

# Welcome

---

## Introduction to Pixier

---

Welcome to the Pixier documentation!

We have presented one of the fastest and digital assets selling E-commerce app built with [Laravel](#), [React](#), [NextJS](#), [TypeScript](#) and [Tailwind CSS](#). Here, we have tried to cover all the available features & topics presented in this application. We are inviting you to go through the whole documentation, so that it may helps you to understand better about what we are offering.

### 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

In your production environment, maintaining similar configuration is recommended.

- MacOS, Windows, and Linux are supported
- Compatible Browsers (Firefox, Safari, Chrome, Edge)
- JavaScript (Node.js 16.15.1 or later)
- Laravel 10 environments with PHP 8.1 (at least)

### Requirements

- node (v16.15.0 or later, [v20.9.0](#) much more appreciable)
- yarn/npm/pnpm
- PHP 8.1
- MySQL 8
- ext\_curl
- ext\_gmp
- 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](#)

## Getting Started

---

For getting started with the application you have to follow the below procedure. For more help we have divided the installation portion for each OS.

- Windows
- Linux or MacOS

## Installation Windows

<https://www.youtube.com/embed/xoQn4AoemxU>

### Prerequisites

- PHP 8.1
- 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(16.15.0 or later)
- yarn
- editor: [Visual Studio Code](#)(recommended)

### Getting Started

1. First download the file from codecanyon.
2. Unzip the downloaded file and folder structure you get

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

3. 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=pixer_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: 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

NB: Move pixier-laravel folder from Pixier - React Laravel Multivendor Digital Marketplace folder

## Installation Mac

### Video

<https://www.youtube.com/embed/RpnWz3wr9VQ>

- Run Docker application first
- Now go to your pixier-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 `pixier-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

4. 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`**

5. Run yarn at the root directory.

```

# on pixier/root directory
yarn

```

6. 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 pixier 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: \*\***

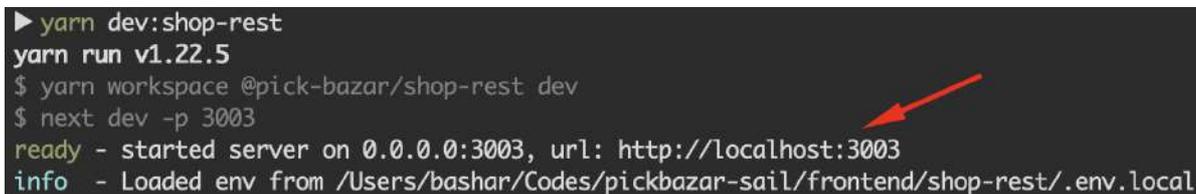
- The page will automatically reload if you make changes to the code. You will see the build errors and lint warnings in the console.
- If you saw any error while running make Sure you setup your API endpoint properly at `.env` file.

#### **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
```

```
▶ 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 build:shop-rest
yarn run v1.22.5
$ 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]                             27.2 kB    268 kB
├   └ css/7936e7aa084bd830e87c.css    4.01 kB
├     └ /grocery
├     └ /makeup
├     └ /bags
├     └ [+2 more paths]
├ ○ /404                             2.74 kB    179 kB
├ • /bakery (ISR: 60 Seconds)         2.66 kB    243 kB
├   └ css/0dcfbd7860d56385c547.css    4.01 kB
├ λ /change-password                 3.25 kB    191 kB
├ ○ /checkout                       4.85 kB    189 kB
├ ○ /contact                         2.83 kB    191 kB
├ ○ /example                         6.71 kB    183 kB
├ ○ /help                            2.75 kB    182 kB
├ ○ /logout                          4.99 kB    181 kB
├ ○ /offers                          9.81 kB    189 kB
├ λ /order                          15.4 kB    209 kB
├   └ css/db04274d0e1caa11303a.css    351 B
├ λ /order-received/[tracking_number] 6.85 kB    186 kB
├ λ /orders                         28.5 kB    208 kB
├   └ css/da2daebf602ff72a3fc0.css    2.1 kB
├ ○ /privacy                        1.99 kB    188 kB
├   └ products/[slug]
├     └ css/efb94a24c22d59f4d562.css    4.17 kB
├     └ /products/apples
├     └ /products/baby-spinach
├     └ /products/blueberries
├     └ [+27 more paths]
├ λ /profile                        11.8 kB    200 kB
├ ○ /terms                          1.98 kB    188 kB
├ ○ /ui                             8.82 kB    185 kB
├ + First Load JS shared by all
├   176 kB
├   chunks/00682edc09922f9a0564b6d51ebfa3fe9fa3f6bc.837c2b.js 3.14 kB
├   chunks/03597ab9dee95bb8e6991ab21a5b170a1bc330ab.CSS.3bbe0c.js 68 B
├   chunks/1c2031c8.c533da.js 23.5 kB
├   chunks/2c96be8e9bf36e109a607a23c3ac04c26631eb28.f78ccb.js 4.76 kB
├   chunks/57ad4ee9d685f4a85f091607d3121ddc66f6f331.d3ef7d.js 35.8 kB
├   chunks/6bc2a55fc5fb5a7f77a0d800e4657cca465fac5a.173fc5.js 7.14 kB
├   chunks/78478e0567901f0a00137009c7f6751fab214ebc.855475.js 15.3 kB
├   chunks/aad9552d4f74729e02303abd0a7d42186859f2ec0.37727a.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.

7. 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.

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

## PHP Artisan Oriented Scripts

Available Scripts:

API Command

```

"php artisan marvel:install": "Installing Marvel Dependencies",
"php artisan marvel:create-admin": "Create an Admin",
"php artisan marvel:seed": "Import Demo Data",
"php artisan marvel:settings_seed": "Import Settings Demo Data",
"php artisan marvel:copy-files": "Copy necessary files",

```

## Sail Oriented Scripts

### Available Scripts:

```
"sail artisan marvel:install": "Installing Marvel Dependencies",
"sail artisan marvel:create-admin": "Create an Admin",
"sail artisan marvel:seed": "Import Demo Data",
"sail artisan marvel:settings_seed": "Import Settings Demo Data",
"sail artisan marvel:copy-files": "Copy necessary files",
```

### Most uses Laravel commands.

```
"php artisan serve": "To start project",
"php artisan config:cache": "change environmental variables",
"php artisan key:generate": "Generate new application key",
"php artisan migrate": "If you want to migrate the database tables",
"php artisan storage:link": "Create a symbolic link in Laravel application.",
"composer require Vendor_name/Package_name": "Install New packages in Laravel application",
"composer update": "Update all your packages or specific package",
```

### Available Scripts:

#### Admin Command

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

```
"clean": "yarn workspaces run rimraf \"
{.next,node_modules,__generated__,.cache,src/graphql/*.d.ts,src/framework/graphql/**/*.*.d.ts}\" && rimraf node_modules",
"dev:admin-rest": "yarn workspace @marvel/admin-rest dev",
"build:admin-rest": "yarn workspace @marvel/admin-rest build",
"start:admin-rest": "yarn workspace @marvel/admin-rest start",
"prepare": "husky install"
```

\*\* Note: \*\* Also, individual Scripts are available under every individual package. You can check out them from there individual package.json file.

### Available Scripts:

#### Shop Command

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

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

\*\* Note: \*\* Also, individual Scripts are available under every individual package. You can check out them from there individual package.json file.

## How can I use this app

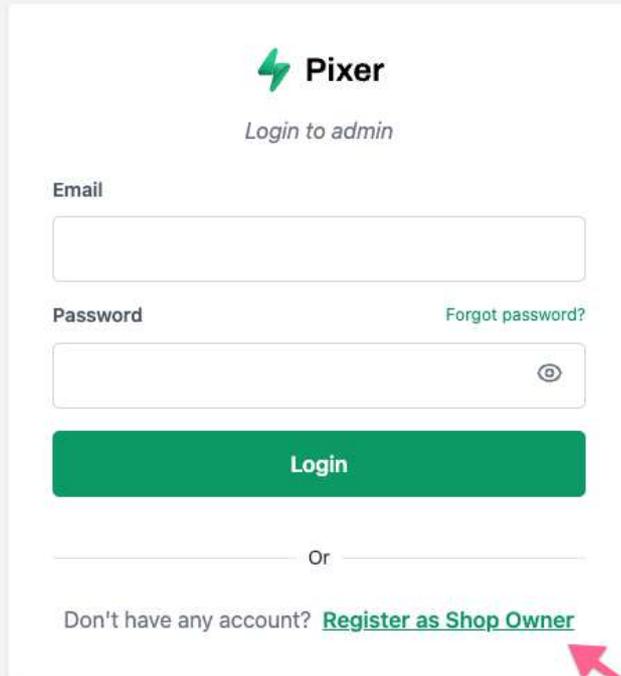
---

### MultiVendor

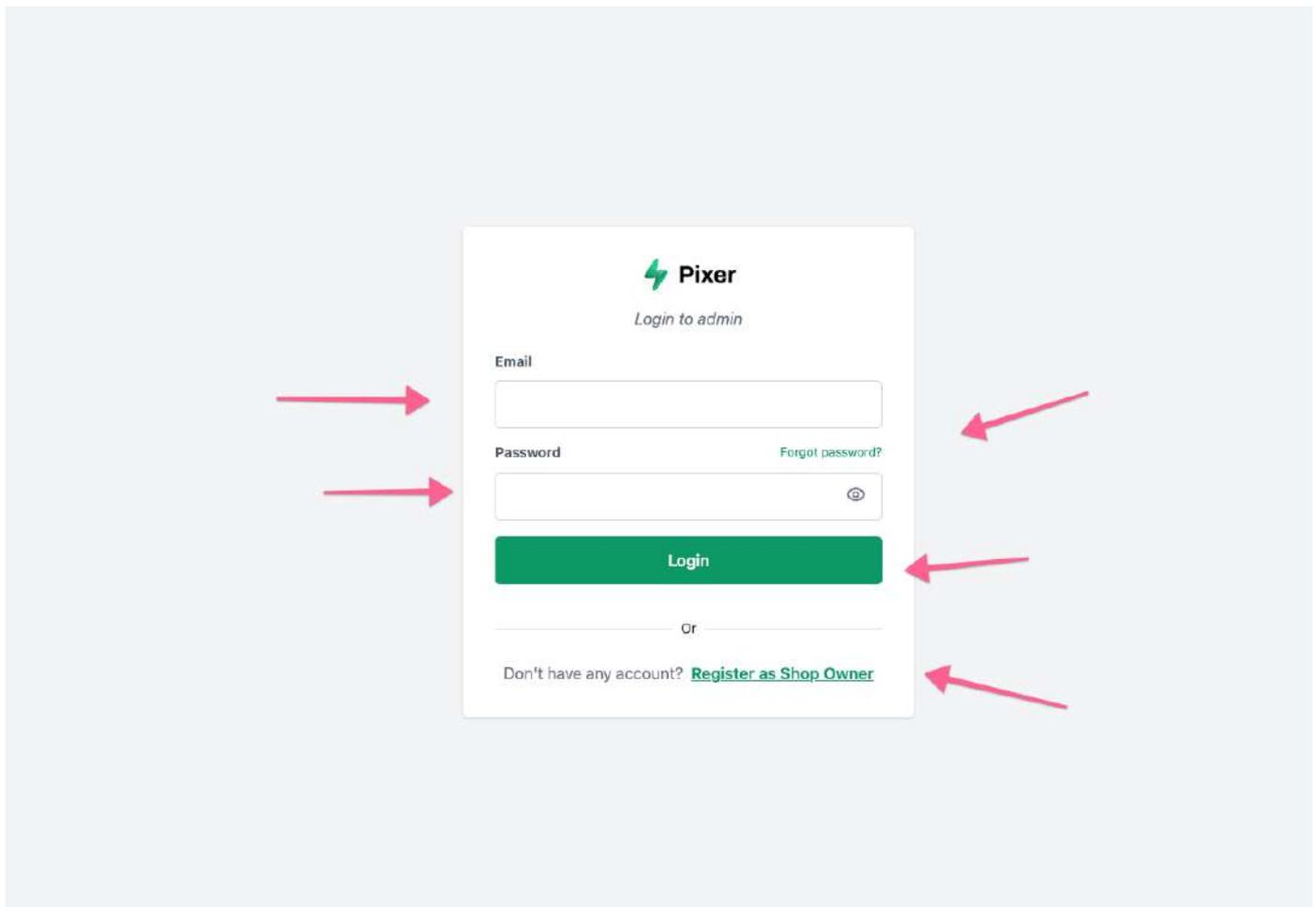
In this chapter we are going to point out with screenshots about the multivendor flow of Pixier application. So that you can understand the view of the application.

#### Create New Shop

To create new shop login as an [administrator](#) or create a new vendor account for creating [shop](#)



The image shows a login form for Pixier. At the top, there is a green lightning bolt icon followed by the text "Pixier". Below this is the text "Login to admin". The form contains two input fields: "Email" and "Password". The "Password" field has a "Forgot password?" link to its right. Below the input fields is a green "Login" button. Underneath the button is a horizontal line with the word "Or" in the center. At the bottom of the form, there is a text prompt "Don't have any account?" followed by a green link "Register as Shop Owner". A red arrow points to the "Register as Shop Owner" link.



After creating the account you'll be redirected to this page,

**Pixer**

Search your route...

Create Shop Visit Site

Language English

Store Owner

**Store Owner**  
vinton@demo.com

This is the store owner and we have 8 shops under our banner. We are running all the shops to give our customers hassle-free service and quality produ...

Read more

Contact: +1 2365141641631

MENU

Dashboard

My Shops

**Summary**

Total Revenue **\$0.00**

Total refunds **\$0.00**

Total Shops **10**

Todays Revenue **\$0.00**

**Order Status**

Today Weekly Monthly Yearly

Pending Order **0**

Processing Order **0**

Completed Order **0**

Cancelled Order **0**

**Sale History**

January February March April May June July August September October November December

**Top 10 Most Rated Products**

No data found  
Sorry we couldn't found any data

**Top 10 Category with most products**

Category ID	Category Name	Shop	Product Count
#ID: 9	Wireframe Kits	BentaSoft	5
#ID: 9	Wireframe Kits	Imagineco	5
#ID: 9	Wireframe Kits	BentaSoft	5
#ID: 9	Wireframe Kits	Imagineco	5
#ID: 7	Angular	Qubitron Solutions	4
#ID: 6	WordPress Theme	BentaSoft	5
#ID: 9	Wireframe Kits	BentaSoft	5
#ID: 5	WordPress Plugin	BentaSoft	4
#ID: 17	Shopify	BentaSoft	4
#ID: 7	Angular	BentaSoft	4
#ID: 11	Illustrations	BentaSoft	4
#ID: 8	CMS	Omatron	4
#ID: 16	Joomla	Maxicon Soft Tech	4
#ID: 15	Bootstrap	BentaSoft	4

After that, click **Create Shop**, and provide all the information for the store.

**Pixer**

Search your route...

Create Shop Visit Site

Language English

Store Owner

**Store Owner**  
vinton@demo.com

This is the store owner and we have 8 shops under our banner. We are running all the shops to give our customers hassle-free service and quality produ...

