

# Pixer - React Laravel Multivendor Digital Marketplace Documentation

---

“

For updated doc, please follow our online doc,

<https://pixer-doc.vercel.app/>

## Introduction

Fastest and secure digital assets selling E-commerce app built with `Laravel`, `React`, `NextJS`, `TypeScript` and `Tailwind CSS`. Its very easy to use, we use `axios` and `react-query` for data fetching . You can setup your api endpoint's very easily and your frontend team will love using it.

## Navigation

You can find different topics in the table of contents. On desktop, you should see it in the left sidebar. On mobile, you should see it after pressing an icon with Hamberger in the top right corner.

## Supported Platforms For Local Development

- Compatible Browsers (Firefox, Safari, Chrome, Edge)
- Server(Node.js 12.13.0 or later)
- MacOs, Windows, and Linux are supported

## Requirements

- node(12.13.0 or later)
- yarn
- PHP 7.4
- MySQL 8
- ext\_curl
- editor: Visual Studio Code (recommended)

## Tech We Have Used

We have used monorepo folder structure with Yarn Workspace. In our app we have three different services:

- api
- shop
- admin

Tech specification for specific part is given below:

## Admin Dashboard

- NextJs
- TypeScript
- Tailwindcss
- React Hook Form
- React-Query

# Shop Frontend

- NextJs
- TypeScript
- Tailwindcss
- React Hook Form
- React-Query
- Next SEO
- Next PWA

# API

- Laravel

For getting started with the template you have to follow the below procedure. For quick guide you can check below videos for installation.



# Prerequisites

- PHP 7.4
- Composer
- Xamp/Wamp/Lamp for any such application for apache, nginx, mysql
- PHP plugins you must need
  - simplexml
  - PHP's dom extension
  - mbstring
  - GD Library

## • Frontend

- node(12.13.0 or later)
- yarn
- editor: Visual Studio Code (recommended)



# Getting Started

First download the file from codecanyon.

Unzip the downloaded file and folder structure you get

```
pixer
|-- pixer-api
|-- admin
|-- shop
```

From the above folder structure you should notice that our app has three parts `pixer-api`, `shop` and `admin`. So you have to run all the parts separately and sequentially.

## - Getting started with api

- Make sure you have run xamp/mamp/wamp/lamp for mysql and php

- Create a database in your mysql and put those info in next step
- Rename .env.example file to .env and provide necessary credentials. Like database credentials stripe credentials, s3 credentials(only if you use s3 disk) admin email shop url etc.

Specially check for this `env` variables

```
DB_HOST=localhost  
DB_DATABASE=pickbazar_laravel  
DB_USERNAME=root  
DB_PASSWORD=
```

- Run `composer install`

```
▶ composer install  
Installing dependencies from lock file (including require-dev)  
Verifying lock file contents can be installed on current platform.
```

```
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: barryvdh/laravel-dompdf
Discovered Package: bensampo/laravel-enum
Discovered Package: cviebrock/eloquent-sluggable
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: fruitcake/laravel-cors
Discovered Package: ignited/laravel-omnipay
Discovered Package: intervention/image
Discovered Package: laravel/legacy-factories
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: mll-lab/laravel-graphql-playground
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: nuwave/lighthouse
Discovered Package: pickbazar/shop
Discovered Package: prettus/l5-repository
Discovered Package: spatie/laravel-medialibrary
Discovered Package: spatie/laravel-permission
Package manifest generated successfully.
95 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

- **run `php artisan key:generate`**

```
▶ php artisan key:generate
```

```
Application key set successfully.
```

- **Run `php artisan marvel:install` and follow necessary steps.**

```

Do you want to migrate Tables? If you have already run this command or migrated tables then be aware, it will erase all of your data. (yes/no) [no]:
> yes

Migrating Tables Now....
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (79.70ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (58.57ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (79.56ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (143.06ms)
Migrating: 2020_04_17_194830_create_permission_tables
Migrated: 2020_04_17_194830_create_permission_tables (839.79ms)
Migrating: 2020_06_02_051901_create_pickbazar_tables
Migrated: 2020_06_02_051901_create_pickbazar_tables (1,601.56ms)
Migrating: 2020_10_26_163529_create_media_table
Migrated: 2020_10_26_163529_create_media_table (71.72ms)
Tables Migration completed.

Do you want to seed dummy data? (yes/no) [no]:
> yes

Copying necessary files for seeding....
File copying successful
Seeding....
Seed completed successfully!

Do you want to create an admin? (yes/no) [no]:
> no

Copying resources files...
Installation Complete

```

- For image upload to work properly you need to run `php artisan storage:link`.

The [/var/www/html/public/storage] link has been connected to [/var/www/html/storage/app/public].  
The links have been created.

- run `php artisan serve`

```

▶ php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sat Apr 10 16:35:26 2021] PHP 8.0.0 Development Server (http://127.0.0.1:8000) started

```

“

NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT`  
env value will be `localhost:8000`/

- For MAC and Linux (with sail and docker)

There is an alternate installation procedure for linux and mac. You can follow below procedure to getting started with `sail`

- Prerequisites

- Docker

- Installation Mac

- REST API

- Run Docker application first
  - Now go to your pixer-laravel root directory and run `bash install.sh`. It will guide you through some process. Follow those steps carefully and your app will be up and running
  - Navigate to `pixer-api` then `sail down` to stop

the container. If you want to remove the volumes then `sail down -v`

“

NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost/`

“

For details api doc and requirements details you can go to [Laravel API](#)

## • Getting Started with Frontend

After configuring API & running it successfully you can choose the directory where you need to work

Below are the directories where you will choose to work for frontend stuffs

```
cd admin
```

```
cd shop
```

After choosing your working directory

Go to specific folder and rename the

.env.template ⇒ .env and put your api endpoint here. You will find .env.template file at the root of your admin or shop

Run yarn at the root directory.

```
# on pixer/root directory  
yarn
```

- **Scripts To Run the fronted App**

## - For Admin :

For starting the admin dashboard part with corresponding api data run below commands.

- using workspace (At the root of the pixer directory, you can run the below commands)

```
# for dev mode run below command  
yarn dev
```

```
▶ yarn dev:admin-rest  
yarn run v1.22.5  
$ yarn workspace @pick-bazar/admin-rest dev  
$ next dev -p 3002  
ready - started server on 0.0.0.0:3002, url: http://localhost:3002  
info  - Loaded env from /Users/bashar/Codes/pickbazar-sail/frontend/admin-rest/.env.local  
event - compiled successfully
```

- without workspace(if you want to run the command within specific project root of **admin/{chosen-directory-name}** )

```
# for dev mode run below command  
yarn dev
```

This command will run the app in development mode. Open the suggested url in your terminal. like ⇒ <http://localhost:3000> .

**\*\* Note:**

- The page will automatically reload if you make changes to the code. You will see the build errors and lint warnings in the console.
- If you saw any error while running make Sure you setup your API endpoint properly at [.env](#) file.

- For Shop :

- using workspace (At the root of the pixer directory, you can run the below commands)

```
# for dev mode run below command  
yarn dev:shop-rest
```

► [yarn dev:shop-rest](#)

```
yarn run v1.22.5
$ yarn workspace @pick-bazar/shop-rest dev
$ next dev -p 3003
ready - started server on 0.0.0.0:3003, url: http://localhost:3003
info  - Loaded env from /Users/bashar/Codes/pickbazar-sail/frontend/shop-rest/.env.local
```

- without workspace(if you want to run the command within specific project root of **shop**)

```
yarn dev
```

\*\* If you want to test your production build admin or shop in local environment then run the below commands. \*\*

- Admin (At the root of the pixer directory, you can run the below commands)

```
# build admin for production  
yarn build  
  
#start admin in production mode  
yarn start
```

- Shop (At the root of the pixer directory, you can run the below commands)

```
# build shop for production  
yarn build  
  
# start shop in production mode  
yarn start
```

```

$ yarn workspace @pick-bazar/shop-rest build
$ next build
info - Loaded env from /Users/bashar/Codes/pickbazar-sail/frontend/shop-rest/.env.local
info - Creating an optimized production build
info - Compiled successfully
info - Collecting page data
info - Generating static pages (48/48)
info - Finalizing page optimization

Page                                Size      First Load JS
[ _app                               0 B       176 kB
• /[type]
  | css/793e7aa084bd830e87c.css     27.2 kB   268 kB
  | grocery
  | makeup
  | bags
  | [+2 more paths]
  ○ 404
  • /bakery (ISR: 60 Seconds)
    | css/0dcfb0d56385c547.css
  λ /change-password
  ○ /checkout
  ○ /contact
  ○ /example
  ○ /help
  ○ /logout
  ○ /offers
  λ /order
    | css/db04274d0e1caa11303a.css
  λ /order-received/[tracking_number]
  λ /orders
    | css/da2daebf602ff72a3fc0.css
  ○ /privacy
  • /products/[slug]
    | css/efb94a24c22d59f4d562.css
    | /products/apples
    | /products/baby-spinach
    | /products/blueberries
    | [+27 more paths]
  λ /profile
  ○ /terms
  ○ /ui
+ First Load JS shared by all          176 kB
| chunks/00682edc09922f9a0564b6d51ebfe3fe9fe3f6bc.837c2b.js   3.14 kB
| chunks/03597eb9dee95bb8e6991ab21a5b170a1bc330ab_CSS.3bbe0c.js 68 B
| chunks/1c2031c8.c533da.js           23.5 kB
| chunks/2c96be8e9df36e109a607a23c3ac04c26631eb28.f78ccb.js  4.76 kB
| chunks/57ad4ee9d685f4a85f091607d3121ddc66ffef331.d3ef7d.js  35.8 kB
| chunks/6bc2a5fc5fb5a7f77ad800e4657cca465fac5a.173fc5.js   7.14 kB
| chunks/78478e0567901f0a00137009c7f6751fab214ebc.855475.js  15.3 kB
| chunks/aa9552d4f74729e02303abd0a7d42186859f2ec0.37727a.js  7.43 kB
| chunks/commons.ff7e9e.js            18.2 kB
| chunks/framework.f6e4f9.js          42.3 kB
| chunks/main.9c0e07.js              6.29 kB
| chunks/pages/_app.558fda.js        9.76 kB
| chunks/webpack.3d3782.js           2.3 kB
css/669ac54f0d96e08d6ac5.css         4.38 kB
css/f53121b36d6036aac827.css        11.2 kB

λ (Server) server-side renders at runtime (uses getInitialProps or getServerSideProps)
○ (Static) automatically rendered as static HTML (uses no initial props)
• (SSG) automatically generated as static HTML + JSON (uses getStaticProps)
  (ISR) incremental static regeneration (uses revalidate in getStaticProps)

