directory named node_modules. If not, you'll get an error saying run as admin. So assign yourself permission if need be. You will know if you have permission by inspecting the read write permissions as in the picture below. Run

```
ls -l
```

If **`root.root`** is next to the **`node_modules`** folder instead of **`ubuntu.ubuntu,`** (see pic below) change the read write privileges

sudo chown -R ubuntu.ubuntu node_modules

ls -l

Double check the change. Get the back end of the application up and running using NPM to install the project dependencies within the package.json

npm install

```
ubuntu@ubuntu-xenial: ~/DaRonDon
npm ERR! Error: EACCES: permission denied, access '/home/ubuntu/DaRonDon/node_mo
les'
npm ERR!  { Error: EACCES: permission denied, access '/home/ubuntu/DaRonDon/node
odules'
npm ERR!   errno: -13,
npm ERR!   code: 'EACCES',
npm ERR!   syscall: 'access',
npm ERR!   path: '/home/ubuntu/DaRonDon/node_modules' }
npm ERR!
npm ERR! Please try running this command again as root/Administrator.

npm ERR! Please include the following file with any support request:
npm ERR!     /home/ubuntu/.npm/_logs/2017-04-19T14_54_25_127Z-debug.log
ubuntu@ubuntu-xenial:~/DaRonDon$ ls -l
total 212
-rwxrwxr-x  1 ubuntu ubuntu    947 Apr 19 14:48 bower.json
-rw-rw-r--  1 ubuntu ubuntu 187300 Apr 19 14:48 Data (2).json
-rwxrwxr-x  1 ubuntu ubuntu   2565 Apr 19 14:48 gruntFile.js
-rwxrwxr-x  1 ubuntu ubuntu   1128 Apr 19 14:48 LICENSE
drwxr-xr-x 44 root   root     4096 Apr 19 14:50 node_modules
-rwxrwxr-x  1 ubuntu ubuntu   1105 Apr 19 14:48 package.json
-rwxrwxr-x  1 ubuntu ubuntu    415 Apr 19 14:48 README.md
drwxrwxr-x  3 ubuntu ubuntu   4096 Apr 19 14:48 src
ubuntu@ubuntu-xenial:~/DaRonDon$
```

There is another package.json file buried in the project path whose dependencies must also be installed. To do so, navigate via the command line

cd src/main/nodejs

cd c

Then hit the tab key to autocomplete the directory name (it's long. You can see it by typing **ls** via the command line). The path should be in the form of

~/src/main/nodejs/com.ronaldmcdonaldhouse.volunteerservice

Install the dependencies in this folder as well

`npm install`

Node.js is used to get the local server running from within the server folder

`cd server`

`node www.js`

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

`Ronald McDonald House Volunteer Service`

```
ubuntu@ubuntu-xenial: ~/DaRonDon/src/main/nodejs/com.ronaldmcdonaldhouse.volunt...
│   ├── muri@1.2.1
│   ├── regexp-clone@0.0.1
│   └── sliced@1.0.1
├── multer@0.1.8
│   ├── busboy@0.2.14
│   │   ├── dicer@0.2.5
│   │   │   ├── readable-stream@1.1.14
│   │   │   │   └── isarray@0.0.1
│   │   │   └── streamsearch@0.1.2
│   │   └── readable-stream@1.1.14
│   │       └── isarray@0.0.1
│   ├── mkdirp@0.3.5
│   ├── qs@1.2.2
│   └── type-is@1.5.7
│       └── mime-types@2.0.14
│           └── mime-db@1.12.0
└── nodemailer@4.0.1

ubuntu@ubuntu-xenial:~/DaRonDon/src/main/nodejs/com.ronaldmcdonaldhouse.voluntee
rservice$ cd server
ubuntu@ubuntu-xenial:~/DaRonDon/src/main/nodejs/com.ronaldmcdonaldhouse.voluntee
rservice/server$ node www.js
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 must duplicate the shell by right clicking on the PuTTy cpu icon in the upper left corner of the shell window (see above pic) 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 (see pic below). We can forge ahead however. Lastly, Grunt is used to get the local server for the front end running using

grunt serve

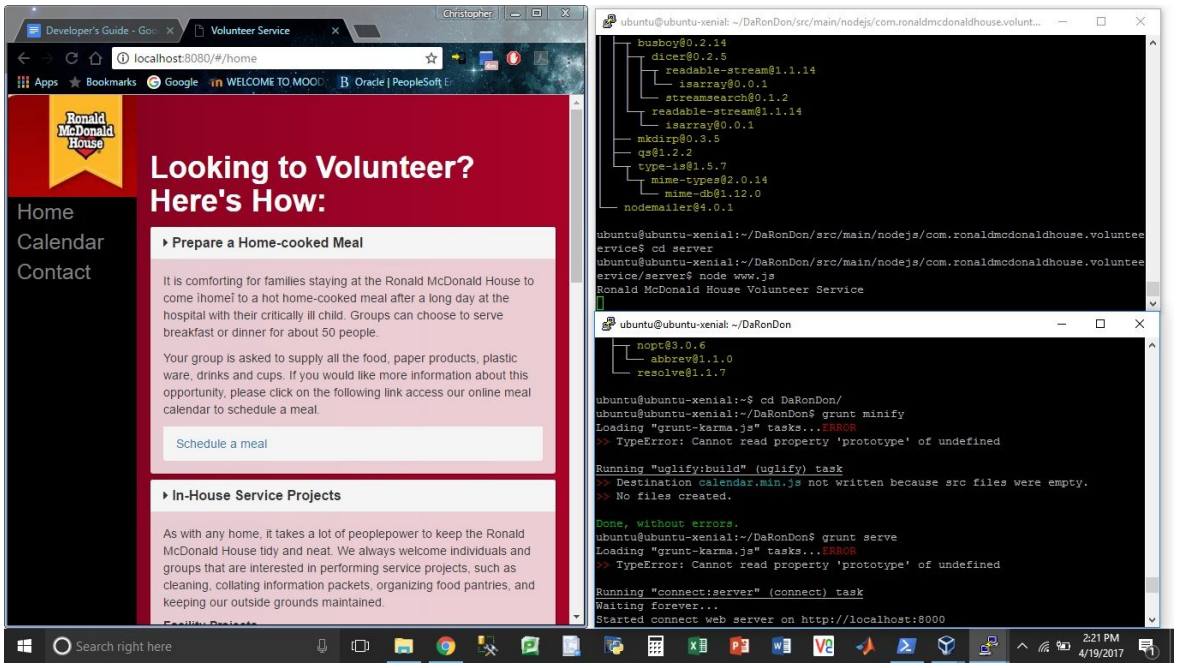
Another error maybe thrown (see pic below). Despite the error, the server should be up and running and can be verified by the following message

```
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. The two Ubuntu shells and the home page for the calendar



And the calendar