Setting Up a Laravel Project on an EC2 Instance

Rimsha Azmat

July 25, 2024

1 Introduction

This document provides a comprehensive guide for setting up a Laravel project on an Amazon EC2 instance. It includes detailed instructions for installing necessary software, configuring the environment, resolving common issues, and ensuring a successful setup.

2 Prerequisites

Before starting the setup, ensure that you have:

- An AWS EC2 instance with Ubuntu 20.04 or later.
- SSH access to the EC2 instance.
- Basic knowledge of terminal commands and Linux systems.

3 Initial Setup

3.1 Connecting to EC2 Instance

To start, connect to your EC2 instance using SSH:

```
ssh -i "your-key.pem" ubuntu@your-ec2-ip
```

Why? This is necessary to access and manage your EC2 instance.

3.2 Updating System Packages

Update the system packages to ensure everything is up-to-date:

```
sudo apt update && sudo apt upgrade -y
```

Why? Ensures that all software and libraries are current, reducing the risk of security vulnerabilities and compatibility issues.

4 Installing Required Software

4.1 Installing Apache

Install the Apache web server:

```
sudo apt install apache2 -y
```

Why? Apache is a widely used web server that will serve your Laravel application.

4.2 Installing PHP and Extensions

Install PHP and the necessary extensions for Laravel:

```
sudo apt install php libapache2-mod-php php-mysql php-xml php-mbstring php-zip php-cu
```

Why? Laravel requires PHP along with several extensions to function properly.

4.3 Installing Composer

Install Composer, a dependency manager for PHP:

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
```

Why? Composer is needed to manage PHP dependencies and packages.

4.4 Installing Node.js and npm

Install Node.js and npm for managing JavaScript dependencies:

```
curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt install -y nodejs
```

Why? Node.js and npm are used to handle JavaScript and front-end dependencies.

4.5 Setting Up Swap Space

Create and enable swap space to handle memory usage:

```
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
sudo swapon --show
```

Why? Swap space helps to improve system performance and stability by providing additional virtual memory.

5 Cloning and Configuring Laravel Project

5.1 Cloning the Project Repository

Clone the Laravel project repository from GitHub:

```
cd /var/www
sudo git clone https://github.com/your-repo/laravel-boilerplate.git
cd laravel-boilerplate
```

Why? Cloning the repository copies the project code to your server.

5.2 Installing Project Dependencies

Install PHP and Node.js dependencies for the project:

```
sudo npm install --no-optional
composer install
```

Why? This installs all required libraries and packages for both front-end and back-end functionalities.

Issue: If you encounter errors related to missing modules or outdated packages, ensure that 'package.json' and 'composer.json' are correctly configured. You might need to update the packages or resolve version conflicts.

5.3 Setting File Permissions

Set the correct file permissions for the Laravel project:

```
sudo chown -R www-data:www-data /var/www/laravel-boilerplate
sudo chmod -R 775 /var/www/laravel-boilerplate/storage /var/www/laravel-boilerplate/b
```

Why? Correct permissions ensure that the web server can read and write necessary files, which is crucial for Laravel's functionality.

6 Configuring Apache

6.1 Creating Apache Virtual Host Configuration

Create a new Apache configuration file for Laravel:

```
sudo nano /etc/apache2/sites-available/laravel-boilerplate.conf
```

Why? This configuration file sets up how Apache serves your Laravel application.

```
<VirtualHost *:80>
    ServerAdmin webmaster@your-domain.com
    DocumentRoot /var/www/laravel-boilerplate/public
    ServerName your-domain.com
    <Directory /var/www/laravel-boilerplate/public>
        AllowOverride All
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Enable the new site and the rewrite module:

```
sudo a2ensite laravel-boilerplate.conf
sudo a2enmod rewrite
sudo systemctl reload apache2
```

Why? Enabling the site and the rewrite module allows Apache to serve your Laravel application and handle URL rewriting.

7 Final Steps

7.1 Setting Up Environment File

Copy the example environment file and generate the application key:

```
cp .env.example .env
php artisan key:generate
```

Why? The '.env' file contains environment-specific configurations, and generating an application key is essential for encryption and security.

7.2 Running Migrations

Run database migrations to set up the schema:

```
php artisan migrate
```

Why? Migrations are used to create and update database tables required by the Laravel application.