★ Done in 96.20s.
▶ yarn start:shop-rest
yarn run v1.22.5
$ yarn workspace @pick-bazar/shop-rest start
$ next start -p 3003
ready - started server on 0.0.0.0:3003, url: http://localhost:3003
info - Loaded env from /Users/bashar/Codes/pickbazar-sail/frontend/shop-rest/.env.local

```

**\*\* Note \*\*:**

Please see `package.json` file for other builtin helper commands.

For development purpose we use `yarn workspace`

If you want to use it then see the `package.json` file at root, for various workspace specific command.

- if you prefer single template then just copy the required template folder and separate them. you'll find their `package.json` file within them and follow the command for dev, build, start.

For further development & customization check our [Frontend Customization](#) guide.

## Admin dashboard

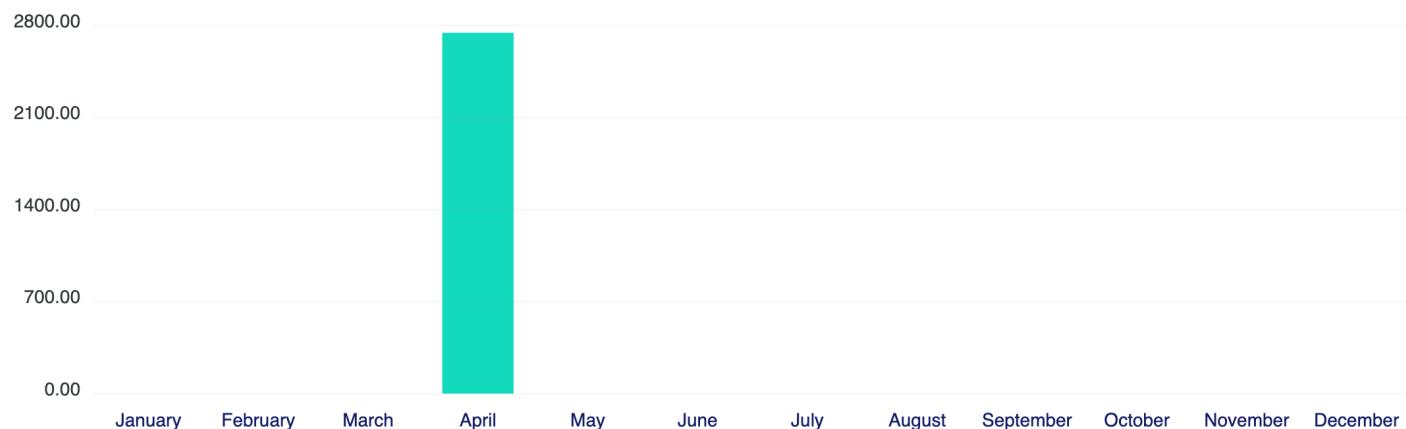
---

- **Analytics dashboard**

You will get a complete analytics dashboard to know the overview of your shop.

Total Revenue (Last 30 Days)		Total Order (Last 30 Days)		New Customer (Last 30 Days)		Todays Revenue	
\$7,844.05		25		16		\$0.00	

#### Sale History



#### Recent Orders

Tracking Number	Total	Order Date	Status
GkszijsVAcce	\$2,744.33	5 days ago	Order Received
lrrdghqshkys	\$1,748.30	14 days ago	Order Dispatched
iTUVhT87QdE8	\$526.98	17 days ago	Order Dispatched
A9mvTnzhGxkc	\$294.49	17 days ago	Order Received
B1OnTtmsHHHE	\$12.80	17 days ago	Order Received
rulb1kz4UIIN	\$2.45	19 days ago	Shipment Refused by Consignee
NqlykelW6awe	\$97.80	18 days ago	Delivered
TfzYmld1AR32	\$18.00	18 days ago	Ready To Dispatch
PXqs8RXQBGM5	\$7.20	19 days ago	Delivered
K2isul12Cetf	\$540.00	19 days ago	Ready To Dispatch

Popular Products					
ID	Name	Group	Shop	Price/Unit	Quantity
51	Borobazar React Next Grocery Template	Fixed	BentaSoft	\$69.00	500
50	Scholar Multipurpose Education WordPress Theme	Liquid	Bitronic	\$79.00	500
52	ShppingPro Joomla Template	Responsive	Imagineco	\$100.00	500
49	ChawkBazar Laravel Flutter Mobile App	Responsive	Omnico Team	\$39.00	500
19	Blogsy Agency Blog Theme	Responsive	Maxicon Soft Tech	\$79.00	500
40	Addingly Modern WordPress Theme	Responsive	Bitronic	\$35.00	500
41	Shippipro Rental Laravel Script	Fixed	Temper studios	\$69.00	500
39	Addingpro Charity Template	N/A	Omatron	\$29.00	500
38	Superprops 2.0 Shopify WordPress Theme	Fixed	Imagineco	\$55.00	500
21	Sootify Laravel Wireframe Kits	N/A	BentaSoft	\$49.00	500

## • Manage Layout Type

In **Layout Types** menu you will get the product types and you can add, remove or modify product type from there.

Layouts Type		Icon	Actions
ID	Name		
4	N/A	∅	
3	Responsive	▢	
2	Liquid	💧	
1	Fixed	★	

## • Manage Product Category

In **Categories** menu you will get the product types and you can add, remove or modify product categories from there.

ID	Name	Details	Image	Icon	Slug	Actions
17	Shopify	Along With Wordpress T...			shopify	
16	Joomla	Along With Wordpress T...			joomla	
15	Bootstrap	Along With Wordpress T...			bootstrap	
14	3D Assets	Along With Wordpress T...			3d-assets	
13	Mobile App	Along With Wordpress T...			mobile-app	
12	Icon Sets	Along With Wordpress T...			icon-sets	
11	Illustrations	Along With Wordpress T...			illustrations	
10	UI templates	Along With Wordpress T...			ui-templates	
9	Wireframe Kits	Along With Wordpress T...			wireframe-kits	
8	CMS	Along With Wordpress T...			cms	

## • | Product Management

In **Products** menu you will get the products and you can add, remove or modify products from there.

Products							
Image	Name	Group	Shop	Price/Unit	Quantity	Status	Actions
	ShippingPro Joomla Template	Responsive	Imagineco	\$100.00	500	<button>publish</button>	
	Borobazar React Next Grocery Template	Fixed	BentaSoft	\$69.00	500	<button>publish</button>	
	Scholar Multipurpose Education WordPress Theme	Liquid	Bitronic	\$79.00	500	<button>publish</button>	
	ChawkBazar Laravel Flutter Mobile App	Responsive	Omnico Team	\$39.00	500	<button>publish</button>	
	MagazinPro Lyfestyle Blog Template	N/A	BentaSoft	\$35.00	500	<button>publish</button>	
	Reactify Searching Engine	Fixed	Maxicon Soft Tech	\$55.00	500	<button>publish</button>	
	RNB Modern Laravel React Rental System	Liquid	Imagineco	\$0.00	500	<button>publish</button>	
	StoryHub WordPress Blog Theme	Fixed	Omatron	\$59.00	500	<button>publish</button>	
	Reactify Ecommerce Theme with Dashboard	Fixed	Qubitron Solutions	\$49.00	500	<button>publish</button>	

## • A portion of product form

**Featured Image**  
Upload your product featured image here

**Gallery**  
Upload your product image gallery here

**Group\***  
Responsive

**Categories**  
Angular x Shopify x Joomla x Icon Sets x Wireframe Kits x Free x

**Tags**  
modern x Dashboard x E-commerce x Crypto x Retail x

## • ORDER STATUS

In **Order Status** menu you will get the order status list and you can add, remove or modify order status from there.

The screenshot shows a table-based interface for managing order statuses. At the top left is a header 'Order Status'. To its right is a search bar containing the placeholder 'Type your query and press enter'. On the far right is a green button labeled '+ Add Order Status'. Below the header is a table with columns: ID, Name, Serial, and Actions. The table contains 11 rows of data. Red arrows point to three specific elements: one to the search bar, one to the '+ Add Order Status' button, and one to the 'Actions' column header. The table data is as follows:

ID	Name	Serial	Actions
1	Order Received	1	
2	Order Processing	2	
3	Ready To Dispatch	3	
4	Order Dispatched	4	
5	At Local Facility	5	
6	Out For Delivery	6	
8	Failed to collect payment	8	
9	Failed to contact Consignee	9	
10	Shipment Refused by Consignee	10	
7	Delivered	11	

At the bottom right of the table are navigation icons: a left arrow, a green square with the number '1', and a right arrow.

## Order Management

In **Order** menu you will get the order list and

... order more, you will see the list and

you can add, remove or modify order from there.

Tracking Number	Total	Order Date	Status	Actions
[+] magivv6B8c4x	\$39.90	2 days ago	Confirmed	(o)
[+] qZM6xOyvFz0V	\$84.80	2 days ago	Confirmed	(o)
[+] oudg2Ebf9Gww	\$105.00	2 days ago	Confirmed	(o)
[+] scU4nn9EOABM	\$162.75	2 days ago	Confirmed	(o)
[+] SA6cCiT3Lod5	\$150.00	2 days ago	Confirmed	(o) ↗
[+] xATLDw80ipU0	\$195.00	2 days ago	Confirmed	(o)
[+] RkekswCuelP6	\$158.00	2 days ago	Confirmed	(o)
[+] QM0fKBzywmLF	\$49.00	2 days ago	Confirmed	(o)
[+] BtrYveTqtFmc	\$110.00	2 days ago	Confirmed	(o)

- A portion of order management.
- Order status change.

Order ID - magjvy6B8c4x



Download Invoice

Confirmed       On Hold 

Select...  Change Status

Confirmed

On hold



Products	Total
 Superify x 2	\$38.00
Sub total	\$38.00
Tax	\$1.90
Delivery fee	
Discount	
<b>Total</b>	<b>\$39.90</b>

## ←!— ## Coupon Management

In **Coupon** menu you will get the Coupon list and you can add, remove or modify Coupon from there.

Coupons

Type your query and press enter

+ Add Coupon

ID	Banner	Code	Amount	Active	Expired	Actions
3	\$12 OFF	EID12	\$12.00	15 days ago	in a month	
4	\$15 OFF	HELLO15	\$15.00	15 days ago	in 12 days	
5	\$18 OFF	BAZAR18	\$18.00	15 days ago	in 4 days	
6	\$20 OFF	BAZAR20	\$20.00	15 days ago	in a year	
7	\$10 OFF	BAZAR10	\$10.00	14 days ago	in a month	
10	\$4 OFF	4OFF	\$4.00	14 days ago	14 days ago	
9	\$5 OFF	5OFF	\$5.00	14 days ago	in 2 months	



## Customer Management

In **Customer** menu you will get the Customer list and you can add, remove or modify Customer from there.

The screenshot shows a list of customers with columns for Avatar, Name, Status, and Actions. A modal window titled "Block Customer" is displayed in the center, asking "Are you sure you want to block this customer?". It contains two buttons: "Cancel" (green) and "Block" (red). Red arrows point from the top right towards the "Add Customer" button and the "Actions" column header. Another red arrow points from the bottom right towards the "Block" button in the modal.

Avatar	Name	Status	Actions
	again@another.net	Active	
	abcd@dd.io	Inactive	
	Litan	Active	
	III	Active	
	Pobon Paul	Active	

## Tax Management

In **Tax** menu you will get the Tax list and you can add, remove or modify Tax from there.

The screenshot shows a list of taxes with columns for ID, Name, Rate (%), Country, City, State, ZIP, and Actions. A modal window titled "Taxes" is displayed in the center, containing a search bar and an "+ Add Tax" button. Red arrows point from the top right towards the "Add Tax" button and the "Actions" column header. Another red arrow points from the bottom right towards the edit and delete icons in the Actions column.

ID	Name	Rate (%)	Country	City	State	ZIP	Actions
1	Global	2					

## Shipping Management

## Shipping Management

In **shipping** menu you will get the shipping list and you can add, remove or modify shipping from there.

The screenshot shows a table with a single row of data. The columns are labeled: ID, Name, Amount, Global, Type, and Actions. The data row contains: 1, Global, 50, 1, fixed, and a set of red arrows pointing to the search bar, the 'Add Shipping' button, the 'Name' column, the 'Amount' column, the 'Actions' column, and the delete/edit icons. A green header bar at the top has the title 'Shipments' and a search bar with placeholder text 'Type your query and press enter'. A red arrow also points to the search bar.

ID	Name	Amount	Global	Type	Actions
1	Global	50	1	fixed	

## Shipping Management

In **shipping** menu you will get the shipping list and you can add, remove or modify shipping from there.

The screenshot shows a table with a single row of data. The columns are labeled: ID, Name, Amount, Global, Type, and Actions. The data row contains: 1, Global, 50, 1, fixed, and a set of red arrows pointing to the search bar, the 'Add Shipping' button, the 'Name' column, the 'Amount' column, the 'Actions' column, and the delete/edit icons. A green header bar at the top has the title 'Shipments' and a search bar with placeholder text 'Type your query and press enter'. A red arrow also points to the search bar.

ID	Name	Amount	Global	Type	Actions
1	Global	50	1	fixed	

## Settings Management

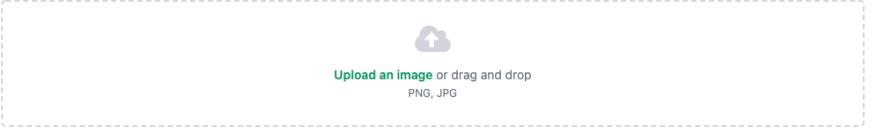
# Settings Management

In **settings** menu you will get the settings management form there.

**Settings**

---

**Logo**  
Upload your site logo from here.  
Dimension of the logo should be **96x32 Pixel**

  
Upload an image or drag and drop  
PNG, JPG

---

**Information**  
Change your site information from here

<b>Site Title</b>	Pixer
<b>Site Subtitle</b>	
<b>Currency</b>	US Dollar
<b>Minimum Order Amount</b>	
<b>Wallet Currency Ratio</b>	

**Information**  
Change your site information from here

<b>Site Title</b>
-------------------

Change your site information from here

Pixer
Site Subtitle
Currency
US Dollar
Minimum Order Amount
Wallet Currency Ratio
0
Sign Up Points
0
Use OTP at checkout
<input checked="" type="checkbox"/>
Tax Class
Qubitron Solutions

## Available Scripts:

You can run below commands in the root folder for your need.

- Shop



```
"clean": "rimrat \"\{node_modules, .next, .cache\}\\"",  
"dev": "next dev",  
"build": "next build",  
"start": "next start",  
"lint": "next lint",  
"prepare": "husky install"
```

- **Admin**

```
"dev": "next dev -p 3002",  
"build": "next build",  
"start": "next start -p 3002"
```

**For customizing the template's default site settings:**

“

[your-frontend-project] = **admin** or **shop**

- To setup your site's basic information like

[Logo, Site title, Description, Menus, etc] go to → `src/settings/site-settings.ts` file

- To customize tailwind configuration go to → `tailwind.config.js` file.
- `/public`: To change your app favicon images here.
- `/src/assets`: We managed our css & images in this directory.
- `/src/components`: This folder contains all the app related components.
  - `auth`: In this folder we contains our auth components and logics.
  - `cart`: Here you will find all of our cart & checkout components, utilities function, context api and etc.
  - `drawer-views`: We managed all of our side Drawer's view, context api & drawer UI in this folder.

- `modal-views` : We managed all of our modal's view, context api & modal UI in this folder.
- `icons` : Our app's custom svg icons components directory, if you need any then add your custom svg icon components here.
- `product` : All the product related card, popup, loaders, description etc components in this folder.
- `search` : Search handler, Search Popup & Results related components are here.
- `shop` : Shop related components are goes in this folder.
- `ui` : This folder contains common reusable ui components.
- `/src/config` : This folder contains all necessary configuration related for this app. Like `env`, `routes` etc.
- `/src/data` : It's contain all the data fetching

related codes along side with our app's static data.

- `/src/data/static` : Here you can find our terms, privacy, help, licensing data and in the `site-settings.ts` file we manage our dark & light mode logo along side with our explore page carousel images.
- `/src/layout` : It's contain all layouts and layout's related components like `header`, `bottom navigation`, `sidebar`, `container` and etc .
- `/src/lib` : This folder contains `constants`, `hooks`, `framer motion` & `general utils functions` .
- `/src/pages` : All the pages created here which is used by nextjs routing mechanism.
- `/src/types` : Reusable function & component's types are located in this folder.

NOTE \*\* Some of these options are customizable

through ADMIN Dashboard.

## | CSS styles:

“

[your-frontend-project] = `admin` or `shop`

We use tailwindcss framework with some customization which you find at :

```
open [your-frontend-project]/tailwind.config.js
```

For tailwindcss documentation:

Go to [Tailwindcss](#)

## | Icons:

for our icons

```
open [your-frontend-project]/src/components/icons
```

## For Adding a custom Icon:

To add a custom icon please follow this procedure.

Open your custom SVG icon file in the code editor and copy all the code.

Then Go to src → components → icons folder and create a new .tsx file.

Then here create a function component and paste your copied SVG code inside the return statement.

Then convert all the SVG's kebab-cases properties into camelCase format except the

data-name property. For ex. change the stroke-width and fill-color into strokeWidth and fillColor. (for refference you can see one of our icon. )

If your custom SVG code has any single custom color then change them into fillColor.

In this template, We have used some custom helper functions which is located in

> *[your-frontend-project]* = *admin* or *shop*

```
cd [your-frontend-project]/src/utils/
```

You can use or customize these helper fuctions

according to your needs.

## For API Integration:

> *[your-frontend-project]* = *admin* or *shop*

We have used env variables using *.env* file format. You have

to put your API url within this file.

For example:

Put that url in the *shop/.env* and *admin/.env*

```
NEXT_PUBLIC_REST_API_ENDPOINT=
```

```
'{put_your_api_url_here}'
```

# Data Fetching

“

[your-frontend-project] = `admin` or `shop`

For this project we provide an laravel rest api integration. We have used `react-query` ~ hook pattern ~ and fetched data from laravel API. Please go to `data/` folder for those hooks.

- Creating the hook.
  - We have imported the `Product` type from `@/types` (We have used typescript path aliasing for this. For more info please see our `tsconfig.json` file).
  - We have built an `axios instance` and a wrapper class which called `http-client`.

- We have put all ours endpoint at `@data/client/endpoints` file using constant value.
- We have created a `client` class which will define all the methods required for API actions.

```
class Client{
  users = {
    me: () => HttpClient.get<User>
  (API_ENDPOINTS.USERS_ME),
    update: (user: UpdateProfileInput) =>
      HttpClient.put<User>
  (`${API_ENDPOINTS.USERS}/${user.id}` , user),
    login: (input: LoginUserInput) =>
      HttpClient.post<AuthResponse>
  (API_ENDPOINTS.USERS_LOGIN, input),
    register: (input: RegisterUserInput) =>
      HttpClient.post<AuthResponse>
  (API_ENDPOINTS.USERS_REGISTER, input),
    forgotPassword: (input: ForgetPasswordInput) =>
```

```
        HttpClient.post<PasswordChangeResponse>(
            API_ENDPOINTS.USERS_FORGOT_PASSWORD,
            input
        ),
    }
    ...
}
export default new Client();
```

## • using the client class

you can use the client class to fetch data from the API. like below

```
import client from '@/data/client';
async () => await client.users.me();
```

- We have built our `product hook useProduct` using `react-query` and `the client instance`.