Read more

Contact: +1 2365141641631

MENU

Dashboard

My Shops

**Create Shop**

**Logo**  
Upload your shop logo from here

Upload an image or drag and drop  
PNG, JPG

**Cover Image**  
Upload your shop cover image from here  
Dimension of the cover image should be **1170 x 435px**

Upload an image or drag and drop  
PNG, JPG

**Basic Info**  
Add some basic info about your shop from here

Name \*

Slug

Description

**Payment Info**  
Add your payment information from here

Account Holder Name \*

Account Holder Email \*

Bank Name \*

Account Number \*

**Shop Address**  
Add your physical shop address from here

Country

City

State

ZIP

Street Address

**Email Notification**  
Set your email notification for messaging feature

Notification email

Enable Notification

**Shop Settings**  
Add your shop settings information from here

Contact Number \*

Website \*

**Social Profile Settings**  
Add your social profile information from here

[Add New Social Profile](#)

[Save](#)

After creating the shop you'll redirect to this page,

©2023 Pixier. Copyright © REDQ. All rights reserved worldwide. REDQ

By default, the shop will be inactive. Only an **administrator** can active a shop from Super Admin dashboard.

ID	Shop	Products	Orders	Owner Name	Status	Actions
#ID: 11	Demo shop	0	0	Store Owner	Inactive	
#ID: 9	Maxicon Soft Tech	6	0	Store Owner	Active	
#ID: 8	BentaSoft	6	0	Store Owner	Active	
#ID: 7	Omnico Team	5	3	Store Owner	Active	
#ID: 6	Imagineco	6	0	Store Owner	Active	
#ID: 5	Omatron	5	0	Store Owner	Active	
#ID: 4	Bitronic	5	0	Store Owner	Active	
#ID: 3	Qubitron Solutions	6	0	Store Owner	Active	
#ID: 2	BentaSoft	6	0	Store Owner	Active	
#ID: 1	Temper studios	6	0	Store Owner	Active	

©2023 Pixier. Copyright © REDQ. All rights reserved worldwide. REDQ

After click on **shop**, you'll be redirected to dashboard page,

The screenshot displays the Pixier dashboard for a vendor named 'Temper studios'. The interface includes a sidebar with navigation options such as Dashboard, Products, Inventory, Withdrawals, Orders, Transactions, Reviews, Questions, Staff, FAQs, and Terms And Conditions. The main content area shows a vendor profile with contact information (vendor@demo.com, 3399 Buffalo Creek Road, Goodlettsville, Tennessee, 37072, USA) and a grid of statistics: 6 Total Products, 0 Total Orders, 10% Admin Commission Rate, \$0.00 Gross Sales, and \$0.00 Current Balance. The background features a mobile app interface showing a list of products and user reviews.

From this dashboard, you can maintain the shop,

## User Role : Super Admin

In a multi-vendor e-commerce platform, the super admin embodies supreme control, overseeing the entire ecosystem. Armed with unparalleled authority, they manage vendors, uphold platform integrity, regulate operations, and ensure a seamless experience for buyers and sellers, pivotal in steering the platform's success and sustained growth.

## User Role : Store Owner or Vendor

In a multi-vendor e-commerce platform, a vendor represents an independent seller or entity offering diverse products or services. Vendors operate individual storefronts within the marketplace, showcasing their unique offerings. They manage inventory, sales, and fulfillment, contributing to the platform's varied and extensive product catalog, fostering competition and consumer choice.

## User Role : Staff

In a multi-vendor e-commerce platform, shop staff are pivotal intermediaries, managing vendor interactions, ensuring seamless operations, and fostering vendor relations. They oversee onboarding, resolve orders & disputes, optimize product listings, and provide ongoing support, crucially upholding quality standards while nurturing a collaborative environment to enhance the platform's marketplace experience.

## super admin

The super admin role in an e-commerce site holds ultimate control. They manage users, permissions, and critical settings, ensuring seamless operations and security at the highest level.

As, super admin has the supreme authority over any vendor, so he/she can control any listed shops in the application. Here are the key notes for super admin,

- Super admin can see access all settings & menu.
- Super admin is also a vendor, staff, customer himself/herself. So, he/she can listed his own shop & control them as a vendor/staff/customer perspective.

## Vendor or Store Owner

Vendors are generally the shop owner. A vendor can register in the application as a seller. And he/she can open a shop based on agreement with super admin. Some key notes for Vendors,

- Vendor can listed his/her own products.
- Vendor can assign staffs for his/her shop.

- Vendor can monitor his orders, based on his provided order status, super admin will take rest actions.
- Vendor can set his/her own FAQs based on his shop individually. For example, a Grocery shop's FAQ may not be similar with a Gadget shop's FAQ.
- Vendor can set shop specific Terms & Conditions, if and only if Super admin gave permission.

## Staff

The staff role in an e-commerce website typically involves managing day-to-day operations of a vendor's shop. Staff members assist customers, process orders, update product listings, and ensure smooth website functionality. They may also handle customer inquiries, resolve issues, and collaborate with vendors and other team members to maintain a seamless shopping experience for customers.

## Customer

### Login Customer

The customer role in Pixier Laravel represents the end-users who visit the site to browse, select, and purchase products or services. Customers create accounts, place orders, make payments, and provide feedback.

## Demo Deployment

---

For Deployment we have provided two different approaches.

- [Manual Installation](#) guide.
- [Automated Script Installation](#) guide.

Please follow those link above based on your choice to know more about how to deploy the apps in virtual private server for inspecting the demo and features.

## How to upgrade the existing deployed laravel 9 server to laravel 10?

You can check older laravel version upgrade guides [here](#)

At first, remove all the existing `php 8.0` and its extensions by using this command,

```
sudo apt purge php8.0-fpm php8.0-mysql
sudo apt purge php8.0-mbstring php8.0-xml php8.0-bcmath php8.0-simplexml php8.0-intl php8.0-mbstring php8.0-gd php8.0-curl
php8.0-zip composer
```

Then remove the composer,

```
sudo rm /usr/bin/composer
```

Then delete the `vendor` folder from the `pixier-laravel -> api` folder.

```
cd /var/www/pixier-laravel/api

sudo chown -R $USER:www-data storage
sudo chown -R $USER:www-data bootstrap/cache

rm vendor -rf
```

Then install PHP 8.1, and its extensions,

```
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

```
sudo apt install php8.1-fpm php8.1-mysql
```

```
sudo apt install php8.1-mbstring php8.1-xml php8.1-bcmath php8.1-simplexml php8.1-intl php8.1-gd php8.1-curl php8.1-zip
php8.1-gmp
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
```

```
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d64d92e6563bbde02') { echo 'Installer
verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

```
sudo mv composer.phar /usr/bin/composer
```

Then update 8.0 fpm to 8.1 fpm from,

```
/etc/nginx/sites-enabled/pixer
```

```
# For API
location /backend {
    alias /var/www/pixer-laravel/pixer-api/public;
    try_files $uri $uri/ @backend;
    location ~ /\.php$ {
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $request_filename;
        fastcgi_pass unix:/run/php/php8.1-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;
}

error_page 404 /index.php;

location ~ /\.php$ {
    fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
```

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

```
sudo nginx -t
```

Next, restart Nginx:

```
sudo systemctl restart nginx
```

Then install composer packages,

```
cd /var/www/pixer-laravel/api
```

```
composer install
```

```
php artisan optimize:clear
```

```
php artisan marvel:install
```

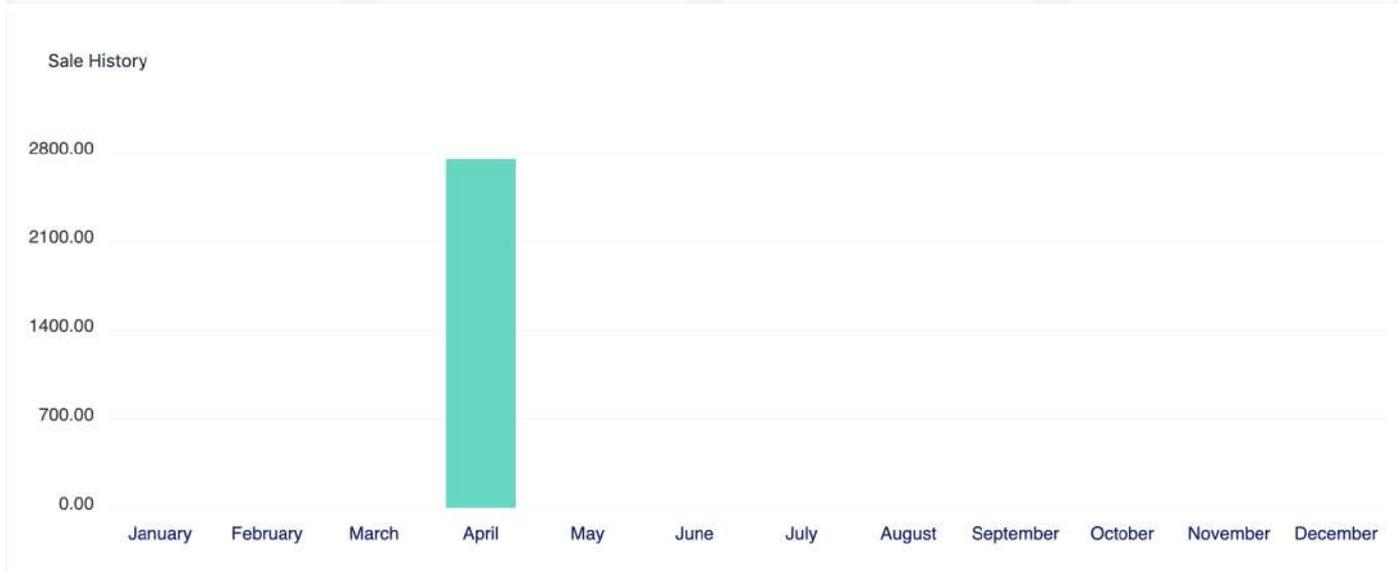
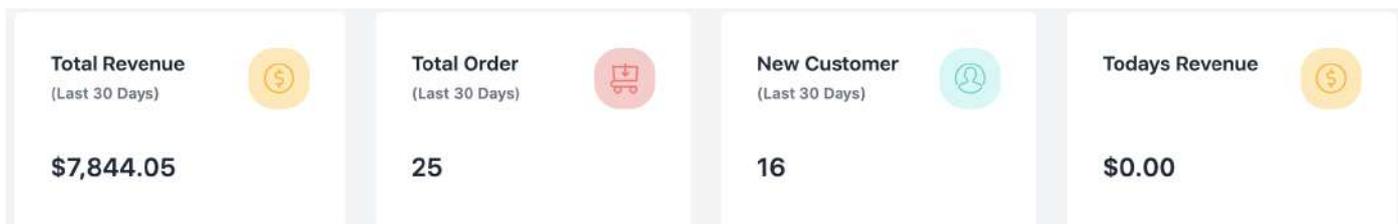
php artisan marvel:install will remove all of your existing data. Ensure you export or backup your data before using that command.

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

## Admin dashboard

- Analytics Dashboard

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



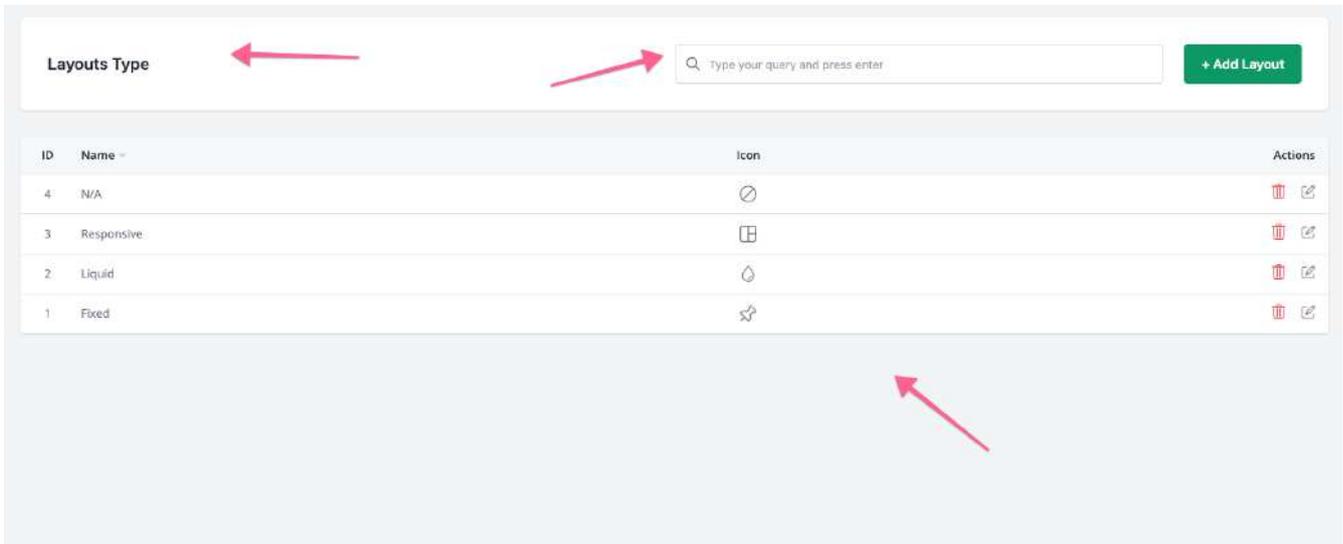
## Recent Orders

Tracking Number	Total	Order Date	Status
GkszijsVAcae	\$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
BlOnTtmsHHHE	\$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 Thi	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

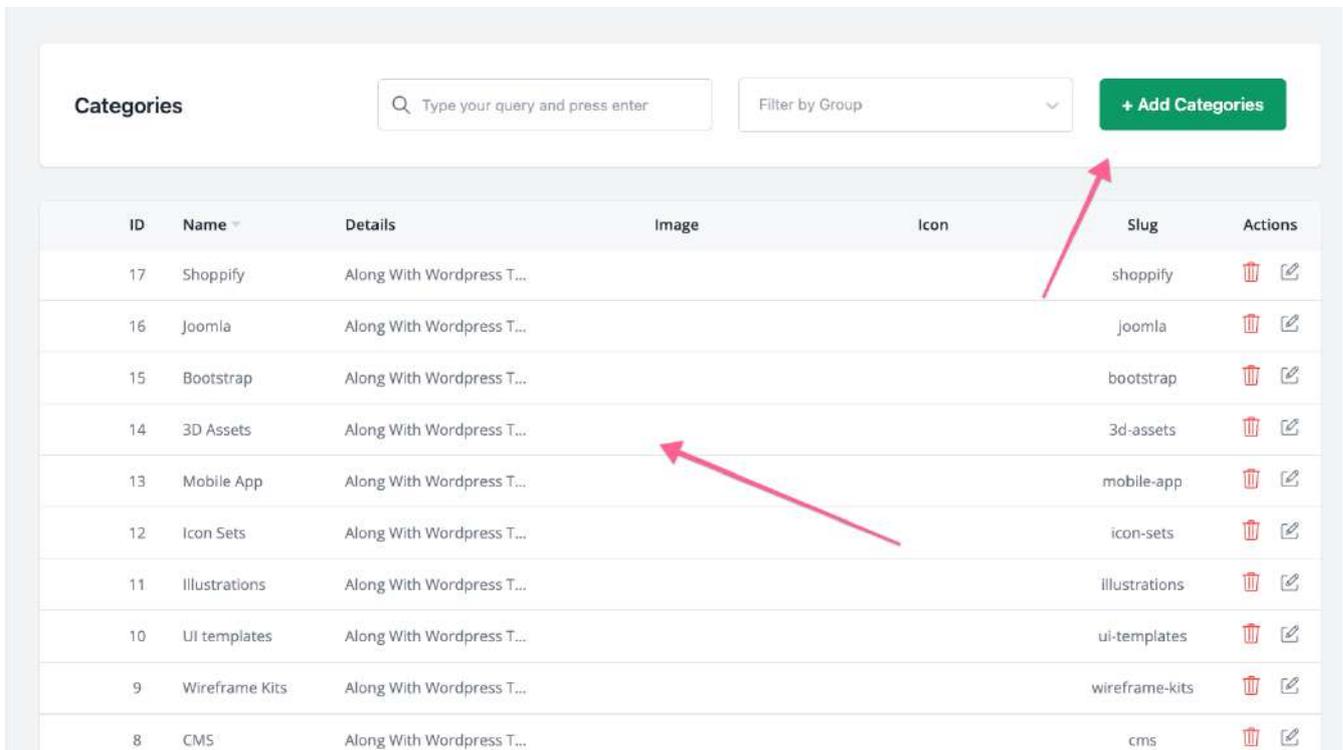
- Manage Layout Type

In [Layout Types](#) menu you will get the product types and you can add, remove or modify product type from there.



