ECKMAR's MARKETPLACE SCRIPT INSTALLATION INSTRUCTIONS

Requirements

- Linux VPS (Any Specifications)
- Bitcoind server

Setting Up VPS

First Login

When purchasing VPS you should get your login details. If you are on Linux or Mac you already have SSH client install. If you are on Windows you will have to download a client like PuTTY. After successful login first command that you should run is:

```
sudo apt-get update
```

You will see a bunch of text, and eventually you will get your command prompt again.

Installing Nginx

In order to install Nginx you have to run only one command:

```
sudo apt-get install nginx
```

Installing MySQL

Shopkin support multiple database solutions (SQLite, Postgres etc.) but we'll be using MySQL. In order to get strated run:

```
sudo apt-get install mysql-server
```

This will start installing MySQL. Let the install run until a bright pink/purple screen pops up. You will want to type in a password to use for the root MySQL user. Choose something secure here and then click the enter key. After you make your first password it will ask you to confirm. Obviously make sure they match. Also, this is a password you need to remember. So make sure you store the password somewhere safe or it is something you can remember. You will be using this password a lot going forward.

Optionally you can run:

```
sudo mysql_secure_installation
```

In order to secure your installation. If you decide to do so, just follow the on screen instructions.

Installing PHP

PHP comes in the form of a plugin called *php-fpm* (FastCGI Process Manager). In order to install php-fpm and other required plugins to make it work with Laravel and MySQL run:

```
sudo apt-get install php-fpm php-mysql php-mbstring php7.0-bcmath php7.0-intl
php7.0-gmp php7.0-mcrypt php-xml
```

After installation is complete you have to configure PHP. In your terminal, open up your *php.ini* file in whatever text editor you wish (VIM, or eMacs) but for simplicity, we will use Nano:

```
sudo nano /etc/php/7.0/fpm/php.ini
```

Press *Ctrl+W* and now type in *cgi.fix_pathinfo=* and click enter. This will take you to the right line right away. You will see a semicolon the left of this line. Delete the semi colon and then change the 1 into a 0 and save the file.

Before the changes can take effect we need to restart *php-fpm* by typing in this command:

```
sudo systemctl restart php7.0-fpm
```

Configuring Nginx

All the configuration we need to make is in the following config file. Go ahead and open it up in Nano using the following command (use another editor if you prefer).

```
sudo nano /etc/nginx/sites-available/default
```

You will see a lot of lines with # in front of them, these are comments. For simplicity, we will remove comments in this tutorial to make it easier to see what changed.

```
server {
listen 80 default_server;
listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }
}

```

In the line with all of the index names, we will add index.php to the list of allowed file types to deliver by default. What this line tells Nginx is to first look for an index file, then look for an index.php file, then an index.html file and so forth.

```
server {
listen 80 default_server;
listen [::]:80 default_server;

    root /var/www/html;
    Index index.php index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Next we need to add our public domain or IP address to the server_name line. This tells Nginx the domain to respond to. I am going to use an IP address for this tutorial since I am not setting up a domain. But if you have a domain name that you want this server to use then you would put the domain name here instead.

```
server {
listen 80 default_server;
listen [::]:80 default_server;

    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;

    server_name107.191.44.91;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Now we need to do a few other housecleaning items. You will want to just trust me on these as they get more complex, but the concepts of what they accomplish should make sense to you. First things first we want to tell Nginx to use your php-fpm that we installed earlier. This will be represented by the first location block that we add (it will actually be the second on in the document though, make sure to leave the first location block alone (for now, we will come back to configure it for Laravel later).

The second location block we are adding (the third in the file) will be telling Nginx to ignore .htaccess files. This is because .htaccess files are for Apache and we are using Nginx. Sometimes Laravel files will have .htaccess files in them by default so let's just make sure that if one gets onto our server to make sure it doesn't interfere with anything and our users do not have access to it.

```
server {
listen 80 default_server;
listen [::]:80 default_server;

    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;

    Server_name IP OR DOMAIN HERE;

    location / {
        try_files $uri $uri/ =404;
    }
    location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass unix:/run/php/php7.0-fpm.sock;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
}
}
```

When your config file looks something like this, save it. After that, we need to check for errors by running:

```
sudo nginx -t
```

If everything was correct then you should get this notice when submitting the command:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

This means you have no errors. In order for changes to take effect run:

```
sudo systemctl reload nginx
```

Create a folder for script

No we need to create folder for our script files, to do that run:

```
sudo mkdir -p /var/www/shopkin
```

Now, we need to edit our nginx config again to make it point to our folder:

```
server {
listen 80 default_server;
listen [::]:80 default_server;

    root /var/www/shopkin/public;
    index index.php index.html index.htm index.nginx-debian.html;

    Server_name IP OR DOMAIN HERE;

    location / {
        try_files $uri $uri/ =404;
    }
    location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass unix:/run/php/php7.0-fpm.sock;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
}
}
```

Change the root to */public* folder inside the script.

Finally we need to make one more change to the config. In order for script to recognize query strings we need to change one line in first location block to look like this:

```
server {
        listen 80 default_server;
        listen [::]:80 default_server ipv6only=on;

        root /var/www/laravel/public;
        index index.php index.html index.htm;

        server_name IP OR DOMAIN HERE;

        location / {
                    try_files $uri $uri/ /index.php?$query_string;
          }

        # more location blocks continue below
        # (no changes needed beyond this point)
```

Save the file and then run:

```
sudo service nginx restart
```

In order for changes to take effect.

Files And Permissions

Next thing to do is upload script files to the folder you made (/var/www/shopkin). You can do this using any FTP client. It will take some time but after you are done uploading, check if you have storage,public,app folders etc in your directory. If you do, everything is fine and now we need to assing some permissions to our folders. To do this run:

```
sudo chown -R :www-data /var/www/shopkin
```

```
sudo chmod -R 775 /var/www/shopkin/storage
```

```
sudo chmod -R 775 /var/www/public/uploads
```

```
sudo chmod -R 775 /var/www/public/bootstrap/cache
```

Database Setup

So we installed MySQL a while back, but we haven't set up an actual database inside of it yet to store our application data. So now is the time to go set that up, so that we can configure it and run our migrations.

Lets get into MySQL by typing the following command:

```
mysql -u root -p
```

Now you will be promted to enter your password.

In order to create table run:

```
CREATE DATABASE shopkin;
```

And then to exit from mysql:

```
exit
```

Configuring App Settings

App settings are stored inside .env file in app root directory. In order to edit it , cd to app directory and run:

```
nano .env
```

Most settings are named as to what they represent so it's pretty self explanatory.

Populate settings with DB prefix with MySQL database data, and settings with BITCOIND perfix with your Bitcoind data.

Now it’s good idea to cache our settings. To do that , run from app root:

```
php artisan config:cache
```

Next thing to do is migrate our database:

```
php artisan migrate
```

If you don’t see any errors you have set up database correctly. If you do, go back to MySQL setup and check for errors.

Next run:

```
php artisan db:seed
```

Looks like you are done !
Good like with your website !

App configuration is now done. If you wisth to know how to run it on TOR this is the tutorial:
https://enter.thewhiterabbit.space/host-tor-hidden-service-with-nginx/
And If you have problem with installing bitcoind on your server follow this series
https://www.youtube.com/watch?v=cMM-t7azzJE