```
import type { Product } from '@/types':
```

```
import { type } from 'cosmiconfig-types';
import { useQuery } from 'react-query';
import { API_ENDPOINTS } from
'@/data/client/endpoints';
import client from '@/data/client';

export function useProduct(slug: string) {
  const { data, isLoading, error } =
useQuery<Product, Error>(
  [API_ENDPOINTS.PRODUCTS, slug],
  () => client.products.get(slug)
);
  return {
    product: data,
    isLoading,
    error,
  };
}
```

For more information about [react-query](#) please visit [React Query](#).

- Using the hook

```
const { product, isLoading, error } =  
useProduct(slug.toString());
```

# AWS (Amazon Web Service)

---

“

If you want to use all the scripts (`shop`, `admin`, `api`) on the same server as this tutorial, then we recommend creating a blank ubuntu-based server with at least 2+ CPU cores and 2GB+ memory.

## How to create ec2 server?

In this AWS tutorial, we're going to create an ec2 server. To do that at first, login to your AWS account and then click,

ec2 -> Instance -> Launch Instance

AWS Management Console

AWS services

Recently visited services

All services

- Compute
  - EC2
  - Lightsail
  - Lambda
  - Batch
  - Elastic Beanstalk
  - Serverless Application Repository
  - AWS Outposts
  - EC2 Image Builder
- Containers
  - Elastic Container Registry
  - Elastic Container Service
  - Elastic Kubernetes Service
- Storage
  - S3

Quantum Technologies

- Amazon Braket

Management & Governance

- AWS Organizations
- CloudWatch
- AWS Auto Scaling
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Systems Manager
- AWS AppConfig
- Trusted Advisor
- Control Tower
- AWS License Manager

Security, Identity, & Compliance

- IAM
- Resource Access Manager
- Cognito
- Secrets Manager
- GuardDuty
- Inspector
- Amazon Macie
- AWS Single Sign-On
- Certificate Manager
- Key Management Service
- CloudHSM
- Directory Service
- WAF & Shield
- AWS Firewall Manager
- Artifact

Stay connected to your AWS resources on-the-go

Download the AWS Console Mobile App to your iOS or Android mobile device.

Explore AWS

Amazon S3 on Outposts

You can now use S3 object storage in your on-premises environment. [Learn more](#)

Free Digital Training

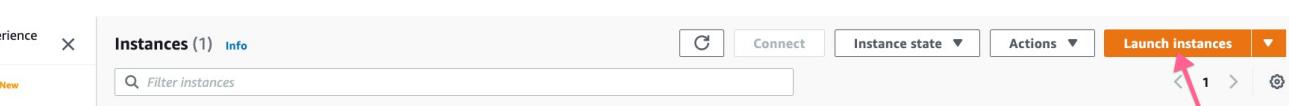
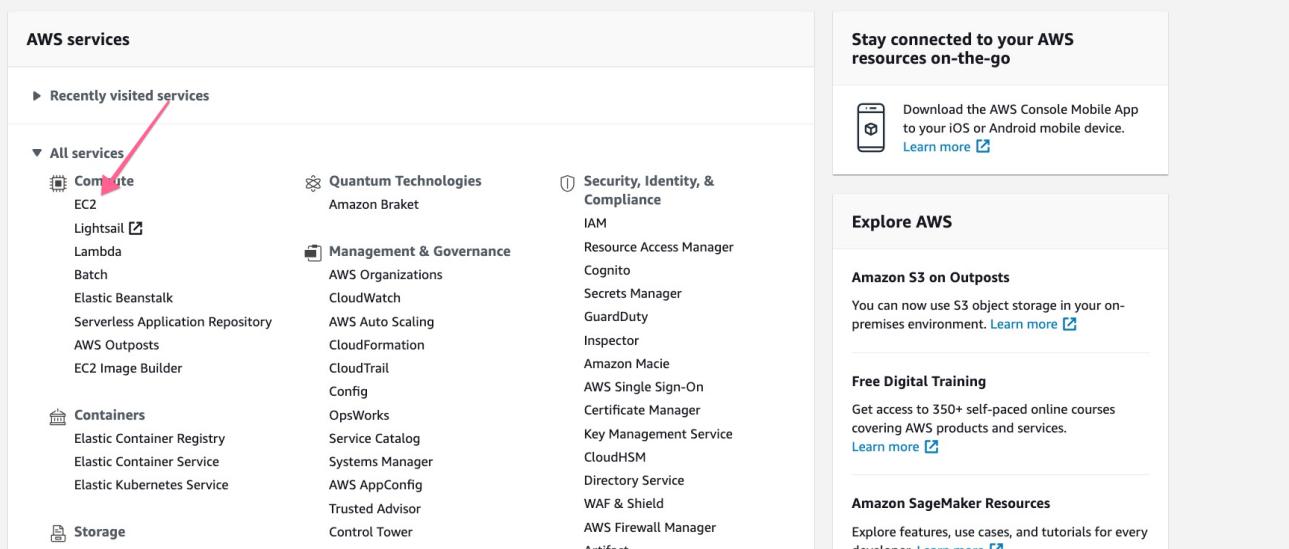
Get access to 350+ self-paced online courses covering AWS products and services.

Amazon SageMaker Resources

Explore features, use cases, and tutorials for every developer. [Learn more](#)

Feedback English (US) ▾

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with sections like Events, Tags, Limits, Instances (with a red arrow pointing here), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts (New), Scheduled Instances, Capacity Reservations, Images (AMIs), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main area shows a table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. One row is highlighted with a red border, showing 'Termina...', 't2.micro', and 'us-east-1e'. Below the table, a message says 'Select an instance above'.

Then select a ubuntu 20.04 server

After that, click **Next** → **Next** → **Next** → **Next**

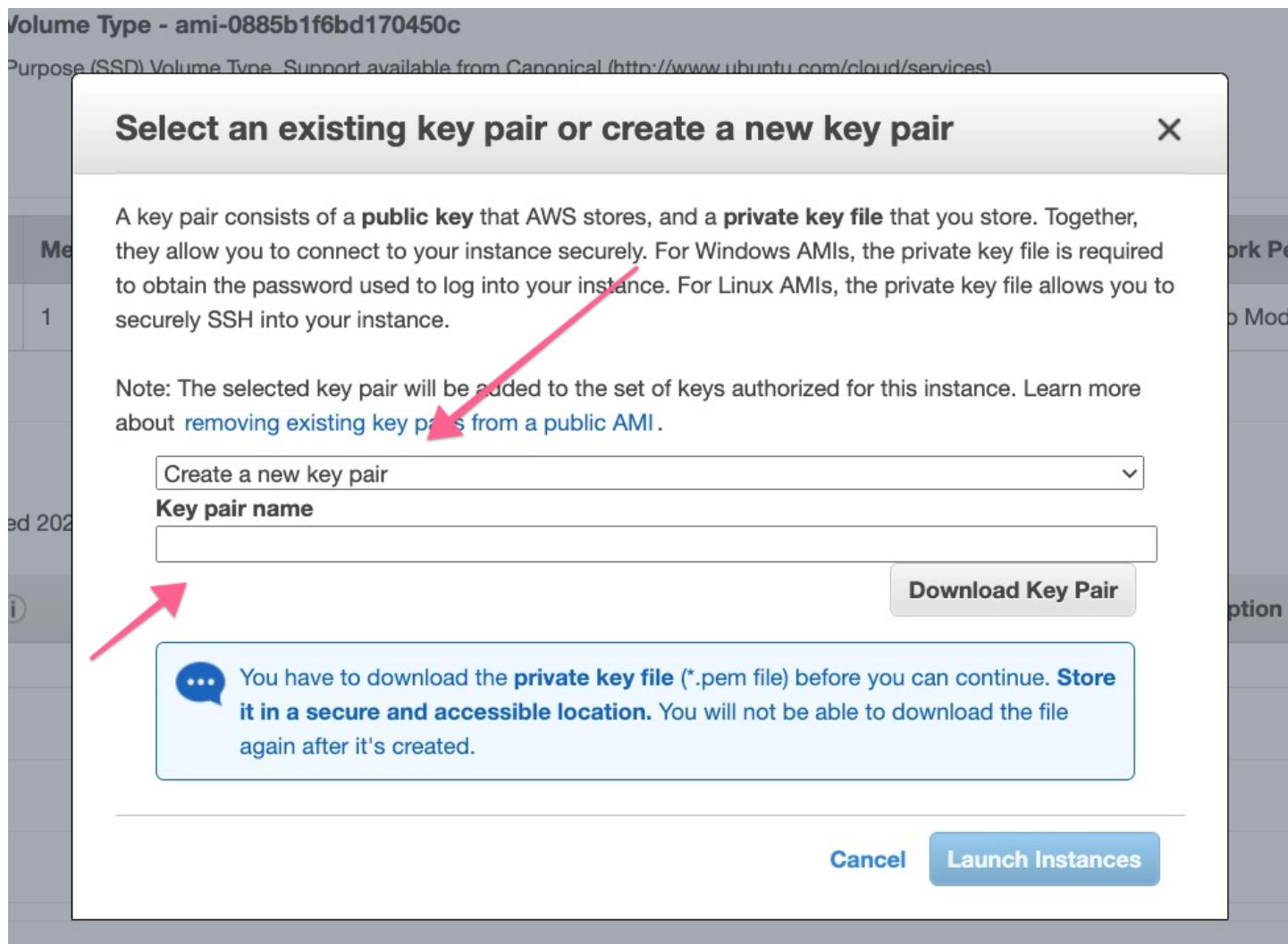
And on security pages, add a rule for **HTTP**,

The screenshot shows the 'Step 6: Configure Security Group' page. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group (which is selected), and 7. Review. The main content area is titled 'Step 6: Configure Security Group'. It explains that a security group is a set of firewall rules that control the traffic for your instance. It shows two rules: one for SSH (TCP port 22) and one for HTTP (TCP port 80). The 'Type' dropdown is set to 'SSH' and 'HTTP'. The 'Protocol' dropdown is set to 'TCP'. The 'Port Range' dropdowns show '22' and '80'. The 'Source' dropdowns are set to 'Custom' with '0.0.0.0/0'. The 'Description' fields are 'e.g. SSH for Admin Desktop'. A red arrow points to the 'Add Rule' button at the bottom left. A warning message at the bottom right says: 'Warning: Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.'

After review, click **Launch**, and you'll get and

popup for KeyPair, which will be required to login to the server using ssh.

If you already have a previous KeyPair, you can use that; otherwise, you can create a new one. After completing that, make sure you download that KeyPair.



After launching the instance, you'll get the

server IP, which will be required to login into ssh.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, Events, Tags, Limits, Instances (selected), Images, AMIs, and Elastic Block Store. The main area displays two instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-0fb9dda581f15817	Terminated	t2.micro	-	2 alarms	us-east-1e	-
car	i-0b3ad96ed56b60940	Running	t2.micro	-	No alarms	us-east-1d	ec2-54-90-111-167

Below the table, a modal window is open for the instance with ID i-0b3ad96ed56b60940, named "pickbazar". It has tabs for Details, Security, Networking, Storage, Status Checks, Monitoring, and Tags. The Details tab is selected. Under Instance summary, the Public IPv4 address is listed as 54.90.111.167, with a link to "open address".

## Domain Setup

Now copy the server IP and connect it with your domain name.

This screenshot shows the same AWS EC2 Instances page as the previous one, but now only one instance is listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
car	i-0b3ad96ed56b60940	Running	t2.micro	-	No alarms	us-east-1d	ec2-54-90-111-167

The modal window for this instance is still open, showing the Public IPv4 address as 54.90.111.167.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like Tags, Limits, Instances, Images, and Elastic Block Store. The Instances section is expanded, showing two instances: one terminated and one running. A red arrow points to the running instance, which is named 'pickbazar'. Another red arrow points to the 'Public IPv4 address' field, which contains '54.90...'. The main content area shows the instance details for 'pickbazar', including tabs for Details, Security, Networking, Storage, Status Checks, Monitoring, and Tags. The Details tab is selected, displaying the Instance ID, Public IPv4 address, and Private IPv4 addresses.

“

Please contact your domain provider for detailed explanation of how to do that.

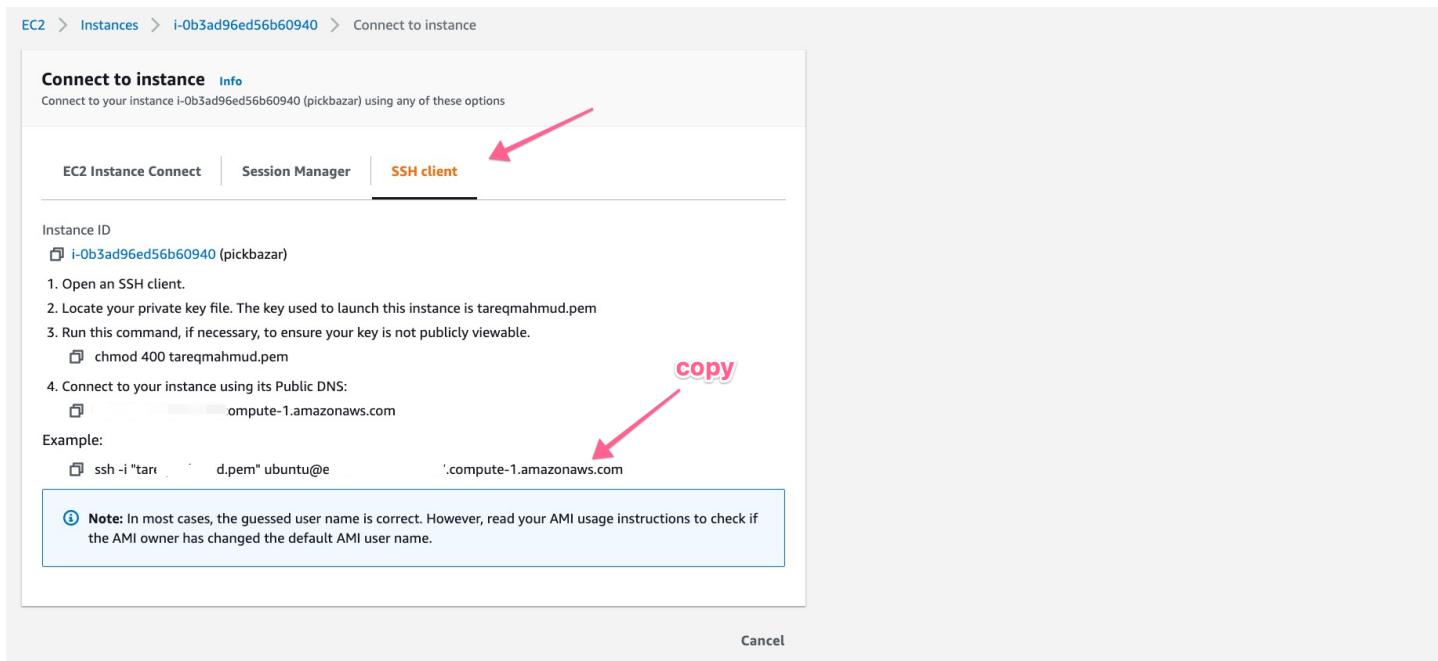
## Login to Server

At first, login to your **AWS** server using ssh. to do that, go to the folder from the terminal where **KeyPair** is downloaded.

then click **Connect**



From the **Connect** dashboard, go to **SSH Client** and copy the example line and paste it to your terminal.



With this command, you will successfully connect to your server throw ssh.

- **Change permission .pem**

You've to change the permission downloaded **.pem**

file to `400` to access the server. To do that, at first go to the location where `.pem` store then run,

```
chmod 400 pixer.pem
```

Change the `pixer.pem` filename if you use a different name during generate the key.

“

Now go to the `VPS Server` section for deploy the `Pixer Laravel`

- [VPS Server](#)

# Virtual Private Server

---

With this tutorial, you can install Pixer to

any type of blank or empty ubuntu server. For example, Digital Ocean Droplets, Amazon Lightsail, AWS, Google Cloud Virtual Private Server, Azure Ubuntu Virtual Private Server , etc.

“

If you want to use all the scripts (`shop` , `admin` , `api`) on the same server as this tutorial, then we recommend creating a blank ubuntu-based server with at least 2+ CPU cores and 2GB+ memory.

## Access Server

At first login your server using `SSH` and `Terminal`

## Install NodeJS & Required

# Application

## • Install NodeJS

At first, we've to install NodeJS and npm to run the pixer app. To install NodeJS and npm, run this command on your terminal,

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

## • Install Yarn

Pixer is highly dependent on **yarn**, it would be best to handle all the script parts using **yarn**. So to install yarn, use this command,

```
sudo npm i -g yarn
```

If you face any permission issue, then please check this official doc to resolve that,

### Npm Permission Issue

#### • Install Zip & Unzip

```
sudo apt install zip unzip
```

#### • Install PM2

Now we will install **PM2**, which is a process manager for Node.js applications. **PM2** provides an easy way to manage and daemonize applications (run them in the background as a service). To install **PM2** use this command,

```
sudo npm install -g pm2
```

After restarting the server or if the server crash, then pm2 will halt the process. To prevent that, we'll add pm2 as a **startup** process to run automatically after restart the server.