• Manage Product Category

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



• Product Management

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

Products  Filter

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

A portion of product form

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

**Gallery**  
Upload your product image gallery here

**Group & Categories**  
Select product group and categories from here

Group\*  
Responsive

Categories  
Angular x Shipping 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.

**Order Status**

Search: Type your query and press enter

+ Add Order Status

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	

Page 1 of 1

## Order Management

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

**Orders**

Search: Type your query and press enter

	Tracking Number	Total	Order Date	Status	Actions
	magjvy6B8c4x	\$39.90	2 days ago	Confirmed	
	qZM6xOyvFz0V	\$84.80	2 days ago	Confirmed	
	oudg2EbF9Gww	\$105.00	2 days ago	Confirmed	
	scU4nn9EOABM	\$162.75	2 days ago	Confirmed	
	SA6cCIT3Lod5	\$150.00	2 days ago	Confirmed	
	xATLDw80ipU0	\$195.00	2 days ago	Confirmed	
	RkekswCuelP6	\$158.00	2 days ago	Confirmed	
	QM0fKBzywmLF	\$49.00	2 days ago	Confirmed	
	BtrYveTqtFmc	\$110.00	2 days ago	Confirmed	

A portion of order management. Order status change.

Order ID - magjvy6B8c4x

Confirmed — On Hold (2)

Select...  
Confirmed  
On hold

Download Invoice

Change Status

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

### Customer Management

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

Customers

Type your query and press enter

+ Add Customer

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

**Block Customer**

Are you sure you want to block this customer?

Cancel Block

### Tax Management

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

**Taxes**

Search by Name

ID	Name	Rate (%)	Country	City	State	ZIP	Actions
1	Quibitron Solutions	5	United States	Los Angeles	California	90017	

## Settings Management

In [settings](#) menu you will get the settings management form there.

**Settings**

**Information**  
Change your site information from here

**Site Title**  
Pixier

**Site Subtitle**

**Currency**  
Euro

**Minimum Order Amount**

**Wallet Currency Ratio**

## Available Scripts:

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

### Shop

```
"clean": "rimraf \"{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 you 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.

1. Open your custom SVG icon file in the code editor and copy all the code.
2. Then Go to `src -> components -> icons` folder and create a new `.tsx` file.
3. Then here create a function component and paste your copied SVG code inside the return statement.
4. Then covert 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. )
5. If your custom SVG code has any single custom color then change them into `fillColor`.

## Utilities

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/utlis/
```

You can use or customize these helper functions according to your needs.

## Backend integration

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 our 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 { 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());
```

## Deployment

### 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
```

The screenshot displays the AWS Management Console interface. The main section is titled 'AWS services' and contains a list of services categorized under 'All services'. A red arrow points to the 'EC2' service under the 'Compute' category. Other categories include 'Containers', 'Storage', 'Quantum Technologies', 'Management & Governance', and 'Security, Identity, & Compliance'. On the right side, there are promotional banners for mobile apps, digital training, and SageMaker resources. The footer shows the language set to English (US) and the copyright notice for 2008-2021.

The screenshot shows the AWS Management Console interface for EC2 instances. The left-hand navigation pane has 'Instances' selected. The main content area shows a table with one instance: i-00fb9dda581f15817, which is in a 'Terminated' state. The 'Launch Instances' button is highlighted in orange in the top right corner.

Then select a ubuntu 20.04 server

After that, click **Next** -> **Next** -> **Next** -> **Next**

And on security pages, add a rule for **HTTP**, **HTTPS** and **SSH**,

Our automation scripts setup **HTTPS** in your domain so you should open **HTTPS**

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom ::/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom ::/0	e.g. SSH for Admin Desktop

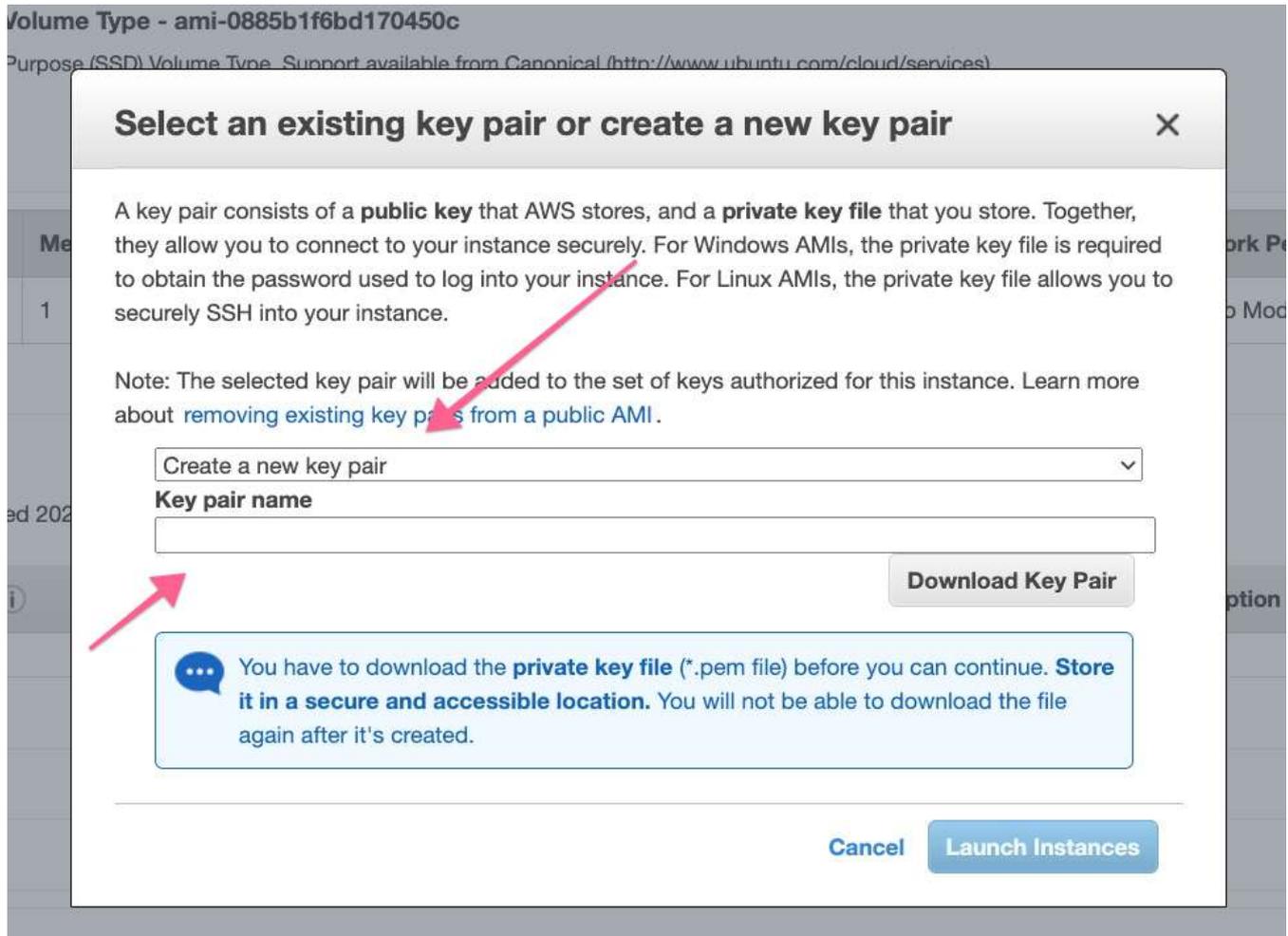
Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

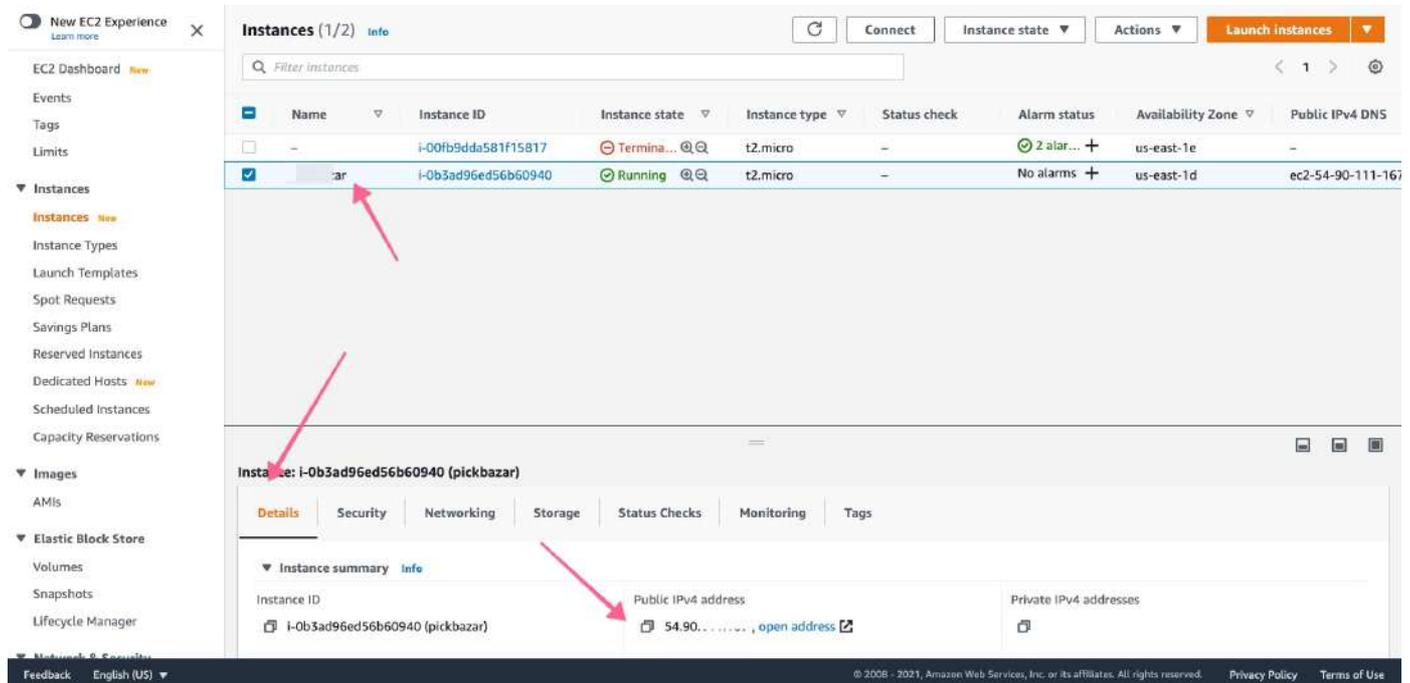
Cancel Save

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.



### Domain Setup

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

The screenshot shows the AWS Management Console interface for EC2 instances. On the left is a navigation menu with categories like EC2 Dashboard, Events, Tags, Limits, Instances, Images, and Elastic Block Store. The main area displays a table of instances. One instance, 'pickbazar', is selected. Below the table, the details for this instance are shown, including its ID, state (Running), type (t2.micro), and public IPv4 address (54.90...). Red arrows highlight the instance name in the table, the instance ID in the details view, and the public IPv4 address.

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**

This screenshot shows the top right corner of the AWS console instance details page. A red arrow points to the 'Connect' button, which is used to access the connection options for the selected instance.

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

The screenshot shows the 'Connect to instance' dashboard. The 'SSH client' tab is selected. It provides instructions on how to connect to the instance using an SSH client. A red arrow points to the 'SSH client' tab, and another red arrow points to the example command: `ssh -i "tareqmahmud.pem" ubuntu@ec2-54-90-111-167.us-east-1.compute-1.amazonaws.com`. A 'copy' label with an arrow points to the command. A note at the bottom states: "Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name."

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 [Pixier Laravel](#)

## VPS Server

### Virtual Private Server (Automated Script)

With this tutorial, you can install Pixier 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 ([v20.0.4 lts](#)) server with at least 2+ CPU cores and 2GB+ memory.

Please connect your [domain](#) with [server](#). We don't recommend/support deployment the project via [IP](#).

Please follow this video with the documentation, and it'll make the installation process relatively easy.



#### Prerequisite

This automated script is for the \*nix system. So if you are using mac or Linux, then you're good to go. But if you are using windows, then install WSL on your computer and use this script using [WSL](#) or follow this [manual installation](#)

Before starting, the procedure ensures that NodeJS 16 (the latest) is installed on your computer.

```
npm i -g yarn zx
```

Now you can follow the script installation procedure,

#### At first login your server from terminal

```
ssh SERVER_USERNAME@SERVERIP
```

Make sure that you are logged in your server then follow the next step and run suggested command.

