Project 2 - Installing a LAMP Web App: Kanboard

Goal:

The goal of this project is to deploy the Kanboard application. It's written in PHP, so you'll configure the LAMP stack for this web application. During this project I refer to the documentation for the application. It's located at the https://kanboard.org website. (https://docs.kanboard.org) You won't need it if you follow the steps in this project, however you can look at it for yourself to see how I came up with this installation process.

Instructions:

Create a Virtual Machine

First, start a command line session on your local machine. Next, move into the working folder you created for this course.

```
cd linuxclass
```

Initialize the vagrant project using the usual process of creating a directory, changing into that directory, and running "vagrant init". We'll name this vagrant project "kanboard".

```
mkdir kanboard
cd kanboard
vagrant init jasonc/centos8
```

Configure the Virtual Machine

Edit the Vagrantfile and set the hostname of the virtual machine to "kanboard". Also, assign the IP address of 10.23.45.25 to the machine.

```
config.vm.hostname = "kanboard"
config.vm.network "private_network", ip: "10.23.45.25"
```

Start the Virtual Machine

Now you're ready to start the VM, test its network connectivity, and connect to it.

```
vagrant up
```

Test the connection by pinging the virtual machine.

Windows users, run the following command:

```
ping 10.23.45.25
```

Mac users, run the following command:

```
ping -c 3 10.23.45.25
```

The ping command is one simple way to test network connectivity. If you see replies, then you can safely assume the IP address is reachable and the host is up. If you see "timeout" messages then the system is not answering your ping requests. In the "real world" this doesn't necessarily mean the system is "down." It means it is not answering your ping requests which could be for a variety of reasons. However, for our purposes here if you get "timeout" messages, then you can assume this system is down or something is wrong. The first thing to try is to simply reboot the VM by running:

```
vagrant reload
```

If the ping command fails again, double check the contents of the Vagrantfile paying special attention to the config.vm.network line. Make any necessary changes, restart the virtual machine and try again. The final step is to reboot the host operating system, I.E. your physical computer. This troubleshooting step primarily applies to Windows users.

If these steps fail, watch the "Vagrant and VirtualBox Troubleshooting" lesson and follow the steps there.

Connect to the Virtual Machine

Run this command to connect to the virtual machine.

```
vagrant ssh
```

Install Apache

Start off by installing the Apache HTTP Server.

```
sudo dnf install -y httpd
```

Install PHP

This web application is written in PHP, so you need to install PHP. Also, the application will be storing information in a database. This means we'll also need the php-mysqlnd package.

```
sudo dnf install -y php php-mysqlnd
```

Install additional PHP Modules

To determine if any additional PHP modules are required for an application, look at its documentation. Typically, it will tell you what is required. If it doesn't and it turns out to need additional modules, you'll need to start the web application and look at log files to determine what modules it is using and missing. Lucky for you, this application is well documented. It requires the GD, MbString, JSON, and XML modules.

Search for each module using dnf. Remember, if you need more information about a package to determine if it's the one you want, run "dnf info" followed by the package name.

```
dnf search php gd
dnf info php-gd
sudo dnf install -y php-gd

dnf search php mbstring
sudo dnf install -y php-mbstring

dnf search php json
sudo dnf install -y php-json

dnf search php xml
sudo dnf install -y php-xml
```

In the future, if you know all the package names for the modules, you can supply them on one command, like this. Since you've already installed all the modules, this command will not change anything at this point.

```
sudo dnf install -y php php-mysqlnd php-gd php-mbstring php-json php-xml
```

Start and Enable the Web Server

Now that the web server and PHP are installed, we can go ahead and start it. We want it to be enabled on boot, so we'll enable it as well.

```
sudo systemctl start httpd
sudo systemctl enable httpd
```

Install MariaDB

Go ahead and install MariaDB. While you're at it start and enable it.

```
sudo dnf install -y mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

Secure MariaDB

Let's secure the default MariaDB installation. The only information you need to supply is the password for the root database user. For our purposes of practicing our skills we can use a simple password such as "root123". In the real world I would suggest using a stronger password. For the remaining questions you can accept the defaults by pressing ENTER.