```
pm2 startup systemd
```

## Setup Server

### • Introduction

Nginx is one of the most popular web servers in the world. In this deployment tutorial, we're going to use Nginx to host our website. In this tutorial, we're going to use ubuntu 20.04 to host pixer

## • Step 1 - Installing Nginx

After creating the server, make sure the apt library is up to date. To update the apt library, use this command,

```
sudo apt update
```

After the update apt, we're going to install Nginx. To do that, use this command

```
sudo apt install nginx
```

## • Step 2: Adjusting the Firewall

Before testing Nginx, the firewall software

needs to be adjusted to allow access to the service. Nginx registers itself as a service with `ufw` upon installation, making it straightforward to allow Nginx access.

To check the `ufw` list, use this command,

```
sudo ufw app list
```

You will get a listing of an application list like this,

```
Available applications:  
  Nginx Full  
  Nginx HTTP  
  Nginx HTTPS  
  OpenSSH
```

At first, add ssh to the firewall,

```
sudo ufw allow ssh
```

```
sudo ufw allow OpenSSH
```

After that, to enable Nginx on the firewall,  
use this command,

```
sudo ufw allow 'Nginx HTTP'
```

Now enable the firewall,

```
sudo ufw enable  
sudo ufw default deny
```

You can verify the change by typing:

```
sudo ufw status
```

The output will be indicated which HTTP traffic

is allowed:

To	Action	From
--	-----	-----
Nginx HTTP 22/tcp	ALLOW	Anywhere
Nginx HTTP (v6) 22/tcp (v6)	ALLOW	Anywhere (v6)
	ALLOW	Anywhere (v6)

## • Step 3 – Checking your Web Server

Now check the status of the Nginx web server by using this command,

```
systemctl status nginx
```

You'll get an output like this,

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-01-12 07:35:20 UTC; 7min ago
     Docs: man:nginx(8)
 Main PID: 2940 (nginx)
    Tasks: 2 (limit: 1164)
   Memory: 5.1M
      CGroup: /system.slice/nginx.service
              └─2940 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
                  ├─2941 nginx: worker process
```

## • Step 4 – Install MySQL

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

- **Step 5 - Install PHP**

```
sudo apt install php-fpm php-mysql  
sudo apt install php-fpm php-mysql php-mbstring  
php-xml php-bcmath php-simplexml php-mbstring  
php7.4-gd php7.4-curl composer
```

- **Step 6 - Create MySQL Database & User**

```
sudo mysql
```

```
CREATE DATABASE pixer;

CREATE USER 'pickbazar_user'@'%' IDENTIFIED WITH
mysql_native_password BY 'pickbazar1';

GRANT ALL ON pixer.* TO 'pickbazar_user'@'%';

FLUSH PRIVILEGES;
```

We use MySQL user name **pickbazar\_user** and MySQL password **pickbazar1**. Make sure you change at least **MySQL** password for security.

- **Step 7 - Change permission for the **www** folder**

```
sudo chown -R $USER:$USER /var/www/
```

- **Step 8 - Upload API to Server**

At first, use this command to create a directory on `/var/www/pixer-laravel`

```
mkdir /var/www/pixer
```

Then, go to your `local computer`

- Extract the `pixer-laravel` package that you download from [CodeCanyon](#).
- Rename that folder as `redq-ecommerce`
- On that folder, you'll get another `folder` called `pixer-laravel`.
- On that folder, you'll get a folder called `api`

Now upload this `pixer-api` folder to the server `/var/www/pixer-laravel/`

## • Step 9: Setting Up Server & Project

In this chapter, we'll set up our server and also will set up Reverse Proxy to host all of our sites from the same server.

At first, we'll disable the default configuration.

```
sudo rm /etc/nginx/sites-enabled/default
```

- **Step 10 - Create New Nginx for the domain**

```
sudo touch /etc/nginx/sites-available/pixer  
sudo nano /etc/nginx/sites-available/pixer
```

Add this Nginx config file to that edited file,

```
server {
```

```
listen 80;

server_name YOUR_DOMAIN.com;

add_header X-Frame-Options "SAMEORIGIN";
add_header X-XSS-Protection "1; mode=block";
add_header X-Content-Type-Options "nosniff";

index index.html index.htm index.php;

charset utf-8;

# For API
location /backend {
    alias /var/www/pixer/pixer-api/public;
    try_files $uri $uri/ @backend;
    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME
$request_filename;
        fastcgi_pass unix:/run/php/php7.4-
fpm.sock;
    }
}
```

```
location @backend {
    rewrite /backend/(.*)$ /backend/index.php?/$1
last;
}

# For FrontEnd
location /{
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}

location /admin{
    proxy_pass http://localhost:3002/admin;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}
```

```
location = /favicon.ico { access_log off;
log_not_found off; }

location = /robots.txt { access_log off;
log_not_found off; }

error_page 404 /index.php;

location ~ \.php$ {
    fastcgi_pass unix:/var/run/php/php7.4-
fpm.sock;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
    include fastcgi_params;
}

location ~ /\.well-known.* {
    deny all;
}
```

“ Make sure you change `YOUR_DOMAIN.com` to your specific `domain name`

Special domain name

“

You can change `api` path, if you want to change the the domain path for the laravel application

“

You can change `admin` path, if you want to change the the domain path for the frontend admin

Save and close the file by typing `CTRL` and `X`, then `Y` and `ENTER` when you are finished.

Then enable the config

```
sudo ln -s /etc/nginx/sites-available/pixer  
/etc/nginx/sites-enabled/
```

Make sure you didn't introduce any syntax errors by typing:

sudo nginx -t

Next, restart Nginx:

sudo systemctl restart nginx

## Secure Server

- Step 1: Secure Nginx with Let's Encrypt

sudo apt install certbot python3-certbot-nginx

```
sudo ufw status  
  
sudo ufw allow 'Nginx Full'  
sudo ufw delete allow 'Nginx HTTP'  
  
sudo ufw status  
sudo certbot --nginx -d YOUR_DOMAIN
```

“

After this command, you'll get several command prompt. Make sure you take the necessary steps and provide information on that command prompt.

## Install API

- Step 1: Build and Run [api](#)

At first, go to the `pixer-api` folder, then copy `.env.example` to `.env`,

```
cd /var/www/pixer/pixer-api  
cp .env.example .env
```

Edit `.env`

```
nano .env
```

And add `MySQL`, `stripe`, `mail` or others configuration.

Also, add `https://YOUR_DOMAIN.COM/backend` to `APP_URL`. Without this, the `upload` function will be broken.

```
2 APP_ENV=production ←  
3 APP_KEY=  
4 APP_DEBUG=true  
5 APP_URL=https://YOUR_DOMAIN.COM/backend
```

```
5 APP_URL=https://YOUR_DOMAIN.COM/backend
6 APP_SERVICE=marvel.test
7 APP_NOTICE_DOMAIN=PICKBAZAR_
8 DUMMY_DATA_PATH=pickbazar
9
10
11 LOG_CHANNEL=stack
12 LOG_LEVEL=debug
13
```

Then install all the packages and install **api**

```
composer install
```

```
php artisan key:generate
```

```
php artisan marvel:install
```

“

You'll get several confirmations for migration, dummy data, and admin account. Make sure you check the confirmation step and take the necessary actions based on your requirement.

Enable **laravel storage**,

```
php artisan storage:link
```

Then give proper **permission** for laravel folder,

```
sudo chown -R www-data:www-data storage  
sudo chown -R www-data:www-data bootstrap/cache
```

Now, when you go to the **YOUR\_DOMAIN/backend**  
you'll get a **welcome** page like this

# FrontEnd Project Build

“

TypeScript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move the folder to the server then serve the project. We'll do the second method in this tutorial.

“

We'll suggest you build the frontend part on

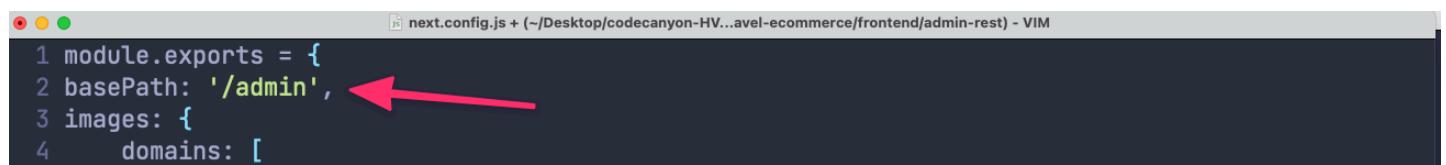
your computer and then upload the build file to the server.

Go to your `pixer-laravel` folder from your local computer .

## •Step 1 - Config Next Admin App For /admin Sub Directory

Edit `admin/next.config.js`,

add `basePath` for '/admin'



```
next.config.js + (~/Desktop/codecanyon-HV...avel-eCommerce/frontend/admin-rest) - VIM
1 module.exports = {
2   basePath: '/admin', ←
3   images: {
4     domains: [
```

```
5   "via.placeholder.com",
6   "res.cloudinary.com",
7   "s3.amazonaws.com",
8   "18.141.64.26",
9   "127.0.0.1",
10  "picsum.photos",
11  "pickbazar-sail.test",
12  "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
13 ],
14 },
15 };
```

```
INSERT > next.config.js[+]
-- INSERT --
```

javascript < utf-8[unix] 13% ≡ 2/15 ↵:20 ↵

## •Step 2 - Install & Build

go to your **pixer-laravel -> admin** folder again

To install all the npm packages run this command,

```
yarn
```

Again,

go to your `pixer-laravel -> shop` folder again

To install all the npm packages run this command,

```
yarn
```

### •Step 3 - Build the project

At first, we've to copy the sample

`.env.template` to production `.env` for the shop and admin first.

Go to,

```
cd shop
```

then use this command to copy,

```
cp .env.template .env
```

---

Now edit .env and add your API url to .env

and use

```
NEXT_PUBLIC_REST_API_ENDPOINT="https://YOUR_DOMAIN/  
backend"
```

After that, go to the admin folder,

```
cd ../admin
```

then use this command to copy,

```
cp .env.template .env
```

and use

```
NEXT_PUBLIC_REST_API_ENDPOINT="https://YOUR_DOMAIN/  
backend"
```

Then open `shop -> next.config.js` and `admin -> next.config.js`

and add your domain to `images` object

6

If your API is hosted on a subdomain, then

add that subdomain with root domain on  
`next.config.js`

## - Build Project

Now go to the `pixer-laravel -> admin` folder,  
and run,

```
yarn build
```

again,

Now go to the `pixer-laravel -> shop` folder,  
and run,

```
yarn build
```

Now zip `admin`, `shop` files and upload them to  
the server `/var/www/pixer-laravel`

Now go to the server `/var/www/pixer-laravel`  
using terminal

## Install FrontEnd And Run

Then install all the node packages go to  
`/var/www/pixer-laravel/shop` and use this command,

```
yarn
```

Then go to `/var/www/pixer-laravel/admin` and use  
this command,

```
yarn
```

- Run frontend app

For `shop` app, use this command from `pixer-laravel -> shop` folder,

```
pm2 --name shop-rest start yarn -- run start
```

And, For `admin` app, use this command from `pixer-laravel -> admin` folder,

```
pm2 --name admin-rest start yarn -- run start
```

Now go to Now, go to your `YOUR_DOMAIN` to access the shop page and `YOUR_DOMAIN/admin` for the access admin section.

cPanel

“

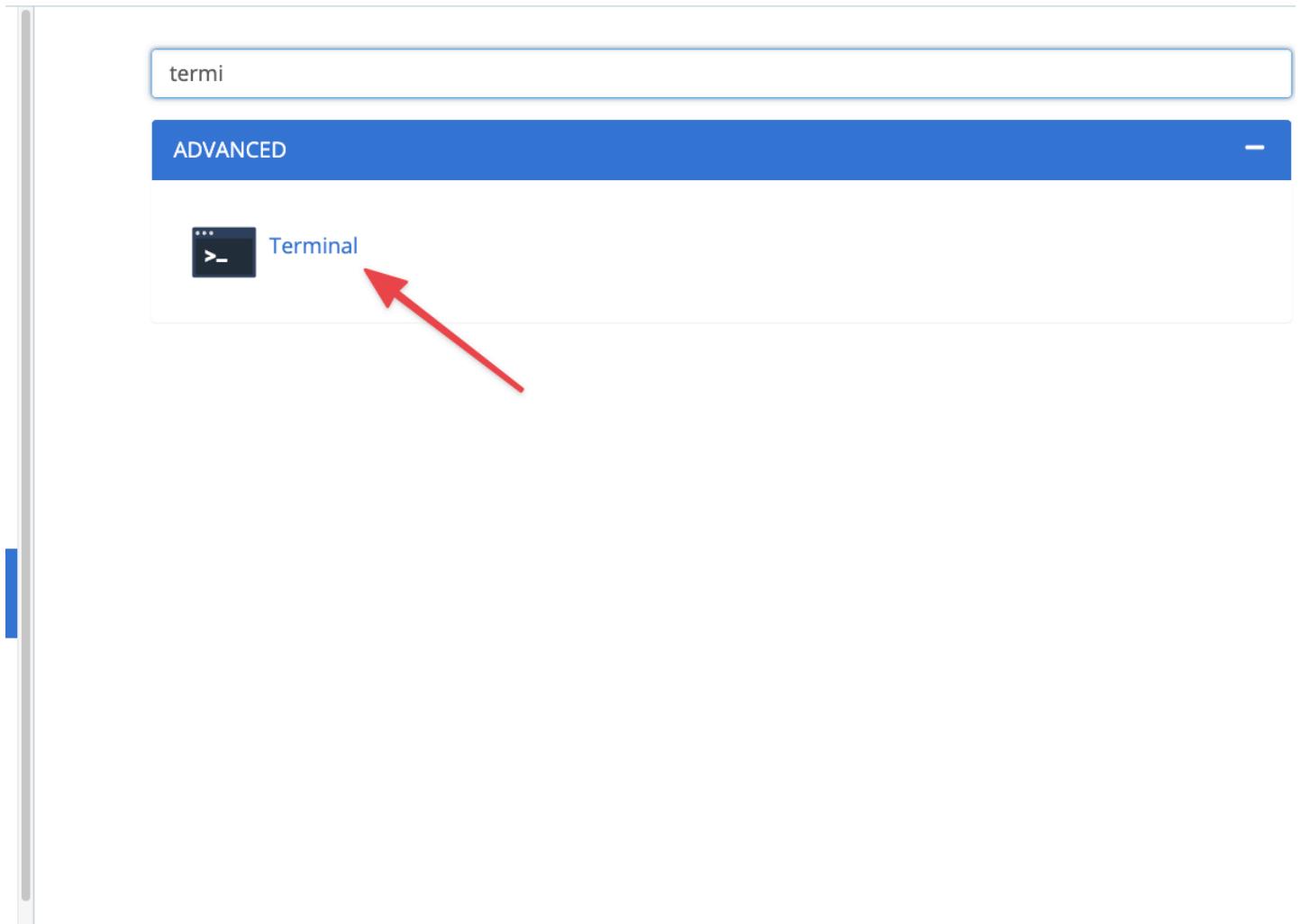
It's quite hard to debug any deployment issue on the cPanel server as the provider manages this type of server, and they've complete control of the server. And for that, We don't recommend Cpanel for deployment. We suggest you use any VPS server where you have complete control of it. you can purchase any \$5 - \$10/mo server from amazon lightsail, ec2 or digitalocean or any ubuntu server

“

If you still decide to proceed with cpanel, our support team won't be able to help you. We have put some resources for Cpanel in this documentation section to help our users to get started but other than that, we don't have much to offer with Cpanel.