#### Upload api and deployment project to Virtual Server form your PC - RUN on Local PC

To upload the zipped [pixier-api](#) and [deployment](#) files to server you need to run the below command form your pixier project root

while running below command you will asked for enter your server [username](#) and [ip address](#) by entering and a successful connection you will also asked for enter your [pixier-api.zip](#) and [deployment.zip](#) files path and the path will be look like `/home/your_project_folder_path/pixier-laravel/pixier-api.zip` for [pixier-api.zip](#) file so forth for [deployment.zip](#)

```
bash deployment/deployment.sh
```

Then login your server using `ssh` and,

#### Server Environment setup script - RUN on Virtual Server

```
bash /var/www/pixer-laravel/deployment/nodesetup.sh
```

#### Nginx Setup And Settings - RUN on Virtual Server

```
zx /var/www/pixer-laravel/deployment/setenv.mjs
```

#### Backend build - RUN on Virtual Server

```
sudo zx /var/www/pixer-laravel/deployment/backendbuildscript.mjs
```

#### Frontend build script - RUN on Local PC

Run the below command from your pixer-laravel project root

```
zx deployment/frontendbuildscript.mjs
```

#### Frontend run script - RUN on Virtual Server

```
zx /var/www/pixer-laravel/deployment/frontendrunscript.mjs
```

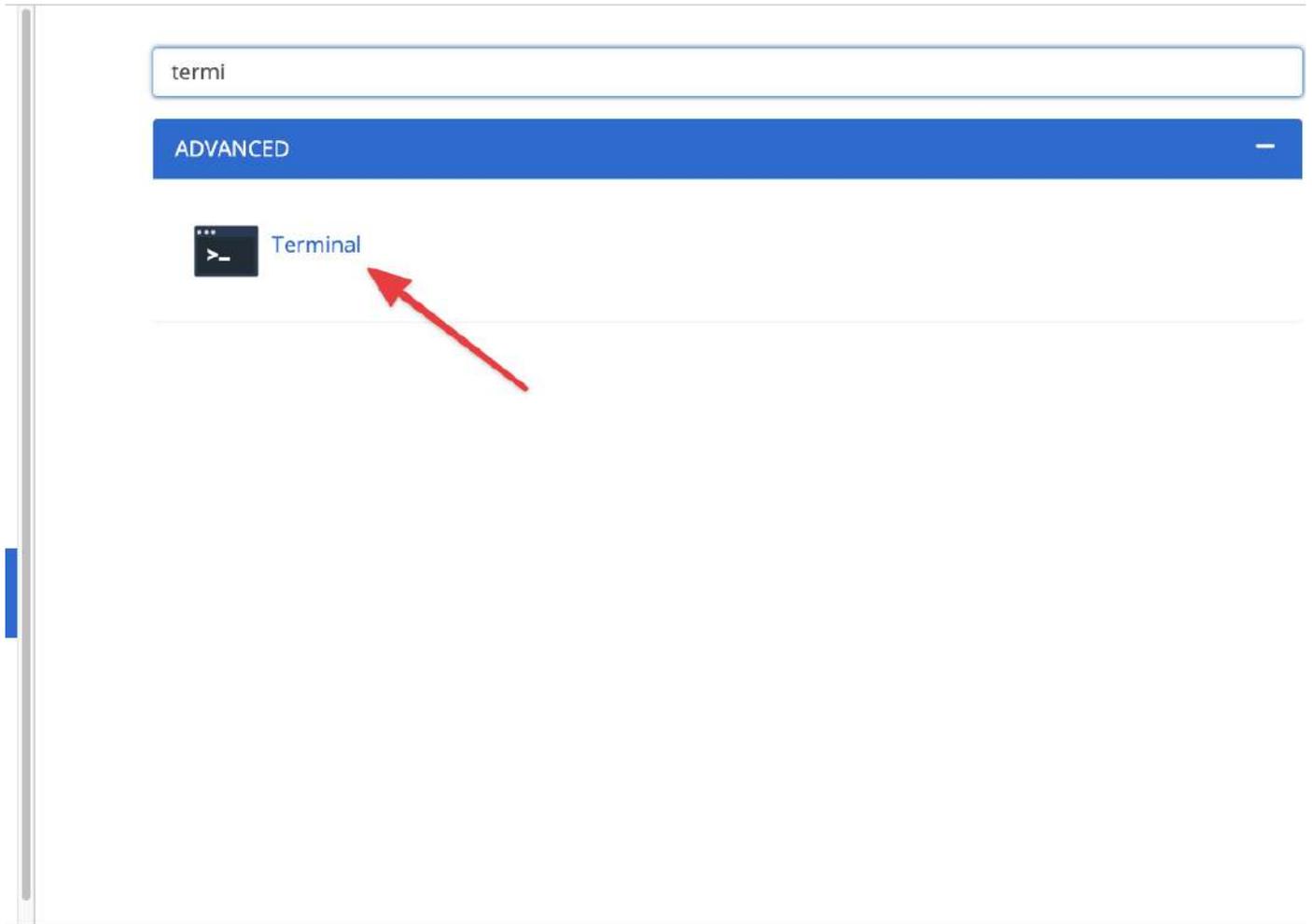
## cPanel

It's quite hard to debug any deployment issue on the cPanel or any managed 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 or any managed server 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

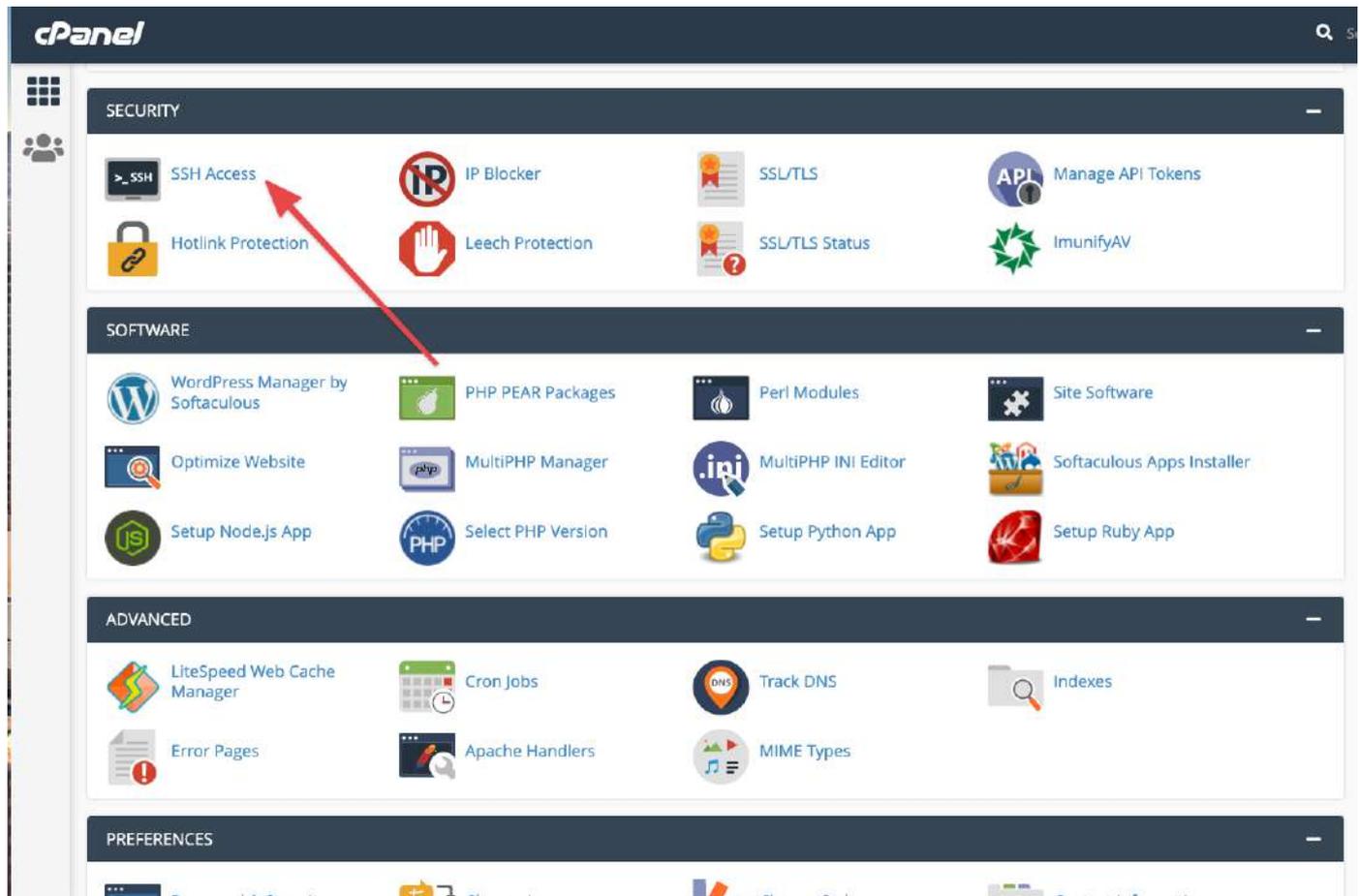
If you still decide to proceed with cpanel, our support team won't be able to help you. We have put some resources for Cpanel in this documentation section to help our users to get started but other than that, we don't have much to offer with Cpanel.

#### 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
```

```
yarn (node)  %1  ..pickbazar-doc (-zsh)  %2  "mahmud" (ssh)  %3
[~]# composer --version
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 `8.1`

### 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,

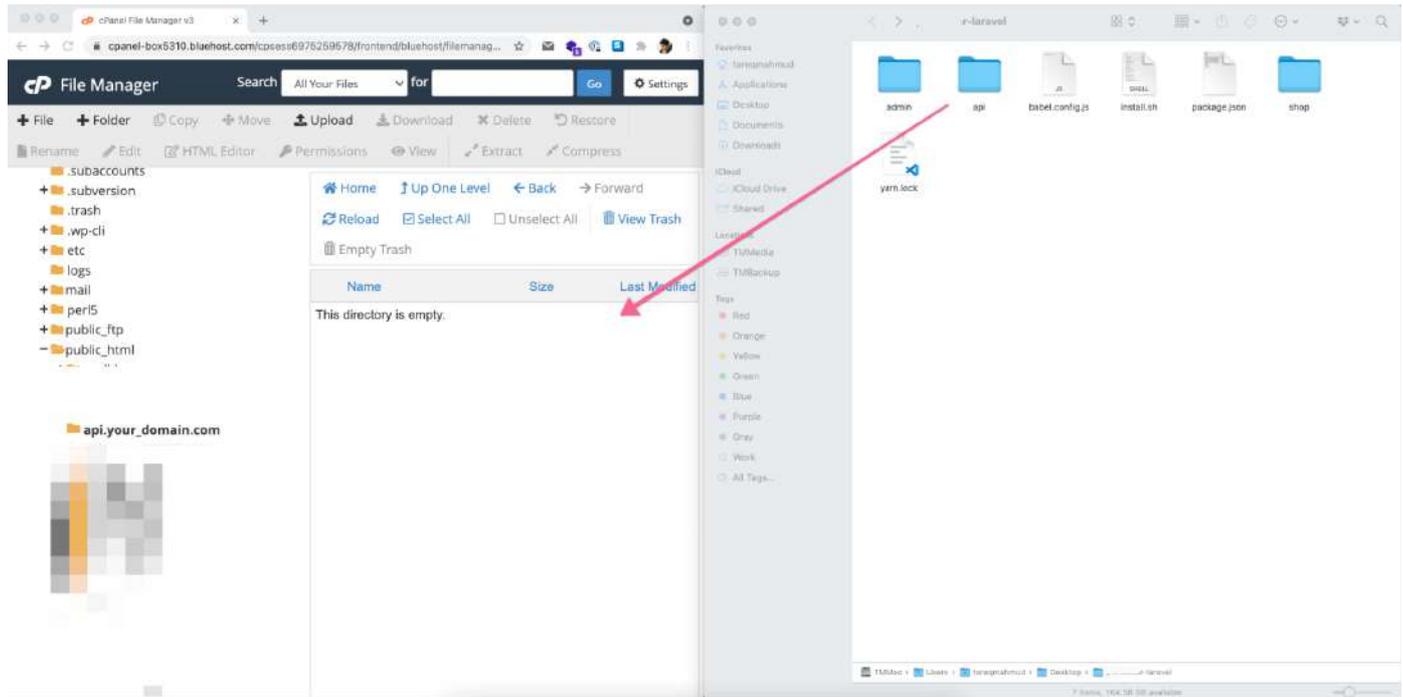
```
-> store.your_domain.com -> host frontend store.
-> api.your_domain.com -> host laravel API.
-> admin.your_domain.com -> host admin dashboard.
```

After creating domain/subdomains, make sure all the domain/subdomains are HTTPS enabled. Please contact your hosting provider to enable this, as most hosting providers provide some sort of free SSL.

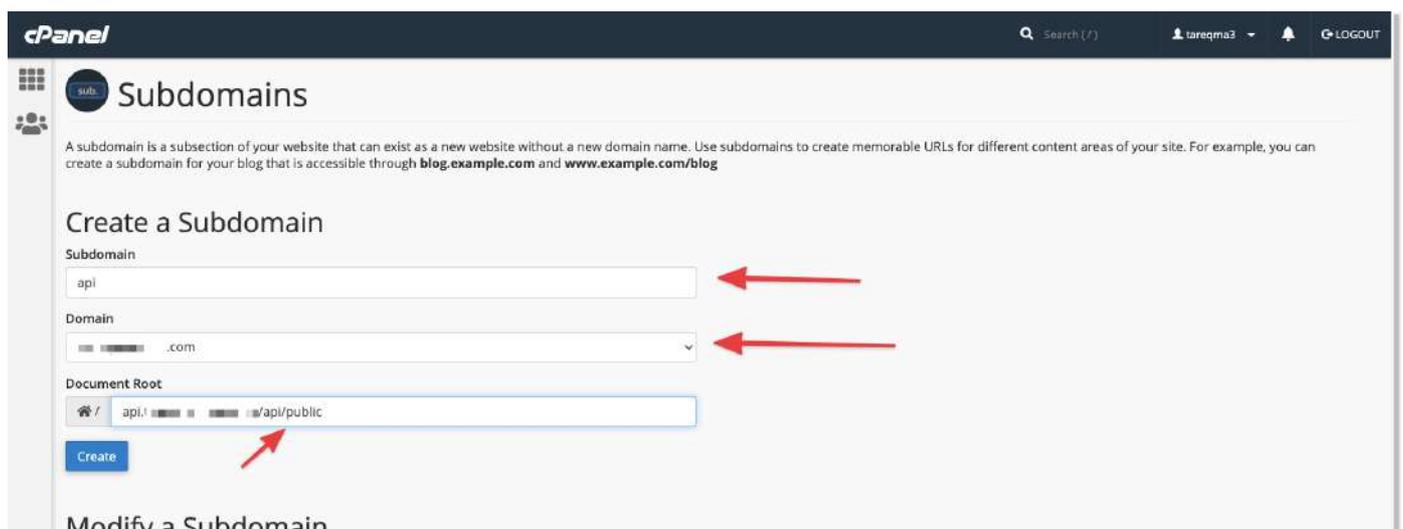
## Install API

1. Extract the `pixier-laravel` package that you download from [CodeCanyon](#).
2. On that folder, you'll get another zip called `pixier-laravel.zip`.
3. Now extract this `pixier-laravel.zip` file.
4. On that file, you'll get a folder called `pixier-api`

Now upload this `pixier-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.