```
sudo mysql_secure_installation
```

Example:

```
...
Enter current password for root (enter for none): (press ENTER)
Set root password? [Y/n] (press ENTER)
New password: root123
Re-enter new password: root123
Remove anonymous users? [Y/n] (press ENTER)
Disallow root login remotely? [Y/n] (press ENTER)
Remove test database and access to it? [Y/n] (press ENTER)
Reload privilege tables now? [Y/n] (press ENTER)
```

Create a Database for the Application

Use the mysqladmin command to create a database named "kanboard". Enter the root password when prompted.

```
mysqladmin -u root -p create kanboard
```

Create a DB User for the Database

First, connect to the MariaDB server using the mysql client. Next, use the GRANT command to create the user, set the password, and allow full permissions to the "kanboard" database. Name the user "kanboard" and set the password to "kanboard123". Remember to flush the privileges.

```
mysql -u root -p
> GRANT ALL on kanboard.* to kanboard@localhost identified by
'kanboard123';
> FLUSH PRIVILEGES;
> exit
```

Download the Web Application

Download the web application. You can do that directly from your Linux system by using the `curl` command. Curl is most used to transfer data over a network such as downloading a file. The "-L" option tells `curl` to obey redirects. Use the "-O" option causes the file to be saved locally with the same name that was used on the remote system.

By the way, when you specify multiple options to a command you can specify them separately like this: `curl -L -O`. However, when those options aren't expecting anything following them you can

use a single hyphen and then combine all the single letter options like this: `curl -LO`. The order doesn't matter either, so you can do this as well: `curl -OL`. So, `curl -L -O`, `curl -O -L`, `curl -LO`, and `curl -OL` are all the same.

Remember to type in the entire command. Copying and pasting from PDFs, web pages, and other documents may cause problems. For example, you may see a hyphen, but actually what is in the document is a "pretty" character that represents that hyphen. If you were to copy and paste the contents in a terminal, it will not understand that character.

```
curl -LO http://mirror.linuxtrainingacademy.com/kanboard/kanboard-v1.2.15.zip
```

Internet download location: https://github.com/kanboard/kanboard/archive/v1.2.15.zip

Extract the Web Application

Now that you've downloaded the web application, it's time to extract its contents. Since it's a zip file, you can extract its contents with the `unzip` command.

```
unzip kanboard-v1.2.15.zip
```

Move the Web Application into the DocumentRoot

We're going to move the web application files into the DocumentRoot. By default, the DocumentRoot is `/var/www/html`, so that's where you'll place the files. Just like installing software with packages, putting files in most locations requires root privileges. Use sudo in conjunction with the `mv` command to perform this task. Confirm the move with `ls`.

```
sudo mv kanboard-1.2.15/* /var/www/html
ls -l /var/www/html
```

We need to tell the web application how to connect to the database. Per the application's documentation we need to create a configuration file named `config.php`. To create the file, simply edit it.

```
sudo nano /var/www/html/config.php
```

Add the following to the file and save it.

```
<?php
define('DB_DRIVER', 'mysql');
define('DB_USERNAME', 'kanboard');
define('DB_PASSWORD', 'kanboard123');
define('DB_HOSTNAME', 'localhost');
define('DB_NAME', 'kanboard');
```

The Web App is Ready

Now you can start using the application. Open up a web browser on your local machine. Enter the IP address of the server (http://10.23.45.25) into the address bar and press enter. You can log in with the username of "admin" and the password of "admin".

Feel free to explore the application. Even though you can start using this web application, the goal was to learn how to *deploy* the application on a Linux system. I'll leave it up to you as to how much you would like to explore the application.