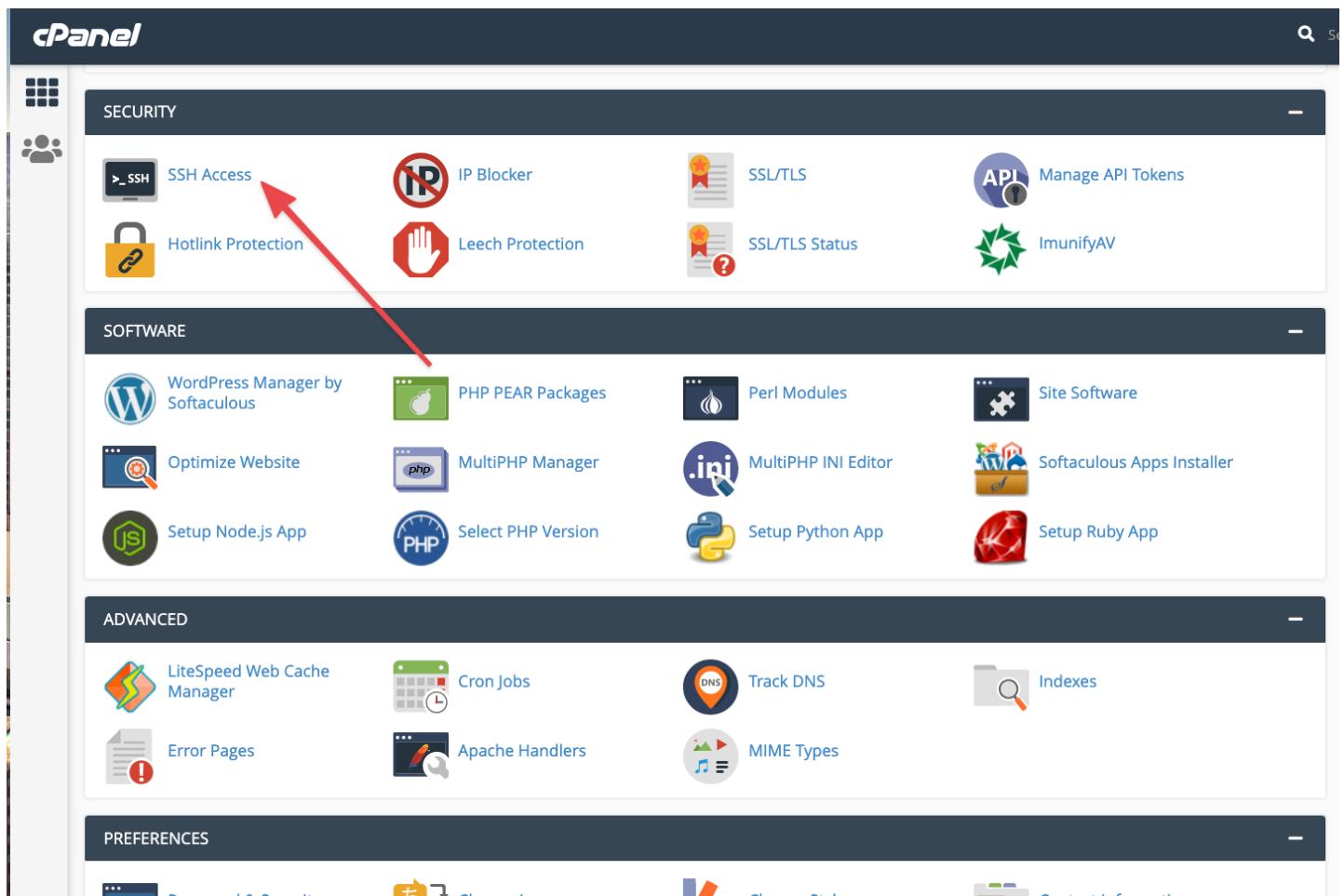
## Access Server

To install the API, access the server using the cPanel terminal first,



If you don't find the terminal, then login to your local computer terminal or [putty](#) for

# Windows using SSH.



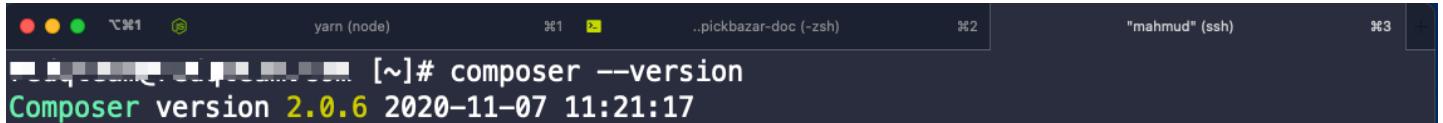
After enabling the ssh login to your server using ssh,

“

If you don't see any option, then contact your hosting provider as cPanel control by hosting provider.

After logging in, Check if the composer is already installed or not using this command,

```
composer -v
```



A screenshot of a macOS terminal window titled "yarn (node)". It shows three tabs: tab 1 is "yarn (node)", tab 2 is "..pickbazar-doc (-zsh)", and tab 3 is "'mahmud' (ssh)". The active tab (tab 3) displays the command "[~]# composer --version" followed by the output "Composer version 2.0.6 2020-11-07 11:21:17".

If composer is not installed then, install **composer** to your server.

Check this [YouTube Video](#) for install **composer** on your server,

After that, check the PHP version using,

```
php -v
```

make sure it's **7.4**

## Create Subdomains

Now create two subdomains, for example,

```
-> your_domain.com -> host frontend store.  
-> api.your_domain.com -> host laravel API.  
-> admin.your_domain.com -> host admin dashboard.
```

Or if you want to host all the script on subdomains, then create subdomains like this,

```
-> store.your_domain.com -> host frontend store.  
-> api.your_domain.com -> host laravel API.  
-> admin.your_domain.com -> host admin dashboard.
```

“

After creating domain/subdomains, make sure all the domain/subdomains are HTTPS enabled

all the security functions are now enabled.

Please contact your hosting provider to enable this, as most hosting providers provide some sort of free SSL.

## Install API

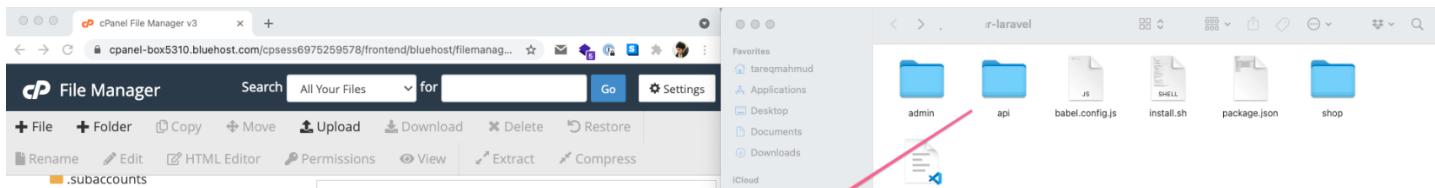
Extract the `pixer-laravel` package that you download from [CodeCanyon](#).

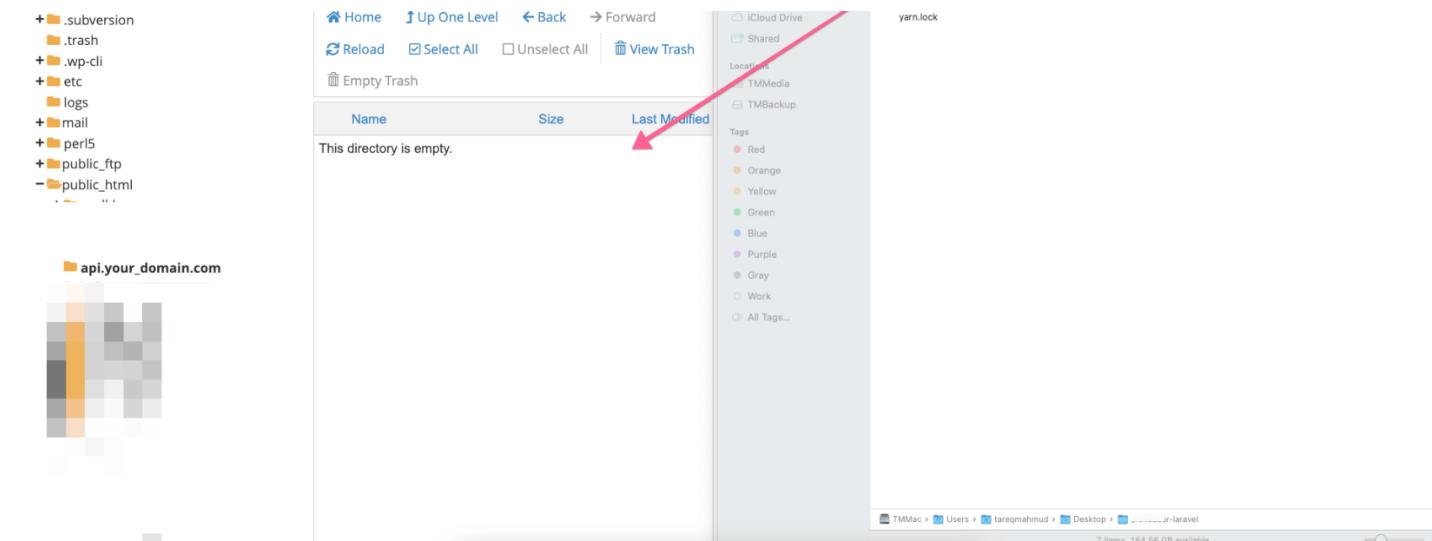
On that folder, you'll get another `zip` called `pixer-laravel.zip`.

Now extract this `pixer-laravel.zip` file.

On that file, you'll get a folder called `pixer-api`

Now upload this `pixer-api` folder to the `api.your_domain.com` folder in your server





“  
Make sure your `api.your_domain.com` subdomain Document Root points to that `api/public` folder.

**cPanel**

## Subdomains

A subdomain is a subsection of your website that can exist as a new website without a new domain name. Use subdomains to create memorable URLs for different content areas of your site. For example, you can create a subdomain for your blog that is accessible through `blog.example.com` and `www.example.com/blog`

### Create a Subdomain

Subdomain

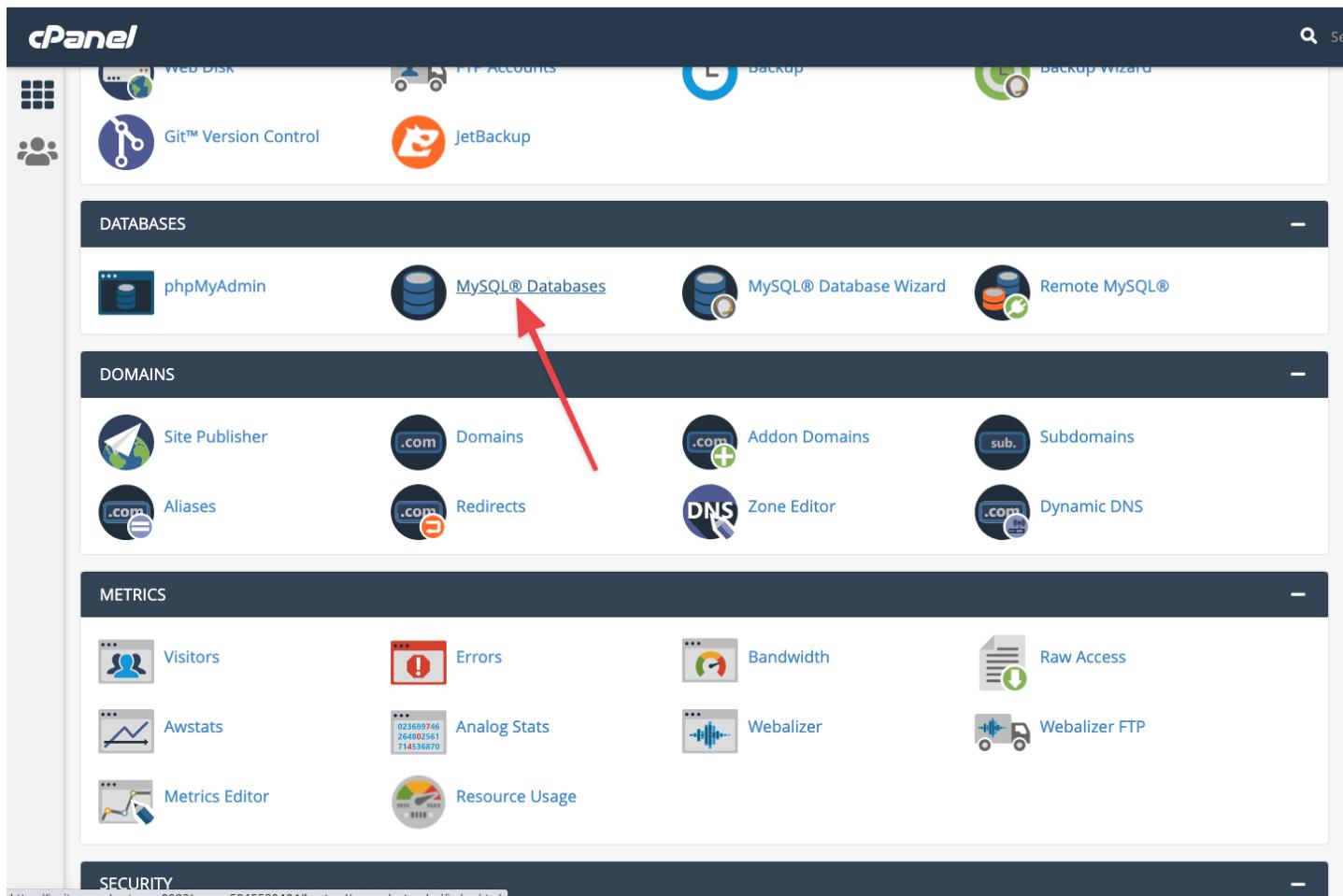
Domain

Document Root

**Create**

### Modify a Subdomain

Now create a MySQL database and user from MySQL wizard



After creating the MySQL database, go to your **api** folder from your cPanel file manager and copy **.env.example** to **.env**.