Now create a MySQL database and user from MySQL wizard

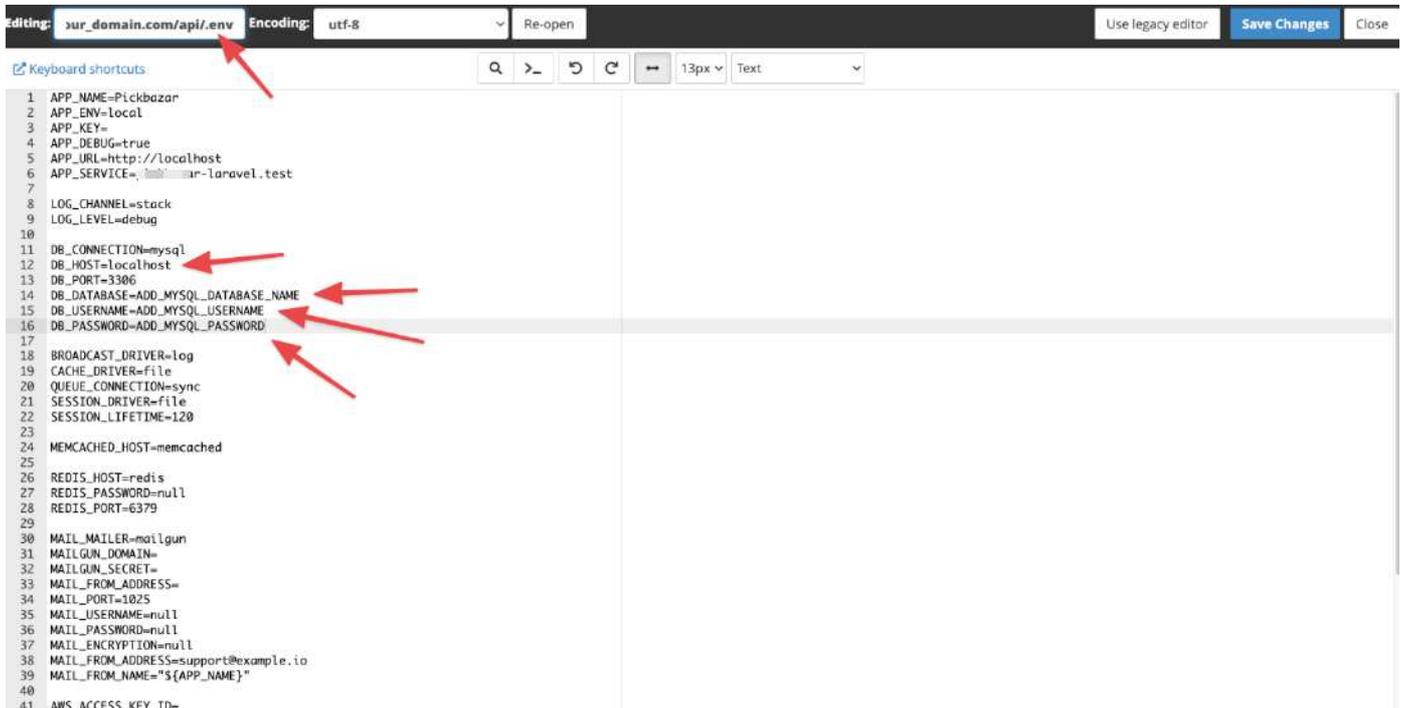
The screenshot shows the cPanel dashboard with several sections: WEB DISK, FTP Accounts, Backup, Backup Wizard, Git™ Version Control, and JetBackup. The DATABASIS section is expanded, showing phpMyAdmin, MySQL® Databases (highlighted with a red arrow), MySQL® Database Wizard, and Remote MySQL®. The DOMAINS section includes Site Publisher, Domains (highlighted with a red arrow), Addon Domains, Subdomains, Aliases, Redirects, Zone Editor, and Dynamic DNS. The METRICS section contains Visitors, Errors, Bandwidth, Raw Access, Awstats, Analog Stats, Webalizer, Webalizer FTP, Metrics Editor, and Resource Usage. The SECURITY section is partially visible at the bottom.

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

The screenshot shows a file manager interface with a table of files and folders. The table has columns for Name, Size, Last Modified, Type, and Permissions. The files listed include folders like `app`, `bootstrap`, `config`, `database`, `packages`, `public`, `resources`, `routes`, `storage`, `tests`, and files like `.editorconfig`, `.env`, `.env.example`, `.gitattributes`, `.gitignore`, `.styleci.yml`, `artisan`, `composer.json`, `composer.lock`, and `docker-compose.yml`. Red arrows point to the `.env` and `.env.example` files.

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	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,

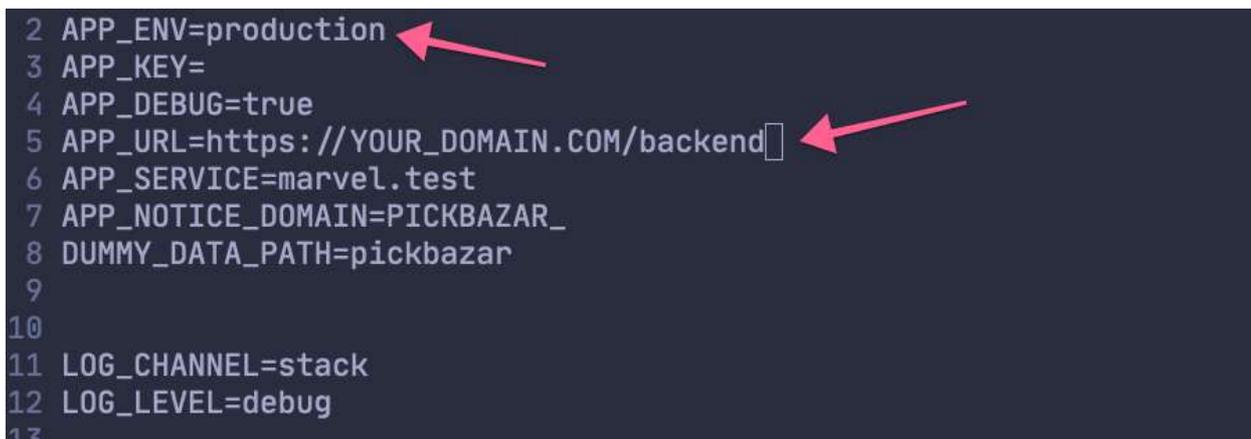


```

1 APP_NAME=Pickbazar
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6 APP_SERVICE=marvel-laravel.test
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](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`, and you'll get a webpage like this,

# Marvel Laravel

[GRAPHQL PLAYGROUND](#) [DOCUMENTATION](#) [SUPPORT](#) [CONTACT](#)

## 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 `pixier-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 `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"
```

```

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/fr
7      amework/graphql/**/*.d.ts\"",
8      "dev:rest": "next dev -p 3003",
9      "build:rest": "next build",
10     "codegen": "node -r dotenv/config $(yarn bin)/graphql-let",
11     "dev:gql": "yarn codegen && next dev -p 3001",
12     "build:gql": "yarn codegen && next build",
13     "start": "next start",
14     "lint": "next lint"
15   },
16   "dependencies": {
17     "@apollo/client": "^3.4.16",
18     "@headlessui/react": "^1.4.1",
19     "@hookform/resolvers": "^2.8.2",
20     "@reach/portal": "^0.16.0",
21     "@stripe/react-stripe-js": "^1.5.0",
22     "@stripe/stripe-js": "^1.20.2",
23     "apollo-upload-client": "^16.0.0",
24     "axios": "^0.23.0",
25     "body-scroll-lock": "^4.0.0-beta.0",
26     "camelcase-keys": "^7.0.1",
27     "classnames": "^2.3.1",
28     "dayjs": "^1.10.7",
29     "deepmerge": "^4.2.2",
30     "framer-motion": "^4.1.17",
31   }
32 }

```

## 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') {
      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 `pixier-laravel -> admin` folder again

To install all the npm packages run this command,

```
yarn
```

Again,

go to your `pixier-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 you `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,

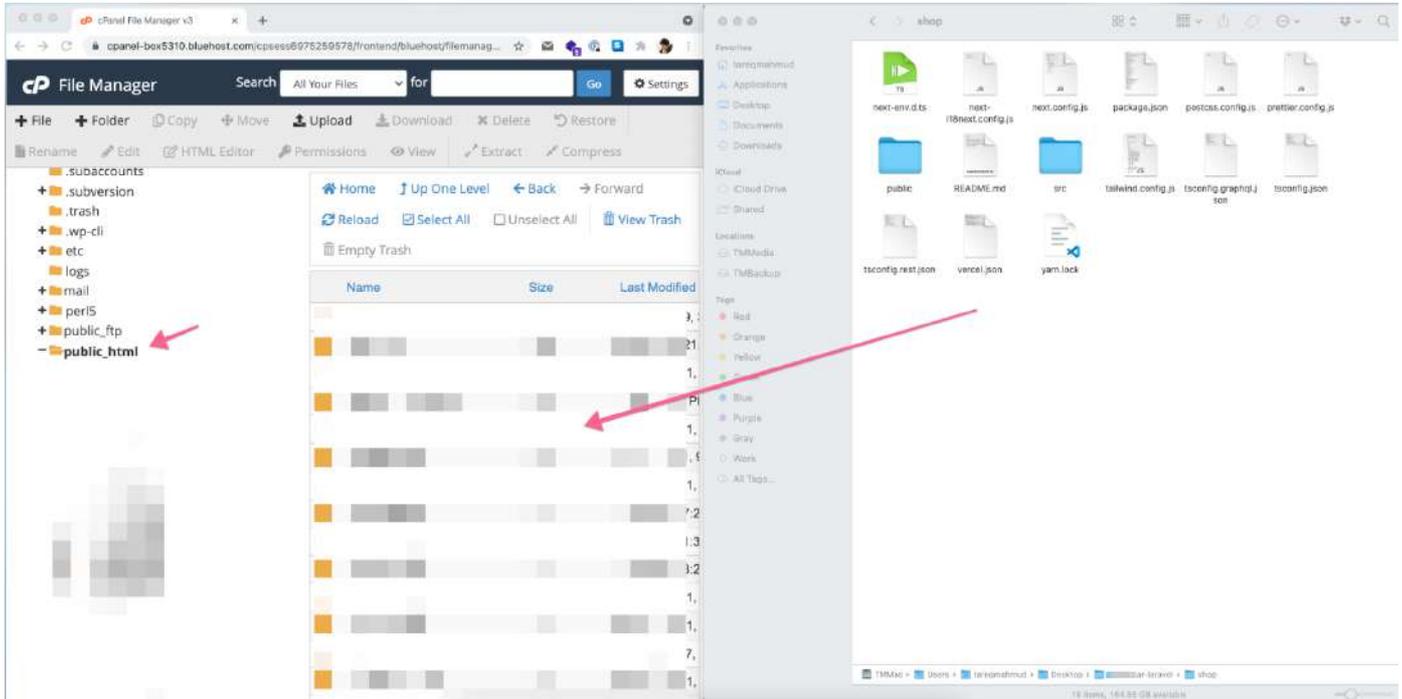
```
yarn build
```

and run,

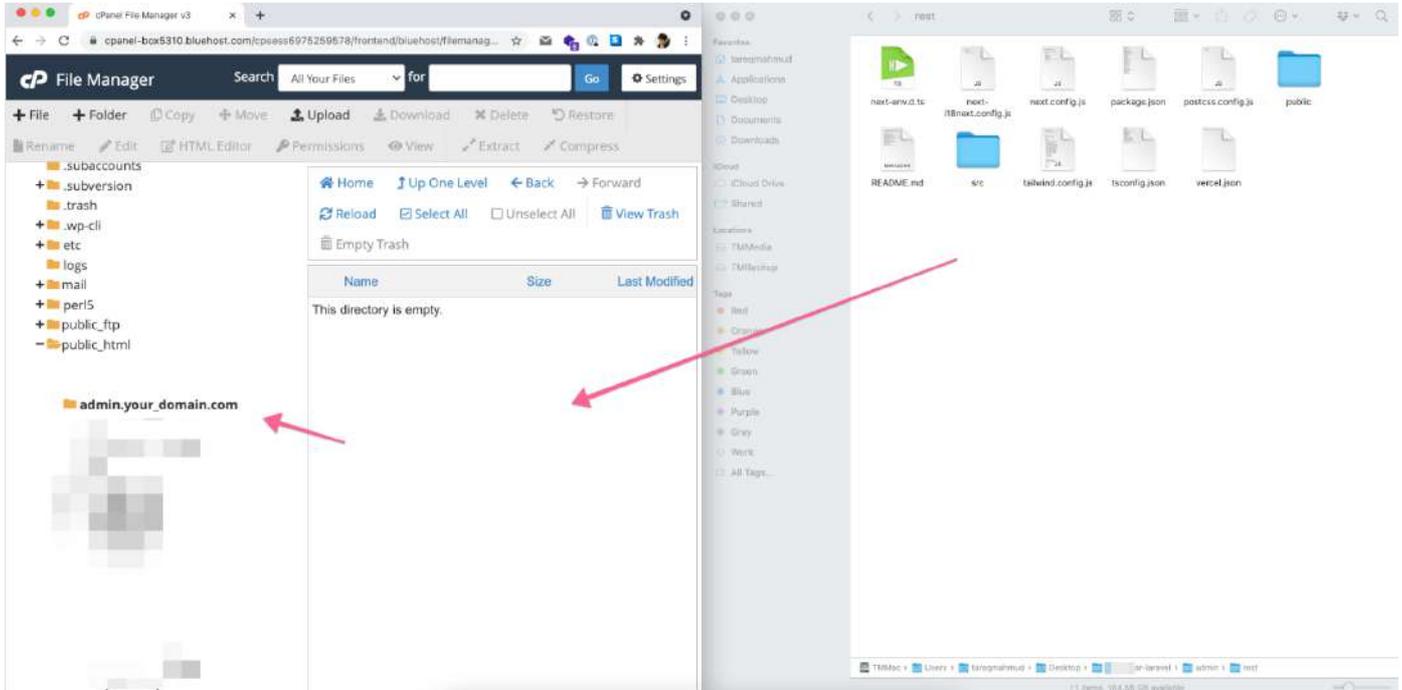
After build the project upload the

- `shop` to `root_domain` -> `public_html` folder
- `admin-rest` to `admin.your_domain.com` folder

