**Deployment Setup**

The system was deployed in Digital Ocean Droplet – Ubuntu System. The following steps are done for the system to be able to run:

**Prerequisites**

In order to complete this guide, you will first need to perform the following tasks on your Ubuntu 18.04 server:

* Create a sudo user and enable ufw
* Install Nginx. If you haven’t set this up yet
* Install MySQL
* Install PHP 8 or higher

**Install Composer**

To verify which version of PHP your server is currently using, run:

$ php -v

Before you can install Laravel, you need to install a few PHP modules that are required by the framework. You’ll also need to install PHP-FPM in case you haven’t set that up yet.

Laravel requires the php-mbstring, php-xml and php-bcmath PHP modules. These PHP extensions provide extra support for dealing with character encoding, XML, and precision mathematics. Additionally, you’ll need php-mysql to be able to connect to the database via PHP code.

1. sudo apt update
2. sudo apt install php7.4-fpm php7.4-mysql php7.4-mbstring php7.4-xml php7.4-bcmath

**Creating a Database for the Application**

1. sudo mysql
2. CREATE DATABASE evsu-org;
3. GRANT ALL ON travel\_list.\* TO 'travel\_user'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;
4. exit
5. mysql -u travel\_user -p
6. SHOW DATABASES;

**Setting Up and Configuring Laravel**

1. Navigate to /var/www/
2. Clone your project in this directory
3. Run composer Install and npm install
4. Run php artisan migrate --seed

The Laravel configuration files are located in a directory called config, inside the application’s root directory. Additionally, when you install Laravel with Composer, it creates an environment file. This file contains settings that are specific to the current environment the application is running, and will take precedence over the values set in regular configuration files located at the config directory.

Even though there are many configuration variables in this file, you don’t need to set up all of them now. The following list contains an overview of the variables that require immediate attention:

APP\_NAME: Application name, used for notifications and messages.

APP\_ENV: Current application environment.

APP\_KEY: Used for generating salts and hashes, this unique key is automatically created when installing Laravel via Composer, so you don’t need to change it.

APP\_DEBUG: Whether or not to show debug information at client side.

APP\_URL: Base URL for the application, used for generating application links.

DB\_DATABASE: Database name.

DB\_USERNAME: Username to connect to the database.

DB\_PASSWORD: Password to connect to the database.

**Setting Up Nginx**

The updates and installation may take a few moments. We now need to allow HTTP traffic through our firewall.

sudo ufw allow 'Nginx HTTP'

We have installed Laravel on a local folder of your remote user’s home directory, and while this works well for local development environments, it’s not a recommended practice for web servers that are open to the public internet. We’ll move the application folder to /var/www, which is the usual location for web applications running on Nginx.

**Run the following command**

1. sudo chown -R www-data.www-data /var/www/project-name/storage
2. sudo chown -R www-data.www-data /var/www/project-name/bootstrap/cache

The application files are now in order, but we still need to configure Nginx to serve the content. To do this, we’ll create a new virtual host configuration file at /etc/nginx/sites-available:

1. sudo nano /etc/nginx/sites-available/travel\_list
2. server {
3. listen 80;
4. server\_name server\_domain\_or\_IP;
5. root /var/www/project-name/public;
6. add\_header X-Frame-Options "SAMEORIGIN";
7. add\_header X-XSS-Protection "1; mode=block";
8. add\_header X-Content-Type-Options "nosniff";
9. index index.html index.htm index.php;
10. charset utf-8;
11. location / {
12. try\_files $uri $uri/ /index.php?$query\_string;
13. }
14. location = /favicon.ico { access\_log off; log\_not\_found off; }
15. location = /robots.txt { access\_log off; log\_not\_found off; }
16. error\_page 404 /index.php;
17. location ~ \.php$ {
18. fastcgi\_pass unix:/var/run/php/php8.1-fpm.sock;
19. fastcgi\_index index.php;
20. fastcgi\_param SCRIPT\_FILENAME $realpath\_root$fastcgi\_script\_name;
21. include fastcgi\_params;
22. }
23. location ~ /\.(?!well-known).\* {
24. deny all;
25. }
26. }

Copy this content to your /etc/nginx/sites-available/travel\_list file and, if necessary, adjust the highlighted values to align with your own configuration. Save and close the file when you’re done editing.

To activate the new virtual host configuration file, create a symbolic link to travel\_list in sites-enabled:

1. sudo ln -s /etc/nginx/sites-available/travel\_list /etc/nginx/sites-enabled/

To confirm that the configuration doesn’t contain any syntax errors, you can use:

1. sudo nginx -t

To apply the changes, reload Nginx with:

1. sudo systemctl reload nginx

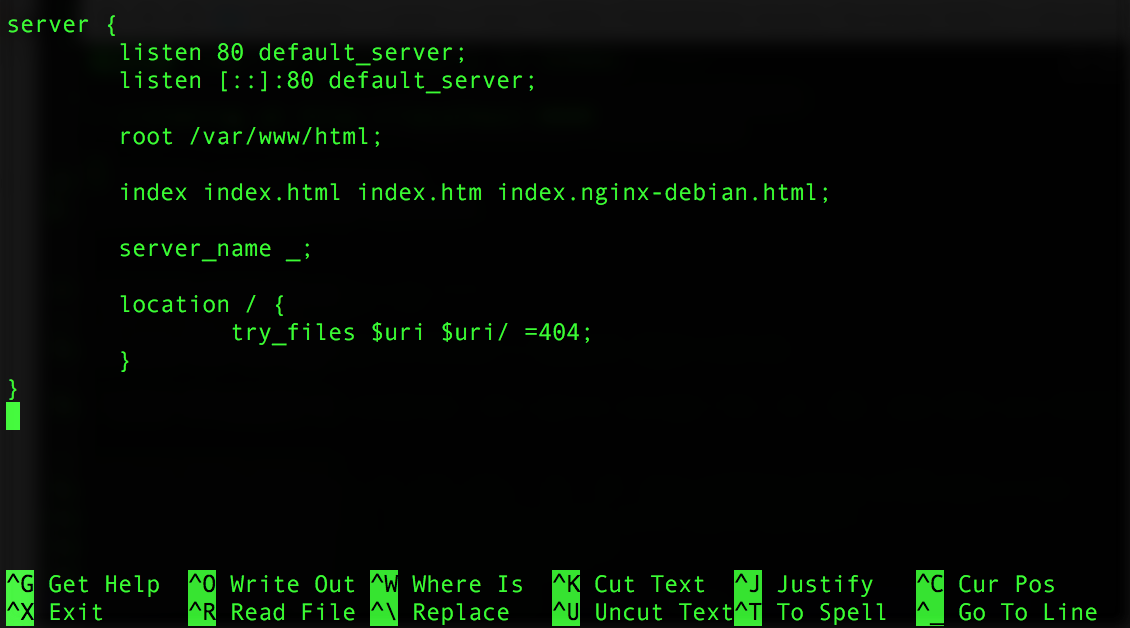
**Setting Up and Configuring Front End**

1. Navigate to /var/www/
2. Clone the frontend project in this directory
3. Run npm install

**Nginx Configuration**

We’re going to jump into the Nginx configuration to point where our application files will be server from.

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

Cut out all the comments with ^K. Your file should then look something like this:

**Now let’s make some minor edits and be done with the configuration. We’re going to change the root path to /var/www/html/Vue/dist. Next we’ll remove the try\_files $uri $uri/ =404; line. Lastly, you’ll add the following 404 redirect below the server\_name:**

error\_page 404 /index.html;

**We set the 404 redirect to our index.html file because that is the entry point for our VueJS application if you use**[**Vue-Router**](https://github.com/vuejs/vue-router)**. Our configuration should now look like the following:**

**Git Repo Setup**

Now we’ll need a way to deploy our app. Here’s the TL;DR of our plan:

Create app repo

Create bare repo

Create githook to receive our commits and direct them to app repo

**So let’s get started. Let’s create and navigate to our app repo.**

cd /var/www/html  
sudo mkdir Vue  
cd Vue

**This will be our repo for our application’s files. We’ll now need to initialize a Git repo inside of our app folder.**

git init

**That’s all we’ll need to do for this folder. Exit back out and let’s create the bare repo and do its setup.**

cd ..  
sudo mkdir repo  
cd repo

**From here, we’ll need to initialize a bare repo with the Git command:**

git init --bare

**We’re going to create a post-receive githook in our hooks folder. This will intercept our commits and push the files to the Vue folder.**

sudo nano hooks/post-receive

**Paste the following code:**

#!/bin/bashgit --work-tree=/var/www/html/Vue --git-dir=/var/www/html/repo checkout -f

**Save out the file with ^O, press enter, and exit the file with ^X. Next we’ll need to give this file permission to be executed.**

sudo chmod +x hooks/post-receive

**And that’s all we’ll need to do for our Git setup.**

**Deploy VueJS Application**

**Keep your terminal for your droplet open, as we’ll come back to it in a few moments. Switch over to your local files and navigate to your Vue application. We’ll need to initialize the local Vue app folder as a Git repo and set it up.**

git init  
git remote add origin root@YOUR\_DROPLET\_IP:/var/www/html/repo  
git add .  
git commit -m "Initial commit"

**This will prepare all of our files to be deployed to our bare repo on our droplet. To deploy, simply push your commit.**

git push origin master

**If you’re not using SSH Keys, you’ll be prompted to enter your droplet’s password. Do so and then the commit will be pushed to the droplet.**

**Anytime you want to push new changes from your local Vue app to your droplet, you will need to git add and git push the commits as we did above. So that’s it! We’ve now deployed our VueJS application to our droplet. But there’s a few last steps to complete!**

**Finalize Vue Application**

**Switch back over to your droplet console and navigate to your Vue folder.**

cd /var/www/html/Vue

**If you run the ls command, you should now see your application’s files all in the folder.**



**We’ll now need to install the app’s dependencies.**

npm install

Depending on how many dependencies your app requires, this may take a few moments. Once they’re installed, we’ll need to “bundle” our application. Since we’re using Webpack with our app, we have a built-in command to bundle our code for production. Let’s run it.

npm run build

This will create a dist folder containing all of our bundled code. Depending on how many modules you’ve added to your application, this process may take a few moments.

Once the bundling is done, your Vue application should now be available to view on the web! If you’ve setup a domain name to your droplet, head to the address, otherwise enter in the droplet IP in your browser. For my application, mine shows the default template for a Vue application.