7.3 Clearing and Caching Configuration

Clear and cache the configuration to ensure changes take effect:

```
php artisan config:cache
php artisan cache:clear
```

Why? Caching the configuration improves performance by reducing the need to repeatedly parse configuration files.

8 Troubleshooting and Solutions

8.1 1. Webpack Compilation Errors

Issue: Errors during ‘npm run dev‘:

```
ERROR in ./node_modules/axios/lib/defaults.js 22:20-27
Module not found: Error: Can’t resolve ’process/browser.js’
```

Solution:

1. **Remove Duplicate Dependencies:** Open the ‘package.json‘ file and ensure that ‘laravel-mix‘ is listed only once and that ‘"process": "^{0}.11.10"‘*isaddedcorrectly* :

   ```
   sudo nano /var/www/laravel-boilerplate/package.json
   ```

Why? Removing duplicates and correctly listing dependencies prevents conflicts and ensures that all required modules are installed.

Reinstall Dependencies: After adjusting the ‘package.json‘ file, reinstall the npm dependencies:

```
sudo npm install
```

Why? Reinstalling dependencies ensures that the correct versions of all packages are installed, which is essential for successful compilation of frontend assets.

8.2 2. Apache Configuration Issues

Issue: Default Apache pages are shown instead of the Laravel application.

Solution:

1. **Correct Apache Configuration:** Ensure the Apache virtual host configuration file points to the Laravel public directory and that the rewrite module is enabled. Verify the configuration in the following file:

   ```
   sudo nano /etc/apache2/sites-available/laravel-boilerplate.conf
   ```

   The file should contain:

   ```
   <VirtualHost *:80>
       ServerAdmin webmaster@your-domain.com
       DocumentRoot /var/www/laravel-boilerplate/public
       ServerName your-domain.com
       <Directory /var/www/laravel-boilerplate/public>
           AllowOverride All
       </Directory>
       ErrorLog ${APACHE_LOG_DIR}/error.log
       CustomLog ${APACHE_LOG_DIR}/access.log combined
   </VirtualHost>
   ```

Enable the site and rewrite module:

```
sudo a2ensite laravel-boilerplate.conf
sudo a2enmod rewrite
sudo systemctl reload apache2
```

Why? Correct configuration ensures that requests are properly routed to your Laravel application rather than showing the default Apache page.

8.3 3. File Permission Issues

Issue: Laravel logs and caching are not functioning correctly.
Solution:

1. **Adjust Permissions:** Set the correct permissions for the Laravel storage and cache directories:

```
sudo chown -R www-data:www-data /var/www/laravel-boilerplate/storage /var/www
sudo chmod -R 775 /var/www/laravel-boilerplate/storage /var/www/laravel-boil
```

Why? Laravel requires write access to these directories to manage logs and cache files effectively. Correct permissions prevent issues with logging and caching functionality.

8.4 4. PHP Artisan Commands

Issue: Need to clear and cache configuration settings.
Solution:

1. **Cache Configuration:** Clear and cache the Laravel configuration to apply any changes:

```
sudo php artisan config:cache
sudo php artisan cache:clear
```

Why? Clearing and caching configuration ensures that Laravel uses the latest settings and improves performance by reducing configuration loading time.

8.5 5. Adding Swap Space

Issue: Out-of-memory errors during ‘npm install‘.
Solution:

1. **Create and Enable Swap Space:** Add swap space to handle memory shortages during intensive operations:

```
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
sudo swapon --show
```

Why? Swap space provides additional virtual memory when the system's physical RAM is exhausted, preventing out-of-memory errors during operations like 'npm install'.

9 Conclusion

Deploying a Laravel application on an EC2 instance involves multiple steps, including software installation, configuration, and troubleshooting. Proper handling of dependencies, server configuration, file permissions, and swap space ensures a smooth deployment and stable operation of the application.

9.1 Key Takeaways

- **Manage Dependencies:** Regularly update and resolve dependencies to avoid conflicts and issues.
- **Configure Apache Properly:** Ensure virtual host and rewrite module configurations are correct for routing and serving your application.
- **Monitor and Adjust Permissions:** Proper permissions are crucial for logging and caching functionalities.
- **Add Swap Space When Necessary:** Prevent out-of-memory errors by providing additional virtual memory during heavy operations.