`shop`,



shop-admin,

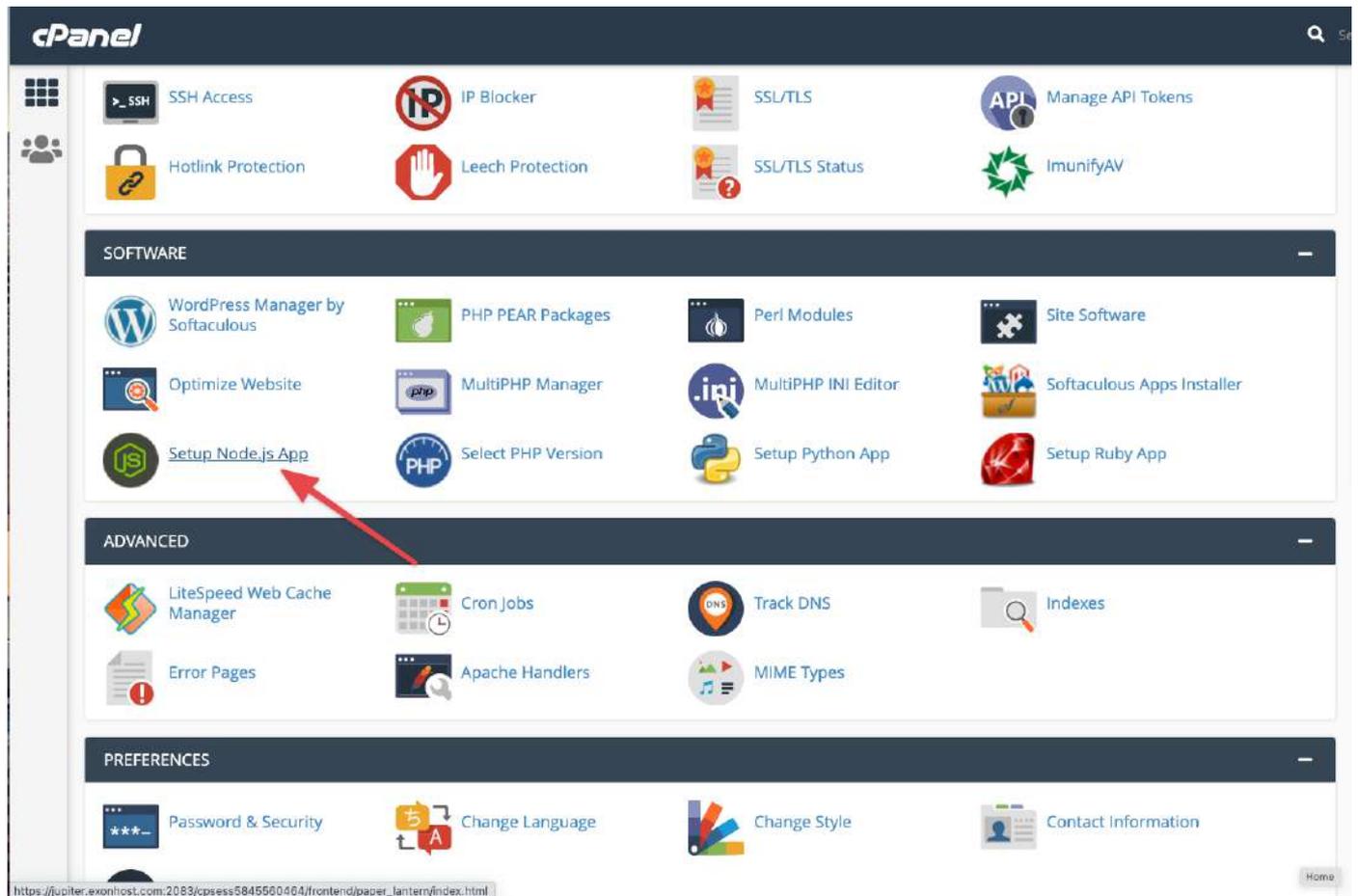


### Install NodeJs Project

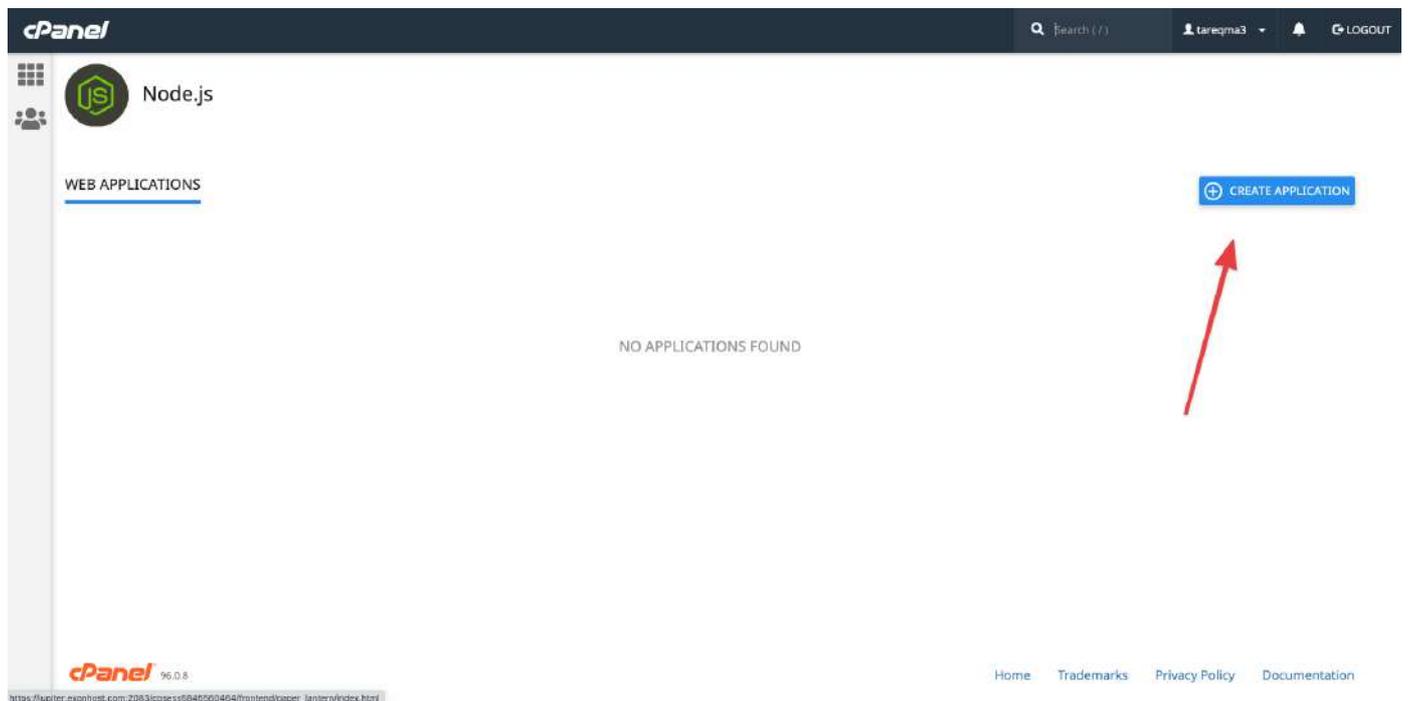
We'll run both `shop` and `admin` using the cPanel NodeJs application in this step.

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

For `shop`,



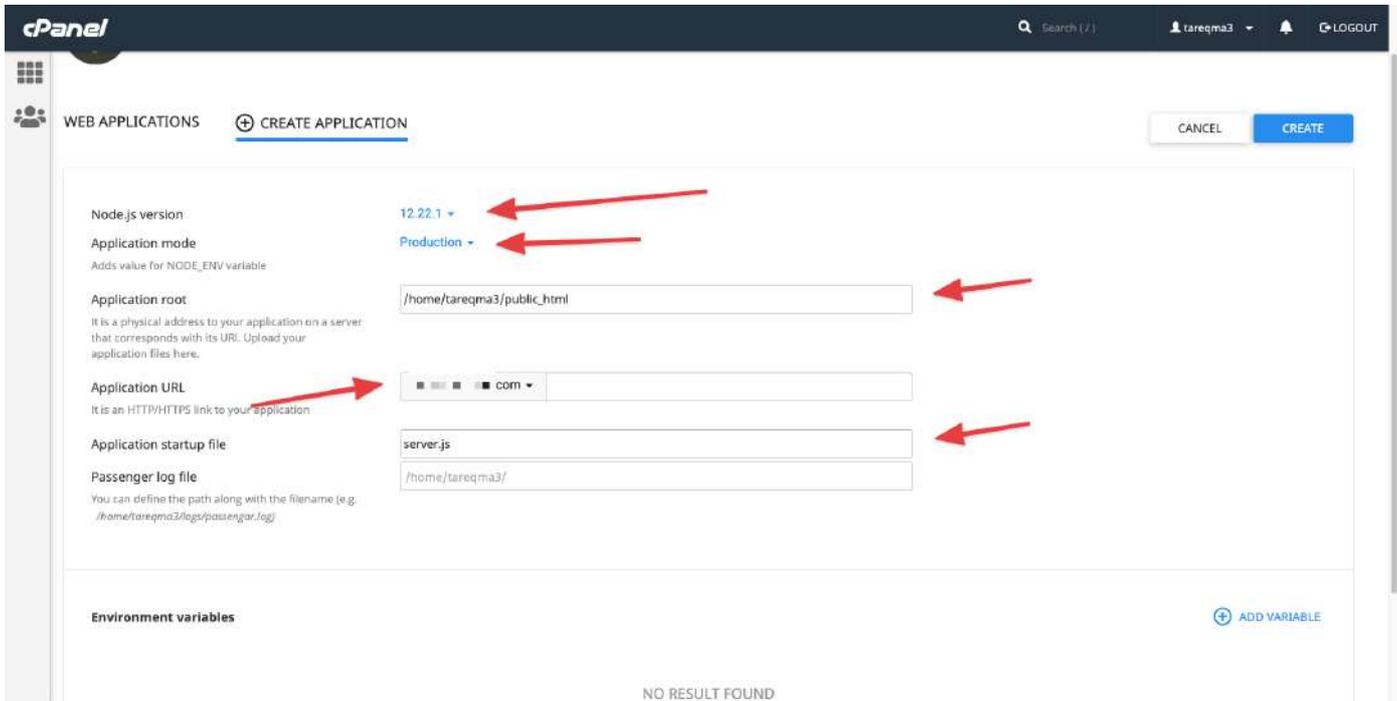
The screenshot shows the cPanel dashboard interface. The top navigation bar includes the cPanel logo and a search icon. Below the navigation bar, there are several utility icons: SSH Access, IP Blocker, SSL/TLS, Manage API Tokens, Hotlink Protection, Leech Protection, SSL/TLS Status, and ImunifyAV. The main content area is divided into three sections: SOFTWARE, ADVANCED, and PREFERENCES. The SOFTWARE section contains icons for 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 ADVANCED section includes LiteSpeed Web Cache Manager, Cron Jobs, Track DNS, Indexes, Error Pages, Apache Handlers, and MIME Types. The PREFERENCES section includes Password & Security, Change Language, Change Style, and Contact Information. The bottom of the dashboard shows the URL: [https://jupiter.exonhost.com:2083/cpsess5845560464/frontend/paper\\_lantern/index.html](https://jupiter.exonhost.com:2083/cpsess5845560464/frontend/paper_lantern/index.html) and a Home button.



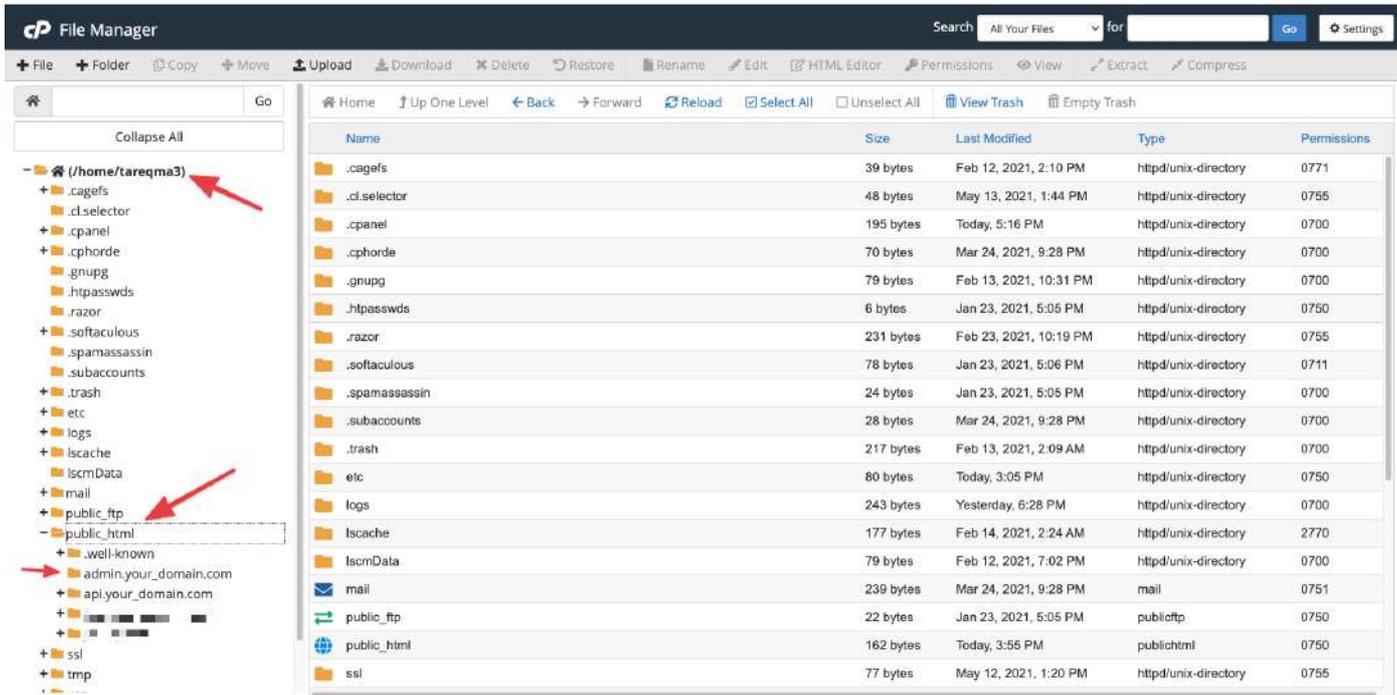
The screenshot shows the cPanel Node.js application management page. The top navigation bar includes the cPanel logo, a search icon, the user name 'tareqma3', a notification bell, and a LOGOUT button. The main content area is titled 'Node.js' and 'WEB APPLICATIONS'. Below the title, there is a 'CREATE APPLICATION' button (highlighted with a red arrow) and the text 'NO APPLICATIONS FOUND'. The bottom of the page shows the cPanel logo and version '96.0.8', and the URL: [https://jupiter.exonhost.com:2083/cpsess5845560464/frontend/paper\\_lantern/index.html](https://jupiter.exonhost.com:2083/cpsess5845560464/frontend/paper_lantern/index.html). The footer includes links for Home, Trademarks, Privacy Policy, and Documentation.

Now,

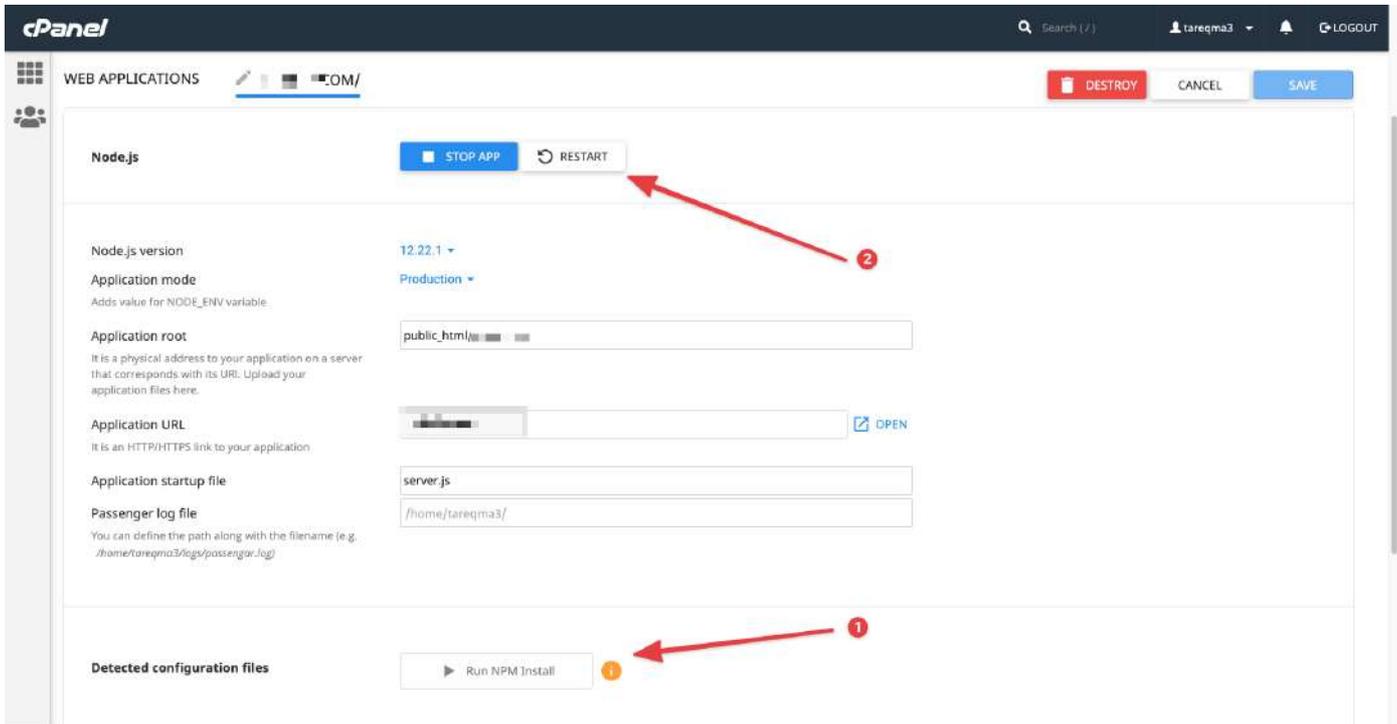
- Select NodeJS version
- Make environment `production`.
- Set Application Root
- And application startup file as `server.js`