File Manager				
Actions				
Name	Size	Last Modified	Type	Permissions
app	82 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
bootstrap	34 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
config	200 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775

	Size	Last Modified	Type	Owner
config	300 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
database	74 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
packages	95 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
public	106 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
resources	52 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
routes	75 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
storage	46 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
tests	83 bytes	Apr 4, 2021, 4:53 PM	httpd/unix-directory	0775
.editorconfig	220 bytes	Apr 4, 2021, 4:53 PM	text/x-generic	0664
.env	871 bytes	Today, 9:33 PM	text/x-generic	0664
.env.example	871 bytes	Apr 4, 2021, 4:53 PM	text/x-generic	0664
.gitattributes	111 bytes	Apr 4, 2021, 4:53 PM	text/x-generic	0664
.gitignore	216 bytes	Apr 4, 2021, 4:53 PM	text/x-generic	0664
.styleci.yml	181 bytes	Apr 4, 2021, 4:53 PM	text/x-generic	0664
artisan	1.65 KB	Apr 4, 2021, 4:53 PM	text/x-generic	0755
composer.json	2.38 KB	Apr 4, 2021, 4:53 PM	text/x-generic	0664
composer.lock	390.9 KB	Apr 4, 2021, 4:53 PM	text/x-generic	0664
docker-compose.yml	1.23 KB	Apr 4, 2021, 4:53 PM	text/x-generic	0664

After the copy, edit `.env` and add MySQL credentials,

Editing: `sur_domain.com/api/.env` Encoding: utf-8 Re-open Use legacy editor Save Changes Close

```

1 APP_NAME=Pickbazar
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6 APP_SERVICE=
7
8 LOG_CHANNEL=stack
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=localhost
13 DB_PORT=3306
14 DB_DATABASE=ADD_MYSQL_DATABASE_NAME
15 DB_USERNAME=ADD_MYSQL_USERNAME
16 DB_PASSWORD=ADD_MYSQL_PASSWORD
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 QUEUE_CONNECTION=sync
21 SESSION_DRIVER=file
22 SESSION_LIFETIME=120
23
24 MEMCACHED_HOST=memcached
25
26 REDIS_HOST=redis
27 REDIS_PASSWORD=null
28 REDIS_PORT=6379
29
30 MAIL_MAILER=mailgun
31 MAILGUN_DOMAIN=
32 MAILGUN_SECRET=
33 MAIL_FROM_ADDRESS=
34 MAIL_PORT=1025
35 MAIL_USERNAME=null
36 MAIL_PASSWORD=null
37 MAIL_ENCRYPTION=null
38 MAIL_FROM_ADDRESS=support@example.io
39 MAIL_FROM_NAME="${APP_NAME}"
40
41 AWS_ACCESS_KEY_ID=

```

Also, add `https://YOUR_DOMAIN.COM/api` to `APP_URL`.  
Without this, the `upload` function will be

broken.

```
2 APP_ENV=production ←  
3 APP_KEY=  
4 APP_DEBUG=true  
5 APP_URL=https://YOUR_DOMAIN.COM/backend[] ←  
6 APP_SERVICE=marvel.test  
7 APP_NOTICE_DOMAIN=PICKBAZAR_  
8 DUMMY_DATA_PATH=pickbazar  
9  
10  
11 LOG_CHANNEL=stack  
12 LOG_LEVEL=debug  
13
```

Then go to your **ssh terminal** again and,

go to **api** folder and run,

```
composer install
```

If **composer** installs all the packages successfully, then run this command on the **api** folder,

```
php artisan key:generate
```

```
php artisan marvel:install
```

“

You'll get several confirmations for migration, dummy data, and admin account. Make sure you check the confirmation step and take the necessary actions based on your requirement.

After that, run this command to link storage,

```
php artisan storage:link
```

After install, go to your [api.your\\_domain\\_name.com](http://api.your_domain_name.com), and you'll get a webpage like this,

# Install FrontEnd

Before proceeding next step, make sure you already create two subdomains like this,

```
-> your_domain.com -> host frontend store.  
-> admin.your_domain.com -> host admin dashboard.
```

OR

```
-> store.your_domain.com -> host frontend store.  
-> admin.your_domain.com -> host admin dashboard.
```

## •FrontEnd Project Build

“

TypeScript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move the folder to the server then serve the project. We'll do the second method in this tutorial.

“

We'll suggest you build the frontend part on your computer and then upload the build file to the server.

## •step 1 - Build Custom Server

go to your **pixer-laravel** folder

## - shop rest

Create custom server for **shop rest**,

```
nano shop/server.js
```

and paste this code,

```
// server.js

const { createServer } = require('http')
const { parse } = require('url')
const next = require('next')

const dev = process.env.NODE_ENV !== 'production'
const app = next({ dev })
const handle = app.getRequestHandler()

app.prepare().then(() => {
  createServer((req, res) => {
    // Be sure to pass `true` as the second
    // argument to `next`
    next(req, res, true)
  })
})
```

```
// Be sure to pass true as the second argument to `url.parse`.  
  
// This tells it to parse the query portion of the URL.  
  
const parsedUrl = parse(req.url, true)  
const { pathname, query } = parsedUrl  
  
if (pathname === '/a') {  
  app.render(req, res, '/a', query)  
} else if (pathname === '/b') {  
  app.render(req, res, '/b', query)  
} else {  
  handle(req, res, parsedUrl)  
}  
}).listen(3003, (err) => {  
  if (err) throw err  
  console.log('> Ready on http://localhost:3003')  
})  
})
```

Now update package.json for **shop rest**,

```
nano shop/package.json
```

and replace `start` script with this,

```
"start": "NODE_ENV=production node server.js"
```

```
package.json (-/Desktop/ar-laravel/shop) - VIM
1 [
2   "name": "@pick-bazar/shop",
3   "version": "4.2.0",
4   "private": true,
5   "scripts": {
6     "clean": "rimraf \".{next,node_modules,__generated__,.cache,src/framework/graphql/**/*.d.ts}\\"",
7     "dev:rest": "next dev -p 3003",
8     "build:rest": "next build",
9     "codegen": "node -r dotenv/config $(yarn bin)/graphql-let",
10    "dev:gql": "yarn codegen && next dev -p 3001",
11    "build:gql": "yarn codegen && next build",
12    "start": "next start", →
13    "lint": "next lint"
14  },
15  "dependencies": {
16    "@apollo/client": "^3.4.16",
17    "@headlessui/react": "^1.4.1",
18    "@hookform/resolvers": "^2.8.2",
19    "@reach/portal": "^0.16.0",
20    "@stripe/react-stripe-js": "^1.5.0",
21    "@stripe/stripe-js": "^1.20.2",
22    "apollo-upload-client": "^16.0.0",
23    "axios": "^0.23.0",
24    "body-scroll-lock": "^4.0.0-beta.0",
25    "camelcase-keys": "^7.0.1",
26    "classnames": "^2.3.1",
27    "dayjs": "^1.10.7",
28    "deepmerge": "^4.2.2",
29    "framer-motion": "^4.1.17",
30  }
31  > package.json      js... << 1% N:1/87 ≡ 21 I... > package.json[+]
32  -- INSERT --      js... << 13% N:12/87 ≡ 51
```

## - admin rest

Similarly, create custom server for `admin rest`,

```
nano admin/server.js
```

and paste this code,

```
// server.js

const { createServer } = require('http')
const { parse } = require('url')
const next = require('next')

const dev = process.env.NODE_ENV !== 'production'
const app = next({ dev })
const handle = app.getRequestHandler()

app.prepare().then(() => {
  createServer((req, res) => {
    // Be sure to pass `true` as the second
    argument to `url.parse`.

    // This tells it to parse the query portion of
    the URL.

    const parsedUrl = parse(req.url, true)
    const { pathname, query } = parsedUrl

    if (pathname === '/a') {
      app.render(req, res, '/a', query)
    } else if (pathname === '/b') {

```

```
        , (req, res) => {
          app.render(req, res, '/b', query)
        } else {
          handle(req, res, parsedUrl)
        }
      }).listen(3002, (err) => {
      if (err) throw err
      console.log('> Ready on http://localhost:3002')
    })
  })
})
```

Now update package.json for **admin rest**,

```
nano admin/package.json
```

and replace **start** script with this,

```
"start": "NODE_ENV=production node server.js"
```

## • Step 2 - Install & Build

go to your `pixer-laravel -> admin` folder again

To install all the npm packages run this command,

```
yarn
```

Again,

go to your `pixer-laravel -> shop` folder again

To install all the npm packages run this command,

```
yarn
```

## • Step 3 - Build the project

At first, we've to copy the sample `.env.template` to production `.env` for the shop and admin first.

Go to,

```
cd shop
```

then use this command to copy,

```
cp .env.template .env
```

Now edit `.env` and add your API url to `.env`

```
nano .env
```

and use

```
NEXT_PUBLIC_REST_API_ENDPOINT=https://api.YOUR_DOMAIN.com/
```

After that, go to the `admin -> rest` folder,

```
cd ../admin
```

then use this command to copy,

```
cp .env.template .env  
nano .env
```

and use

```
NEXT_PUBLIC_REST_API_ENDPOINT=https://api.YOUR_DOMAIN.com/
```

go to your `pixer-laravel -> admin` folder again

To install all the npm packages run this command,

```
yarn build
```

Again,

go to your `pixer-laravel -> shop` folder again

To install all the npm packages run this command,

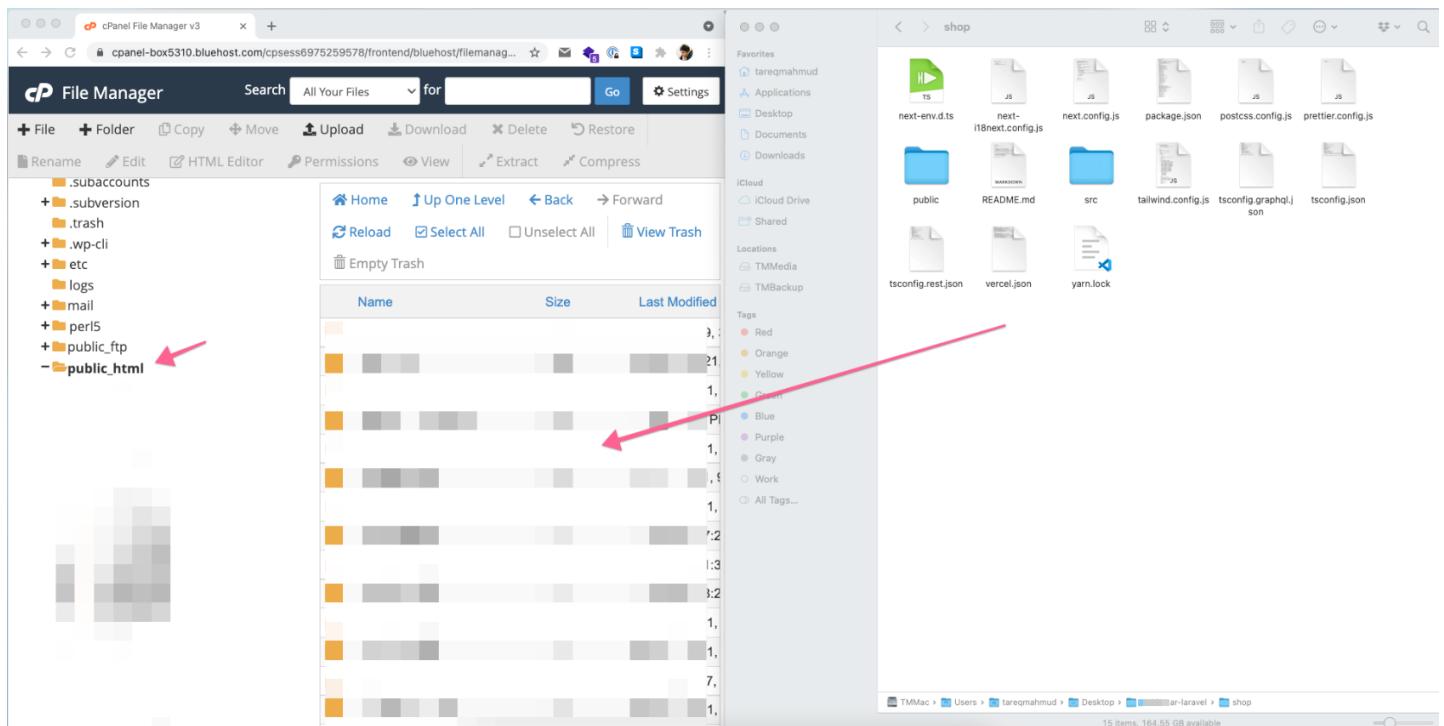
```
yarn build
```

and run,

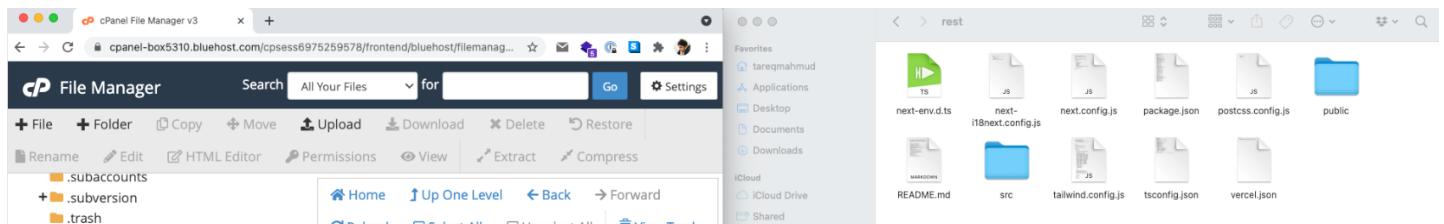
After build the project upload the

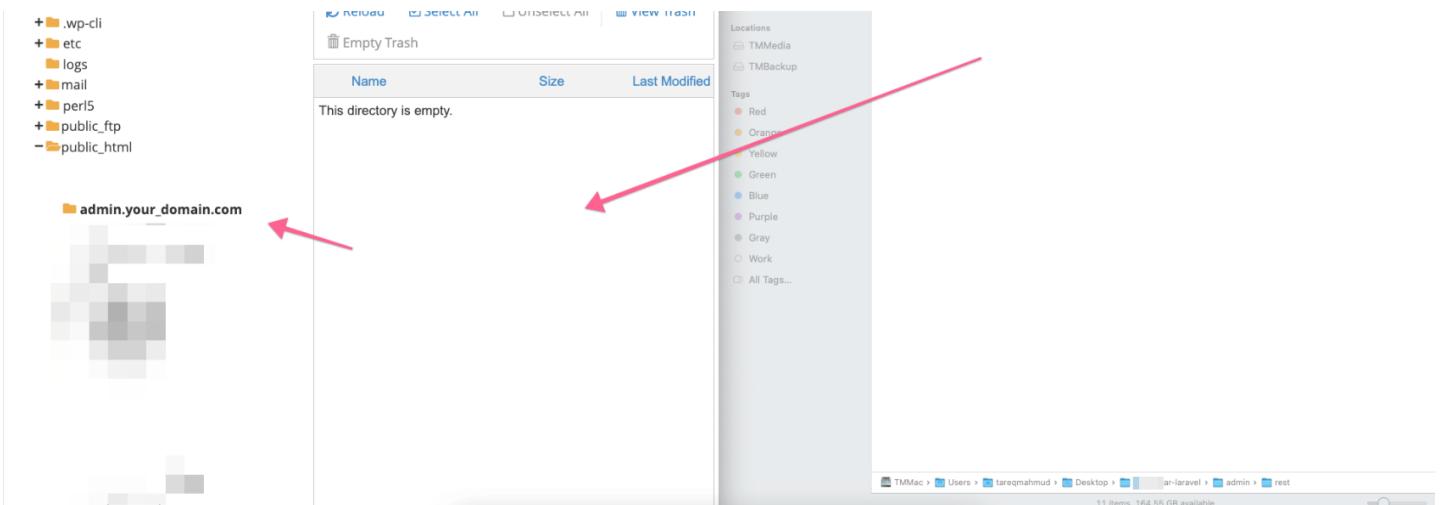
- shop to root\_domain -> public\_html folder
- admin-rest to admin.your\_domain.com folder

shop ,



shop-admin ,





# Install NodeJs Project

We'll run both `shop` and `admin` using the cPanel NodeJs application in this step.