You can get the Application Path from your cPanel file manager



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



For `admin`,

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

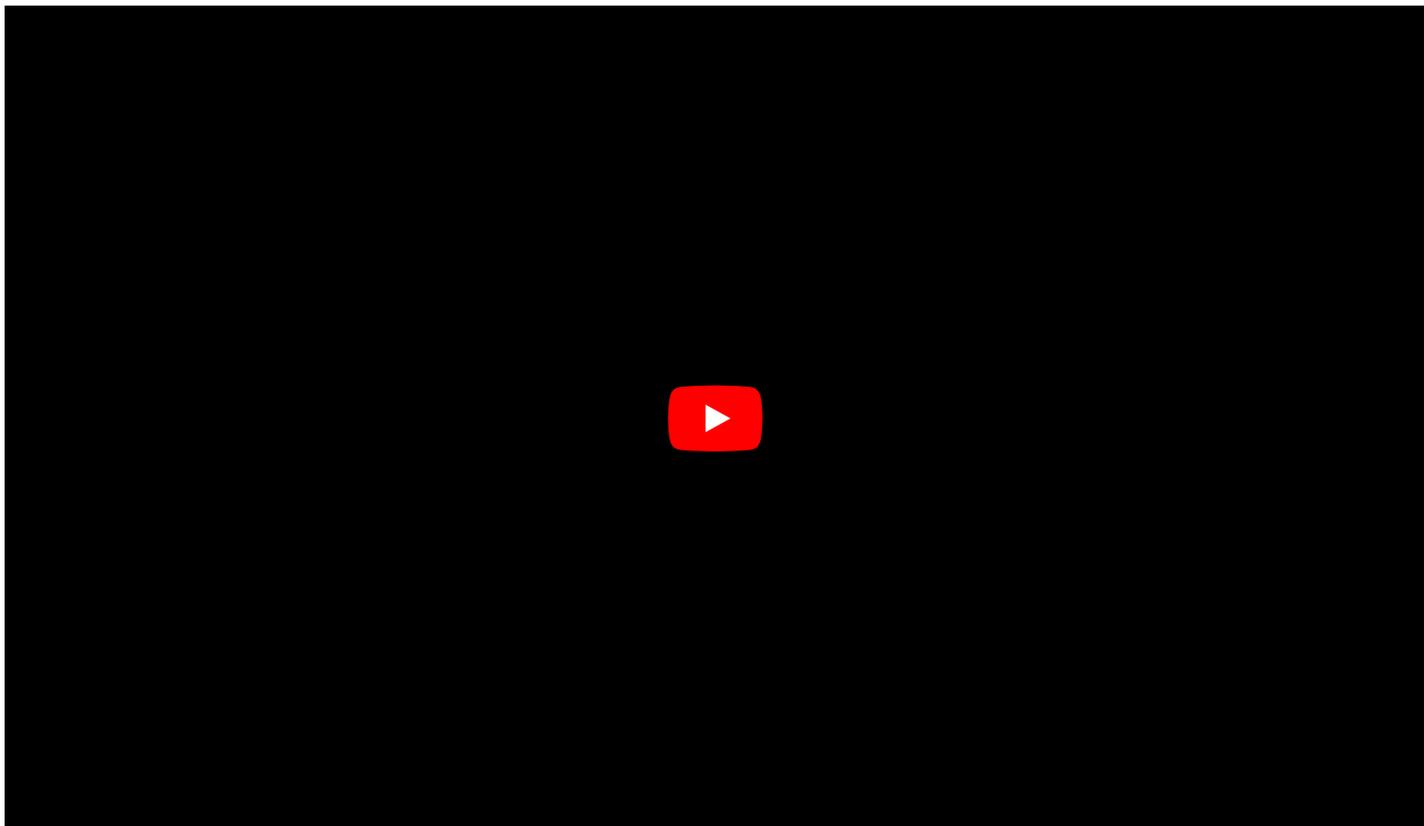
After installing and run both NodeJS application, you can access your domain to check Pixier,

Thank You!

Vercel

[vercel.com](https://vercel.com)

If you want to host the template in vercel.com then follow this documentation with this tutorial,



It's not possible to host the API to vercel. Vercel doesn't support laravel API deployment. So you've to host the API on a separate server. We suggest you create a VPS server and host the API there.

To host the API to a separate server, follow this doc from [Beginning to Install API](#)

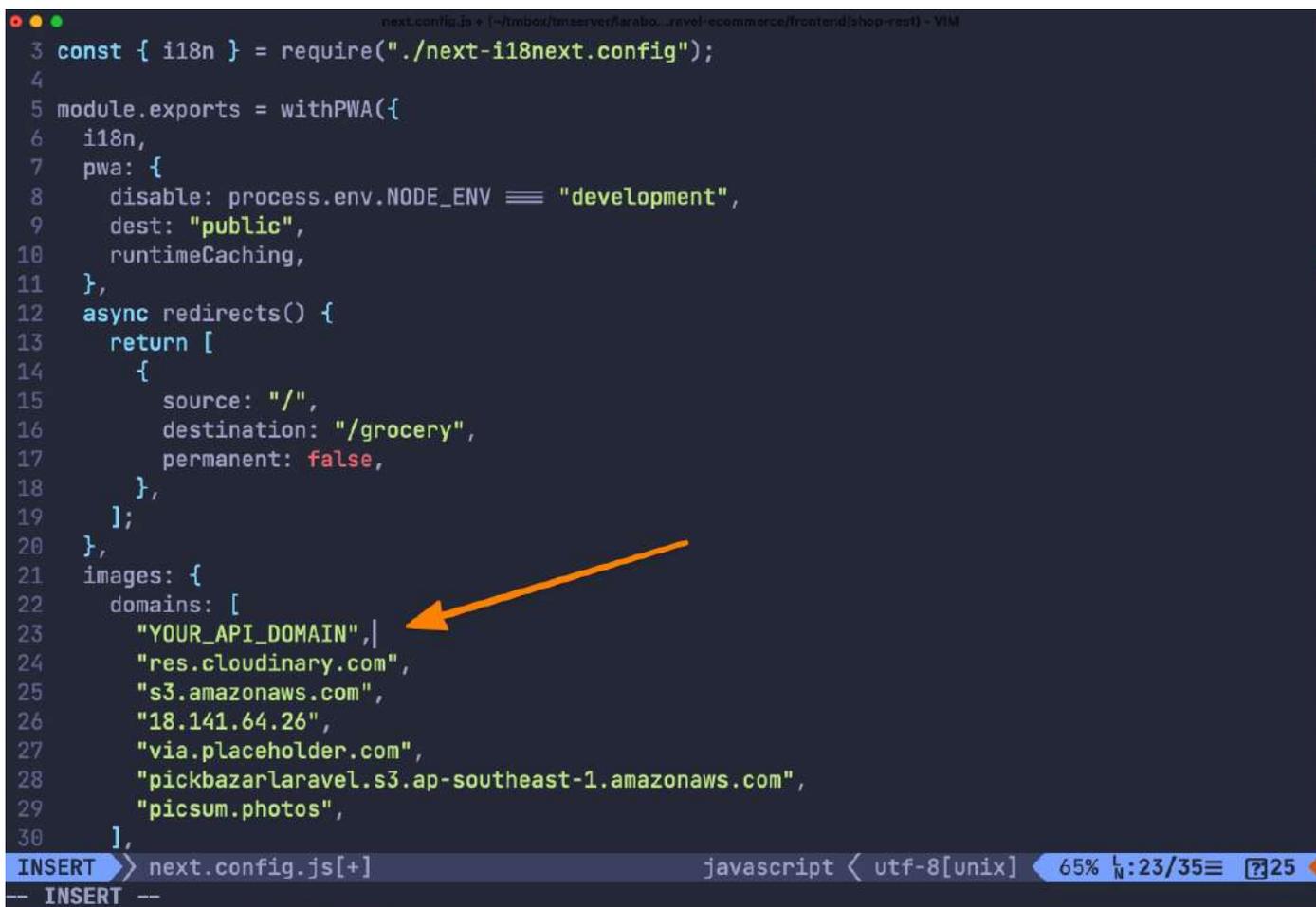
After host the API successfully follow this part,

## Frontend

Now for frontend add API URL and other necessary config details to,

```
shop -> vercel.json
```

Open `shop -> next.config.js` and add your domain to `images` object



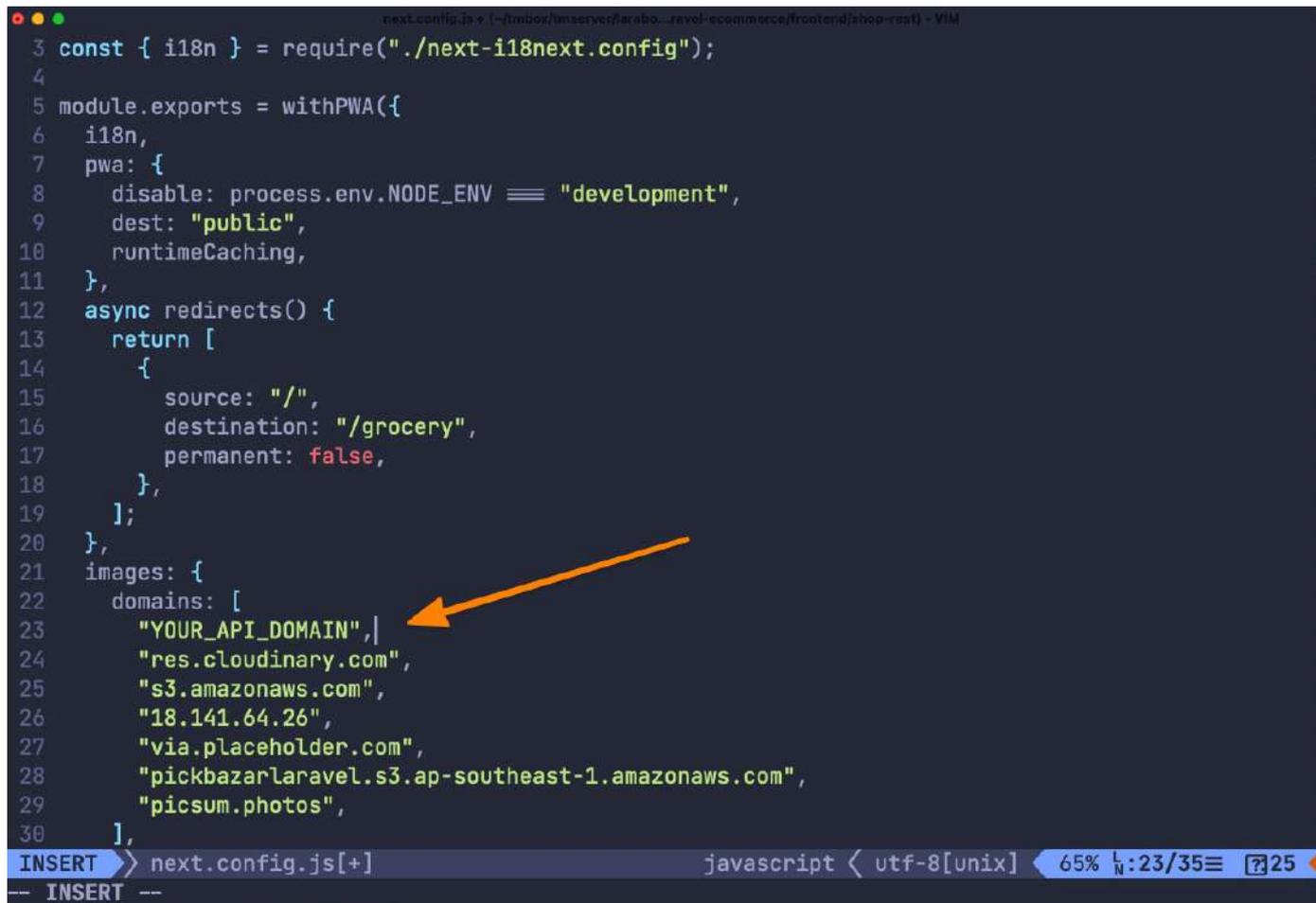
```
3 const { i18n } = require("./next-i18next.config");
4
5 module.exports = withPWA({
6   i18n,
7   pwa: {
8     disable: process.env.NODE_ENV === "development",
9     dest: "public",
10    runtimeCaching,
11  },
12  async redirects() {
13    return [
14      {
15        source: "/",
16        destination: "/grocery",
17        permanent: false,
18      },
19    ];
20  },
21  images: {
22    domains: [
23      "YOUR_API_DOMAIN",
24      "res.cloudinary.com",
25      "s3.amazonaws.com",
26      "18.141.64.26",
27      "via.placeholder.com",
28      "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
29      "picsum.photos",
30    ],
31  },
32 });
```

If your API is hosted on a subdomain, then add that subdomain with root domain on `next.config.js`

```
admin -> vercel.json
```

Open `admin -> next.config.js` and add your domain to `images` object

```
next.config.js + [~/tmpbox/tmpserver/forabo...revel-ecommerce/frontend/shop-rest] - YIM
3 const { i18n } = require("./next-i18next.config");
4
5 module.exports = withPWA({
6   i18n,
7   pwa: {
8     disable: process.env.NODE_ENV === "development",
9     dest: "public",
10    runtimeCaching,
11  },
12  async redirects() {
13    return [
14      {
15        source: "/",
16        destination: "/grocery",
17        permanent: false,
18      },
19    ];
20  },
21  images: {
22    domains: [
23      "YOUR_API_DOMAIN",
24      "res.cloudinary.com",
25      "s3.amazonaws.com",
26      "18.141.64.26",
27      "via.placeholder.com",
28      "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
29      "picsum.photos",
30    ],
31  },
32 });
```



If your API is hosted on a subdomain, then add that subdomain with root domain on `next.config.js`

after that, install `vercel cli` on your computer using this command,

```
npm i -g vercel
```

After that, log in to vercel using this command,

```
vercel login
```

Then go to the `shop` directory and use this command to deploy,

```
vercel
```

Similarly, go to the `admin` directory and use this command to deploy,

```
vercel
```

For more details,

Please follow [nextjs deployment docs](#):

## Virtual Private Server (Manual)

If you want to deploy the app using automated script then [follow this](#)

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.

Please connect your `domain` with `server`. We don't recommend/support deployment the project via `IP`.

Please use the ubuntu 20.04 LTS version for this documentation.

With this tutorial and documentation, you can install Pixier 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.

Please follow this video with the documentation, and it'll make the installation process relatively easy.