To do that at first go to the NodeJS section from your cPanel,

For `shop`,



The screenshot shows the Softaculous interface with three main sections: SOFTWARE, ADVANCED, and PREFERENCES. In the SOFTWARE section, there are eight items: WordPress Manager by Softaculous, PHP PEAR Packages, Perl Modules, Site Software, Optimize Website, MultiPHP Manager, MultiPHP INI Editor, Softaculous Apps Installer, Setup Node.js App (highlighted with a red arrow), Select PHP Version, Setup Python App, and Setup Ruby App.

The screenshot shows the cPanel interface with a navigation bar at the top. Below the bar, there are two main sections: WEB APPLICATIONS and CREATE APPLICATION. The WEB APPLICATIONS section shows a single item: Node.js. The CREATE APPLICATION section has a blue button labeled '+ CREATE APPLICATION'. A red arrow points from the 'Node.js' icon in the WEB APPLICATIONS section towards the '+ CREATE APPLICATION' button.

Now,

- Select NodeJS version
- Make environment **production.**

- Set Application Root
- And application startup file as `server.js`

The screenshot shows the cPanel Web Applications interface. A red arrow points to the "Node.js version" dropdown set to "12.22.1". Another red arrow points to the "Application mode" dropdown set to "Production". A third red arrow points to the "Application root" input field containing "/home/tareqma3/public\_html". A fourth red arrow points to the "Application URL" input field, which has ".com" selected from a dropdown menu. A fifth red arrow points to the "Application startup file" input field containing "server.js". A sixth red arrow points to the "Passenger log file" input field containing "/home/tareqma3/". At the bottom, there is a section for "Environment variables" with a "+ ADD VARIABLE" button.

You can get the Application Path from your cPanel file manager

File Folder Copy Move Upload Download Delete Restore Rename Edit HTML Editor Permissions View Extract Compress

Home Up One Level Back Forward Reload Select All Unselect All View Trash Empty Trash

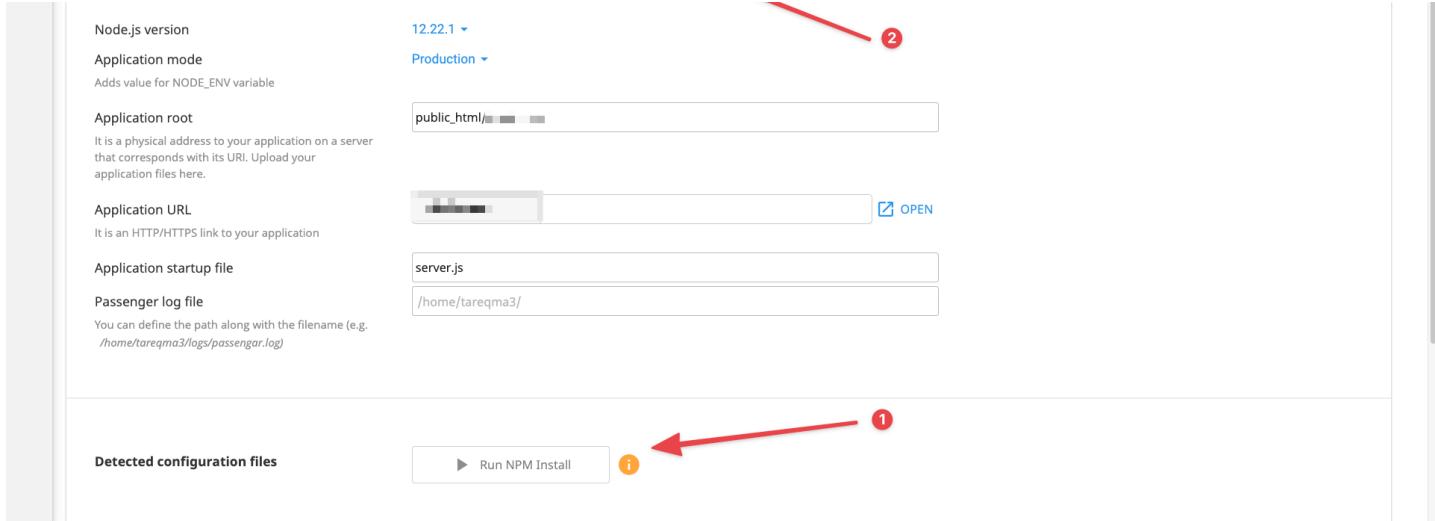
Name	Size	Last Modified	Type	Permissions
.cagefs	39 bytes	Feb 12, 2021, 2:10 PM	httpd/unix-directory	0771
.cl.selector	48 bytes	May 13, 2021, 1:44 PM	httpd/unix-directory	0755
.cpanel	195 bytes	Today, 5:16 PM	httpd/unix-directory	0700
.cphorde	70 bytes	Mar 24, 2021, 9:28 PM	httpd/unix-directory	0700
.gnupg	79 bytes	Feb 13, 2021, 10:31 PM	httpd/unix-directory	0700
.htpasswd	6 bytes	Jan 23, 2021, 5:05 PM	httpd/unix-directory	0750
.razor	231 bytes	Feb 23, 2021, 10:19 PM	httpd/unix-directory	0755
.softaculous	78 bytes	Jan 23, 2021, 5:06 PM	httpd/unix-directory	0711
.spamassassin	24 bytes	Jan 23, 2021, 5:05 PM	httpd/unix-directory	0700
.subaccounts	28 bytes	Mar 24, 2021, 9:28 PM	httpd/unix-directory	0700
.trash	217 bytes	Feb 13, 2021, 2:09 AM	httpd/unix-directory	0700
etc	80 bytes	Today, 3:05 PM	httpd/unix-directory	0750
logs	243 bytes	Yesterday, 6:28 PM	httpd/unix-directory	0700
Iscache	177 bytes	Feb 14, 2021, 2:24 AM	httpd/unix-directory	2770
IscmData	79 bytes	Feb 12, 2021, 7:02 PM	httpd/unix-directory	0700
mail	239 bytes	Mar 24, 2021, 9:28 PM	mail	0751
public_ftp	22 bytes	Jan 23, 2021, 5:05 PM	publicftp	0750
public_html	162 bytes	Today, 3:55 PM	publichtml	0750
ssl	77 bytes	May 12, 2021, 1:20 PM	httpd/unix-directory	0755

After create NodeJS app, install all the packages and restart the app,

cPanel

WEB APPLICATIONS STOP APP RESTART DESTROY CANCEL SAVE

Node.js



For **admin**,

Similarly, create a another NodeJS application for **admin** with **admin subdomain** and **admin subdirectory**

“

After installing and run both NodeJS application, you can access your domain to check Pixer,

# Vercel

“

[your-frontend-project] = `admin` or `shop`

- vercel.com

If you want to host the template in vercel.com  
then follow the below command

- **Frontend**

- **Step 1:**

- After deploying the api you will get the api endpoint url. Put that url in the `[your-frontend-project]/.env` and `vercel.json` file.

```
NEXT_PUBLIC_REST_API_ENDPOINT=
'{put_your_api_url_here}'
```

Also, add others environment variables.

After that, run this command from `shop` and `admin -> rest.` folder

```
vercel
```

```
laravel-e-commerce/frontend/shop-rest  master ✘
▶ vercel --prod
Vercel CLI 21.3.3
🔗 Inspect: https://vercel.com/redq/shop-pickbazar-test/GCTaetjyreC1BJ9tmEUgafRsUYaX [4s]
✓ Production: https://shop-pickbazar-rest.vercel.app [copied to clipboard] [3m]
```

\*\* NOTE: \*\* for deploying to `vercel` you need to install `vercel-cli` on your machine for more information please visit [here](#)

- **For others hosting providers:**

Please follow [nextis deployment docs](#):

# Pixer Update - Virutal Private Server

“

If you follow this [Virtual Private Server](#) docs to host your site, then follow this documentation to update Pixer to a new version.

To build the frontend you've to update the [API](#) first. But before that this time, we'll use git and GitHub to make upload and download

relatively easy.

## Step 1: Setup Git - Server

“

This step is only for first update. From second update start from [Step 2](#)

At first, we've to install git on our server and config it.

- **Install git**

```
sudo apt install git
```

- **Config for first time**

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

Make sure you change `you@example.com` and `Your Name` with your `email` and `name`.

## • Prepare Git Repository

At first, go to your `pixer` directory on your server,

```
cd /var/www/pixer
```

Then initialize a git on that folder,

```
git init
```

Create a new `.gitignore` file

```
nano .gitignore
```

and paste this code to that `.gitignore`,

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
node_modules
/.pnp
.pnp.js

# testing
/coverage
# *~
*.swp
tmp/

# misc
.DS_Store

# If Packages have their independent repo
```

```
# /packages

# ignore log files
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.idea/
.vscode/
node_modules/
.DS_Store
*.tgz
my-app*
lerna-debug.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
/.changelog
.npm/
packages/server/node_modules
packages/server/error.log
packages/server/debug.log

packages/web/.env
packages/reusecore
packages/cloud
```

```
.docz  
.now  
.vercel
```

After that, `save` that file and use this command for the initial commit,

```
git init  
git add .  
git commit -m "Initial commit"
```

Create a separate branch to maintain the updated code.

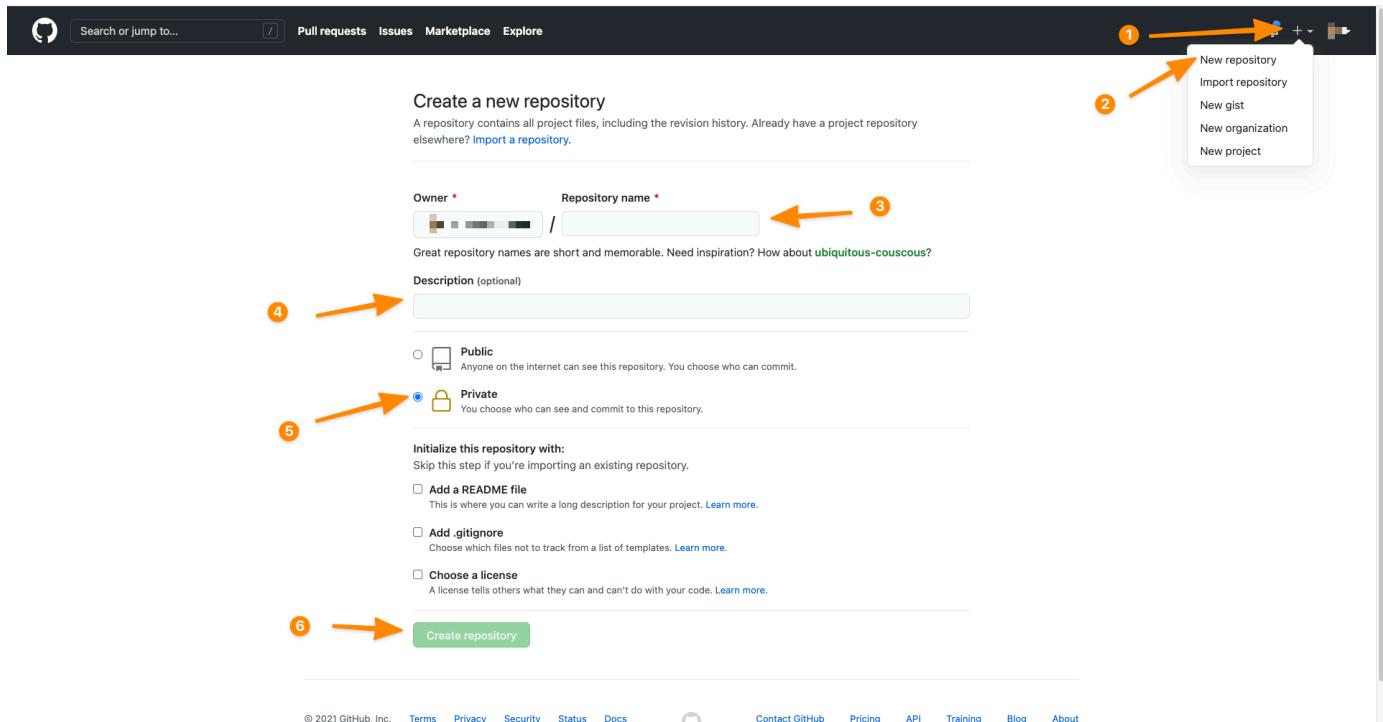
```
git branch pixer
```

## • Git & Github

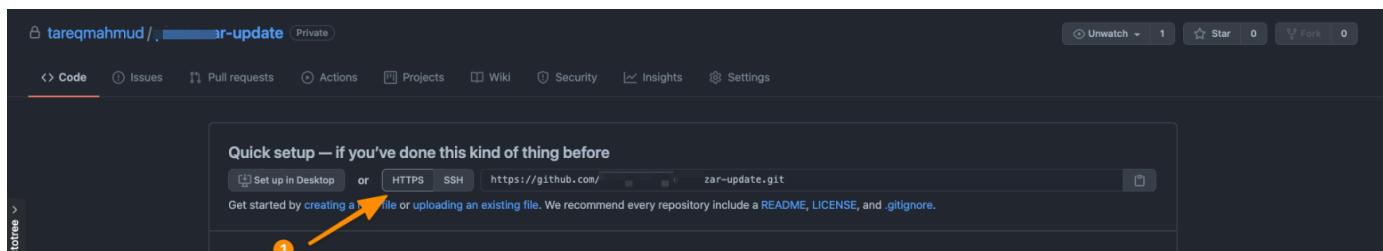
At first, go to <https://github.com/> and create

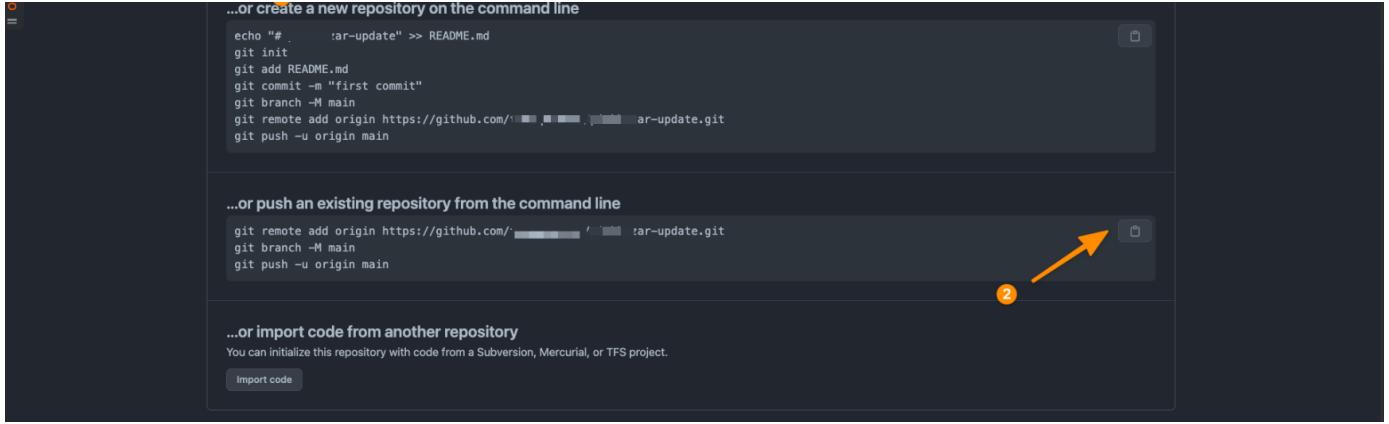
an account first. If you already have an account then **Sign In** on that account.

Then create a new repository,



After creating the repository you'll get a page like this and from this page copy the **second command block** and go to your **server** using **SSH & terminal**,





```
...or create a new repository on the command line
echo "# .tar-update" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/████████████████/ar-update.git
git push -u origin main

...or push an existing repository from the command line
git remote add origin https://github.com/████████████████/ar-update.git
git branch -M main
git push -u origin main

...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code
```

And **paste** that copied command to **pickabazar** folder and press **enter**

It'll ask your GitHub **username** and **password**, provide your GitHub **username** and **password**



```
root@redq:/var/www/pickbazar# git remote add origin https://github.com/████████████████/ar-update.git
fatal: remote origin already exists.
root@redq:/var/www/pickbazar# git branch -M main
root@redq:/var/www/pickbazar# git push -u origin main
Username for 'https://github.com': |
```



Your existing repository is successfully connected with GitHub.

## Step 2: Shut Down Current Process

At first use this command to shut down all the applications for update,

```
pm2 stop 'all'
```

# Step 3: Local Repository & Updated Code

In this step,

Clone that GitHub repository to your local computer

Download update package from CodeCanyon

Open your terminal and clone that repository to your computer

Checkout to pixer branch

```
git checkout -b pixer
```

replace repository file with downloaded pixer-laravel folder content.

Then use this command to add all files to git

```
git add
```

```
git add .  
git commit -m "Update Pixer API"
```

Merge new code with **main** branch

```
git checkout -b main  
git merge pixer
```

“

In this step, you'll face a merge conflict issue. Make sure you resolve all the conflicts to maintain your customization with the updated code. You can check this [video](#) about resolve merge conflict.

After **resolve** and **commit**, push the code to GitHub.

```
git push origin main
```

# Step 4: Update API

In this step, go to your server terminal and go to `/var/www/pixer` directory and pull all updated code,

```
git pull origin main
```

After pull go to `api` folder,

```
cd api
```

and install composer package and optimize compiled file,

```
composer install  
php artisan optimize
```

With that updated API will be installed. To check go to your `YOUR_API_DOMAIN/products`

## Step 5: FrontEnd Project Build

TypeScript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move the folder to the server then serve the project. We'll do the second method in this tutorial.

“

We'll suggest you build the frontend part on your computer and then move the build file to ... . . . . .

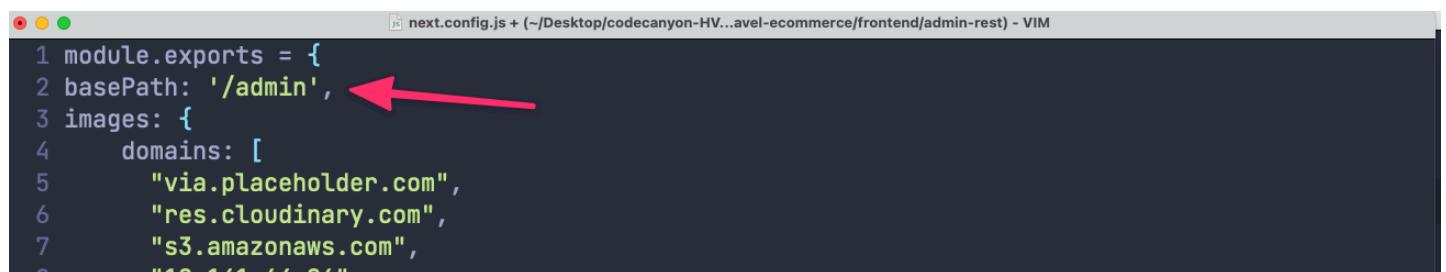
the server using git and github.

Go to your local git repository using terminal.

- Config Next Admin App For /admin Sub Directory

Edit admin/next.config.js ,

add basePath for '/admin'



```
1 module.exports = {
2   basePath: '/admin', ←
3   images: {
4     domains: [
5       "via.placeholder.com",
6       "res.cloudinary.com",
7       "s3.amazonaws.com",
8       "12.161.66.96"
9     ]
10  }
```

```
8   "18.141.64.26",
9   "127.0.0.1",
10  "picsum.photos",
11  "pickbazar-sail.test",
12  "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
13  ],
14 },
15 };
```

```
INSERT >> next.config.js[+]
-- INSERT --
```

javascript < utf-8[unix] < 13% ≡ 2/15 N:20

## • Install & Build

go to your **pixer-laravel -> admin** folder again

To install all the npm packages run this command,

```
yarn
```

Again,

go to your **pixer-laravel -> shop** folder again

To install all the npm packages run this command,

```
yarn
```

- **Build the project**

go to your `pixer-laravel -> admin` folder again

To install all the npm packages run this command,

```
yarn build
```

Again,

go to your `pixer-laravel -> shop` folder again

To install all the npm packages run this command,

```
yarn build
```

- **Upload to GitHub**

Use this command to add all files to git,

```
git add .  
git commit -m "Build frontend"  
git push origin main
```

## Step 6: Upload Frontend & Run

At first go to your server `pixer` or `git` folder and use this command to pull all the build file,

```
git pull origin main
```

Then install all the node packages,  
go to your `pixer-laravel -> admin` folder again

To install all the npm packages run this command,

```
yarn
```

Again,

go to your `pixer-laravel -> shop` folder again

To install all the npm packages run this command,

```
yarn
```

- **Run frontend app**

Use this command to run frontend app as `PM2`

again.

```
pm2 start 'all'
```

Now go to Now, go to your `YOUR_DOMAIN` to access the shop page and `YOUR_DOMAIN/admin` for the access admin section.

## # SEO

For SEO, we use `Next SEO` packages, and we provide two boilerplates at

```
```bash
```

```
shop → src → components → seo
```

```
```
```

With that boilerplate, check [Next SEO](#) docs to use SEO on your pixer site,

<https://github.com/garmeeh/next-seo>

## Analytics

---

We have not used any analytics integration in this template yet. But, you can easily integrate any

analytics using [Next JS examples](#).

# Introduction

Pixer is a laravel multi api package for ecommerce. This package is for building dynamic ecommerce site using pixer package with rest.

# Getting Started

For getting started with the template you have to follow the below procedure. For quick guide you can check below videos for installation.

## • Prerequisites

- PHP 7.4
- Composer
- Xamp/Wamp/Lamp for any such application for  
apache    nginx    mysql

• PHP's built-in functions

- PHP plugins you must need

- simplexml
- PHP's dom extension
- mbstring
- GD Library

- Resources you might need

<https://laravel.com/docs/8.x>

<https://lighthouse-php.com/5/getting-started/installation.html>

<https://github.com/spatie/laravel-medialibrary>

<https://github.com/andersao/l5-repository>

<https://spatie.be/docs/laravel-permission/v3/introduction>

- Packages we have used

```
"nuwave/lighthouse": "^4.18.0",
```

```
"laravel/legacy-factories": "^1",  
"cviebrock/eloquent-sluggable": "^8.0",  
"laravel/sanctum": "^2.7",  
"prettus/l5-repository": "^2.6",  
"spatie/laravel-medialibrary": "^9.4.0",  
"spatie/laravel-permission": "^3.11",  
"php-http/guzzle7-adapter": "^0.1.1",  
"bensampo/laravel-enum": "^3.1.0",  
"league/flysystem-aws-s3-v3": "~1.0"
```

## Installation

- Make sure you have run xamp/mamp/wamp/lamp for mysql and php
  - Create a database in your mysql and put those info in next step
  - Rename .env.example file to .env and provide necessary credentials. Like database credentials stripe credentials, s3

credentials(only if you use s3 disk) admin  
email shop url etc.

Specially check for this `env` variables

```
DB_HOST=localhost  
DB_DATABASE=pickbazar_laravel  
DB_USERNAME=root  
DB_PASSWORD=
```

- Run `composer install`

```
▶ composer install  
Installing dependencies from lock file (including require-dev)  
Verifying lock file contents can be installed on current platform.
```

```
Generating optimized autoload files  
> Illuminate\Foundation\ComposerScripts::postAutoloadDump  
> @php artisan package:discover --ansi  
Discovered Package: barryvdh/laravel-dompdf  
Discovered Package: bensampo/laravel-enum  
Discovered Package: cviebrock/eloquent-sluggable  
Discovered Package: facade/ignition  
Discovered Package: fideloper/proxy  
Discovered Package: fruitcake/laravel-cors  
Discovered Package: ignited/laravel-omnipay
```

```
Discovered Package: intervention/image
Discovered Package: laravel/legacy-factories
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: mll-lab/laravel-graphql-playground
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: nuwave/lighthouse
Discovered Package: pickbazar/shop
Discovered Package: prettus/l5-repository
Discovered Package: spatie/laravel-medialibrary
Discovered Package: spatie/laravel-permission
Package manifest generated successfully.
95 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

- run `php artisan key:generate`

```
▶ php artisan key:generate
Application key set successfully.
```

- Run `php artisan marvel:install` and follow necessary steps.

```
Installing Pickbazar Dependencies...
Do you want to migrate Tables? If you have already run this command or migrated tables then be aware, it will erase all of your data. (yes/no) [no]:
> yes
Migrating Tables Now....
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (79.70ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (58.57ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (79.56ms)
```

```
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (143.06ms)
Migrating: 2020_04_17_194830_create_permission_tables
Migrated: 2020_04_17_194830_create_permission_tables (839.79ms)
Migrating: 2020_06_02_051901_create_pickbazar_tables
Migrated: 2020_06_02_051901_create_pickbazar_tables (1,601.56ms)
Migrating: 2020_10_26_163529_create_media_table
Migrated: 2020_10_26_163529_create_media_table (71.72ms)
Tables Migration completed.

Do you want to seed dummy data? (yes/no) [no]:
> yes

Copying necessary files for seeding....
File copying successful
Seeding....
Seed completed successfully!

Do you want to create an admin? (yes/no) [no]:
> no

Copying resources files...
Installation Complete
```

- For image upload to work properly you need to run `php artisan storage:link`.

```
The [/var/www/html/public/storage] link has been connected to [/var/www/html/storage/app/public].
The links have been created.
```

- run `php artisan serve`

```
▶ php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sat Apr 10 16:35:26 2021] PHP 8.0.0 Development Server (http://127.0.0.1:8000) started
```

“

NB: You must need to run `php artisan marvel:install` to finish the installation. Otherwise your api will not work properly. Run the command and follow the necessary steps.

“

NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost:8000/`

## • For MAC and Linux (with sail and docker)

There is an alternate installation procedure for linux and mac. You can follow below procedure to getting started with **sail**

### - Prerequisites

- Docker

### - Installation Mac

### - Video

### - REST API

- Run Docker application first
- Now go to your nixer-laravel root directory

and run `bash install.sh`. It will guide you through some process. Follow those steps carefully and your app will be up and running

- Navigate to `api` then `sail down` to stop the container. If you want to remove the volumes then `sail down -v`

“

NB: your frontend `NEXT_PUBLIC_REST_API_ENDPOINT` env value will be `localhost/`

## Configuration

All the configurations files are in `packages/marvel/config` folder. You can change any necessary configuration from these files.

You can also publish the shop configuration using `artisan vendor:publish --provider="Marvel\ShopServiceProvider" --tag="config"` command in your root folder.

- Create .env file from our example.env file and put necessary configuration
- By default s3 is using for media storage but you can also use local folder. Change `MEDIA_DISK` IN .env file as your need. Supported options are `public` and '`s3`'
- Set Payment related configuration to `STRIPE_API_KEY` .env variable
- Set `ADMIN_EMAIL` , `SHOP_URL` and necessary Database credentials.

## Console Commands

- `php artisan marvel:install complete`

installation with necessary steps

- `php artisan marvel:seed` seeding demo data
- `php artisan marvel:copy-files` copy necessary files
- `php artisan vendor:publish --provider="Marvel\ShopServiceProvider" --tag="config"` published the configuration file

All of the above custom command you will find in `packages/marvel/src/Console` folder.

## Development

All the rest routes is resides in

`packages/marvel/src/Rest/Routes.php` file and you can easily navigate to corresponding controller

---

and necessary files.

- **Endpoints**

[https://documenter.getpostman.com/view/11693148/  
UVC5Fo9R](https://documenter.getpostman.com/view/11693148/UVC5Fo9R)

- **Folder structure**

- **config**

The `packages/marvel/config` folder contains all the `config` for our app.

- **database**

The `packages/marvel/database` folder contains all

the factories and migrations.

- - **Http:**

Contains two folders. Controllers and Requests. All the necessary controllers and requests are in this two folder.

- - **Database:**

Contains Models and Repositories. For repositories we have used l5-repository (<https://github.com/andersao/l5-repository>).

- **Enums**

All the enums that are used throughout the app is in packages/marvel/src/Enums folder.

- **Events**

All the events are in packages/marvel/src/Events

to later.

- **Listeners**

All the listeners corresponding to the above events are in `packages/marvel/src/Listeners` folder

- **Mail**

All the mailables are in `packages/marvel/src/Mails` folder.

- **Notifications**

Notifications related to order placed is reside `packages/marvel/src/Notifications`. Currently we have provided mail notification but you can

easily add others notification system using laravel conventions.

## • Providers

All the secondary service providers that we have used in our app resides in `packages/marvel/src/Providers` folder. The main `ShopServiceProviders` reside in `packages/marvel/src/` folder.

## • stubs

The `packages/marvel/stubs` folder contains all the necessary email templates and demo data related resources for the app.

“

## • Before Finishing up

Before you finishes the installation process

make sure you have completed the below steps.

- Copied necessary files and content to your existing laravel projects(if using existing projects)
- Installed all the necessary dependencies.
- Ran `marvel:install` commands and followed the necessary steps.
- Created a .env file with all the necessary env variables in the provided projects.
- Put `DISK_NAME` configuration for `public` or '`s3`'
- Set Payment related configuration to `STRIPE_API_KEY`

## • Payment Gateway

We have used `omnipay` for payment and given `stripe` and `cash on delivery` default. We have

stripe and cash\_on\_delivery default. we have used [ignited/laravel-omnipay](#) by forking it in our packages due to some compatibility issue with Laravel 8.

## Extending The Functionality

If you want to extend the functionality of the app you can easily do it on your app. You would not need to modify code from our packages folder. Like you can add any [routes](#) and corresponding [controller](#) in your laravel app as your need. We highly suggest you to do all the modification in your app so you can update the package easily.

## Deployment

Its a basic laravel application so you can

deploy it as any other laravel application. Make sure you have installed all the php required plugins we mentioned above.

## New Static Page

---

Both shop and admin are built using React NextJS framework. So all the existing pages are available to this location. You can create new pages from,

- **Shop ,**

`pixer-laravel/shop/src/pages`

• Admin,

pixer-laravel/admin/src/pages

You can use the NextJS routing feature for the new pages. Check these official NextJS docs for pages and routing,

<https://nextjs.org/docs/basic-features/pages>

<https://nextjs.org/docs/routing/introduction>

**Thank You!**

-----