## 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 pixier app. To install NodeJS and npm, run this command on your terminal,

```
sudo apt-get update
```

```
curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs
```

### Install Yarn

Pixier 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

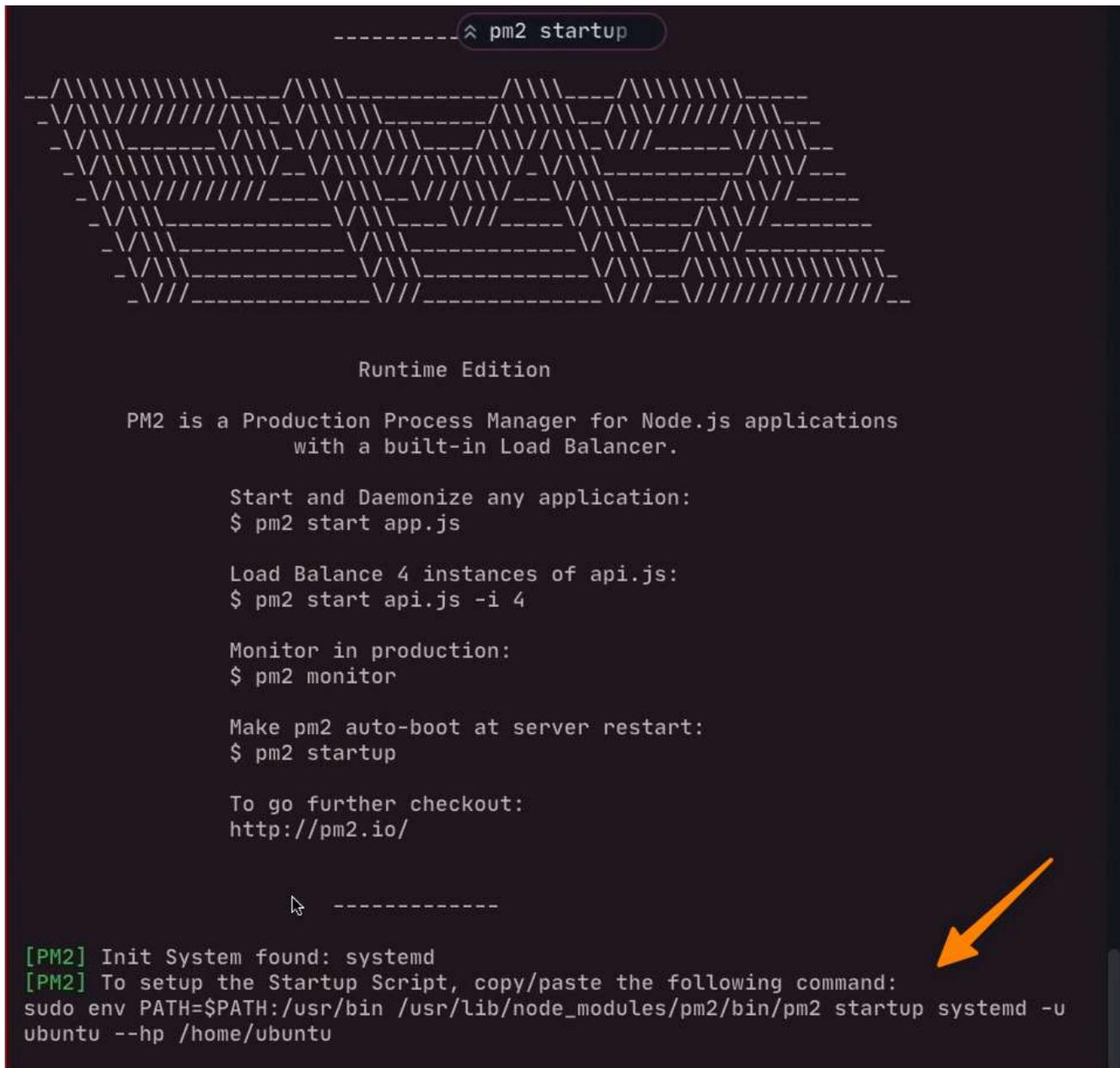
Now we will install **PM2**, which is a process manager for Node.js applications. **PM2** provides an easy way to manage and daemonize applications (run them in the background as a service). To install **PM2** use this command,

```
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
```

After running this command, it'll provide you a command. Just copy -> paste -> enter to execute that command.



```
----- ^ pm2 startup
-----
Runtime Edition

PM2 is a Production Process Manager for Node.js applications
with a built-in Load Balancer.

Start and Daemonize any application:
$ pm2 start app.js

Load Balance 4 instances of api.js:
$ pm2 start api.js -i 4

Monitor in production:
$ pm2 monitor

Make pm2 auto-boot at server restart:
$ pm2 startup

To go further checkout:
http://pm2.io/

-----
[PM2] Init System found: systemd
[PM2] To setup the Startup Script, copy/paste the following command:
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u
ubuntu --hp /home/ubuntu
```

## 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 pixar

### 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
```

Add PPA to get the specific php version

```
sudo add-apt-repository ppa:ondrej/php
```

```
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:

```
Status: active
```

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

```
sudo apt install php8.1-fpm php8.1-mysql
```

```
sudo apt install php8.1-mbstring php8.1-xml php8.1-bcmath php8.1-simplexml php8.1-intl php8.1-gd php8.1-curl php8.1-zip
php8.1-gmp
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d64d92e6563bbde02') { echo 'Installer
verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

```
sudo mv composer.phar /usr/bin/composer
```

### Step 6 - Create MySQL Database & User

```
sudo mysql
```

```
CREATE DATABASE pixer;

CREATE USER 'pixer_user'@'%' IDENTIFIED WITH mysql_native_password BY 'pixer1';
```

```
GRANT ALL ON pixer.* TO 'pixer_user'@'%';  
  
FLUSH PRIVILEGES;
```

We use MySQL user name `pixer_user` and MYSQL password `pixer1`. 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-laravel
```

Then, go to your `local computer`

1. Extract the `zip` package that you download from `CodeCanyon`.
2. On that folder, you'll get another `zip` called `pixer-laravel`.
3. Extract that `zip` package.
4. On that folder, you'll get a folder called `pixer-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;  
  
    client_max_body_size 256M;  
  
    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-laravel/pixer-api/public;  
        try_files $uri $uri/ @backend;  
        location ~ /\.php$ {  
            include fastcgi_params;  
            fastcgi_param SCRIPT_FILENAME $request_filename;
```

```
        fastcgi_pass    unix:/run/php/php8.1-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;
}

error_page 404 /index.php;

location ~ /\.php$ {
    fastcgi_pass    unix:/var/run/php/php8.1-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

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-laravel/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
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

---

## Marvel Laravel

[GRAPHQL PLAYGROUND](#) [DOCUMENTATION](#) [SUPPORT](#) [CONTACT](#)

### 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.

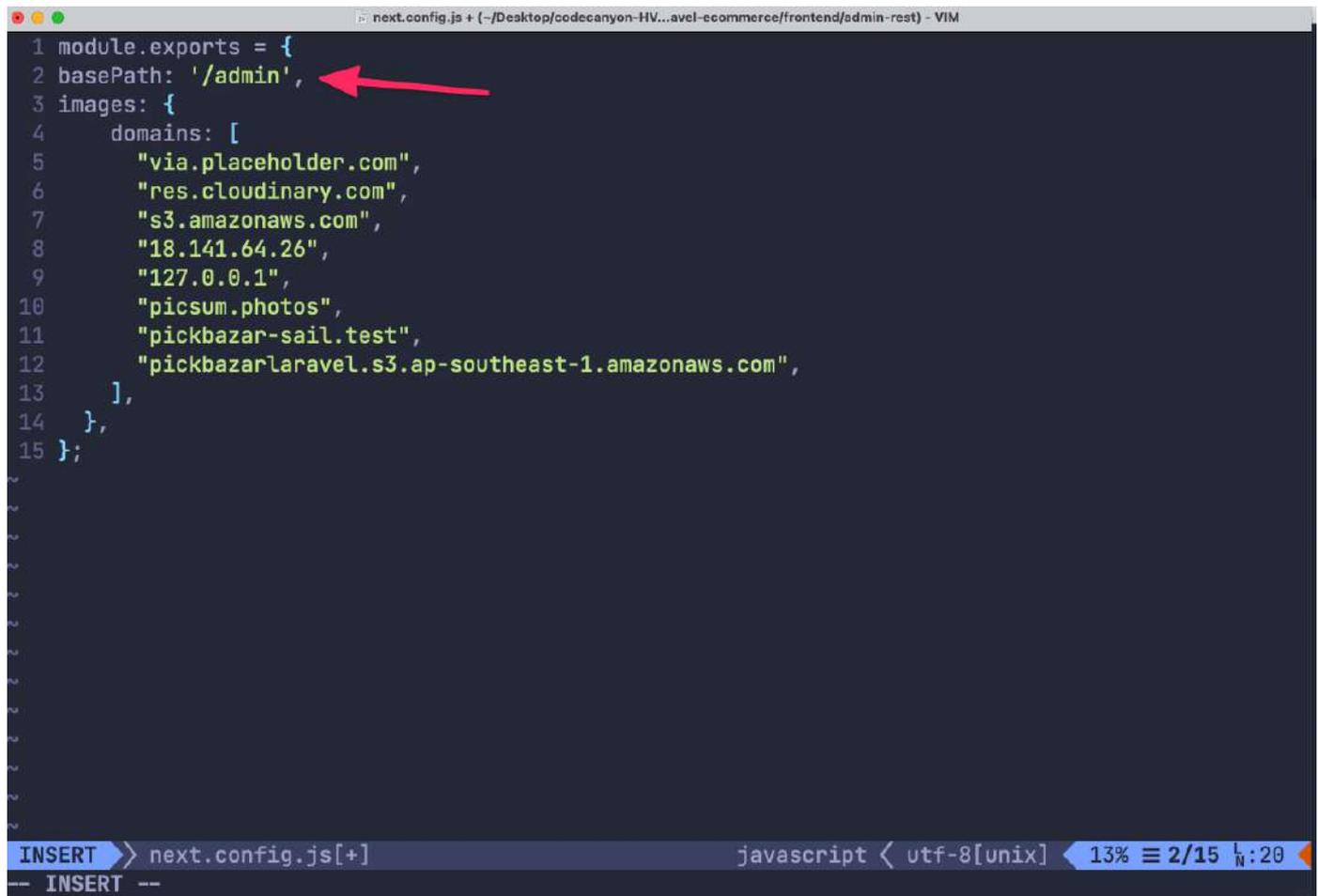
We'll suggest you build the frontend part on your computer and then upload the build file to the server.

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'`



```
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 };
```

## 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 `shop` -> `.env.template` to `shop` -> `.env`.

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

and use

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

After that, copy the sample `admin` -> `.env.template` to `admin` -> `.env`.

and use

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

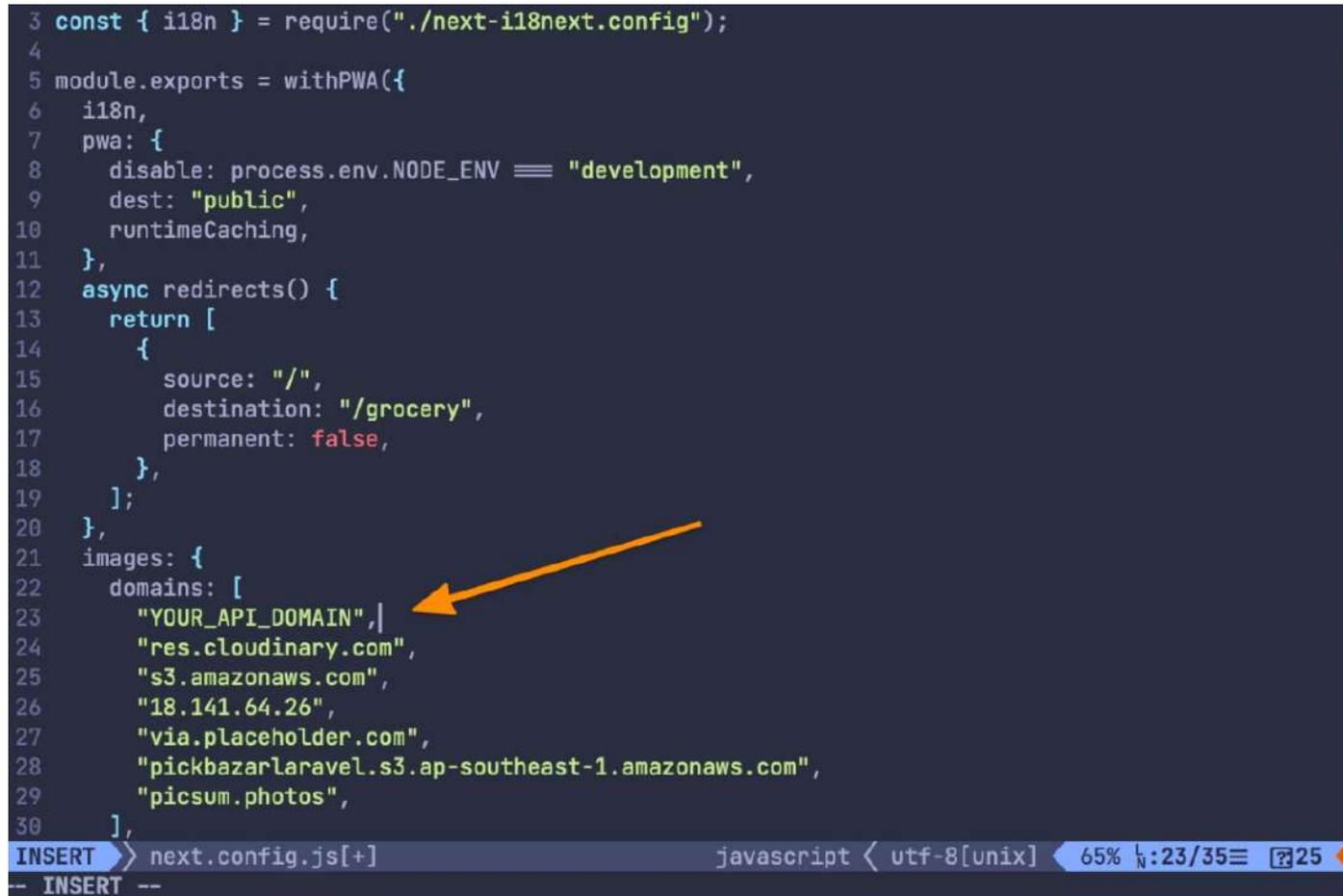
Then open `shop -> next.config.js` and `admin -> next.config.js`

and add your domain to `images` object

```

3 const { i18n } = require("../next-i18next.config");
4
5 module.exports = withPWA({
6   i18n,
7   pwa: {
8     disable: process.env.NODE_ENV === "development",
9     dest: "public",
10    runtimeCaching,
11  },
12  async redirects() {
13    return [
14      {
15        source: "/",
16        destination: "/grocery",
17        permanent: false,
18      },
19    ];
20  },
21  images: {
22    domains: [
23      "YOUR_API_DOMAIN",
24      "res.cloudinary.com",
25      "s3.amazonaws.com",
26      "18.141.64.26",
27      "via.placeholder.com",
28      "pickbazar.laravel.s3.ap-southeast-1.amazonaws.com",
29      "picsum.photos",
30    ],
31  },
32 });

```



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 `pixier-laravel -> admin` folder,

and run,

```
yarn build
```

again,

Now go to the `pixier-laravel -> shop` folder,

and run,

```
yarn build
```

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

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

## Install FrontEnd And Run

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

```
yarn
```

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

```
yarn
```

Run frontend app

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

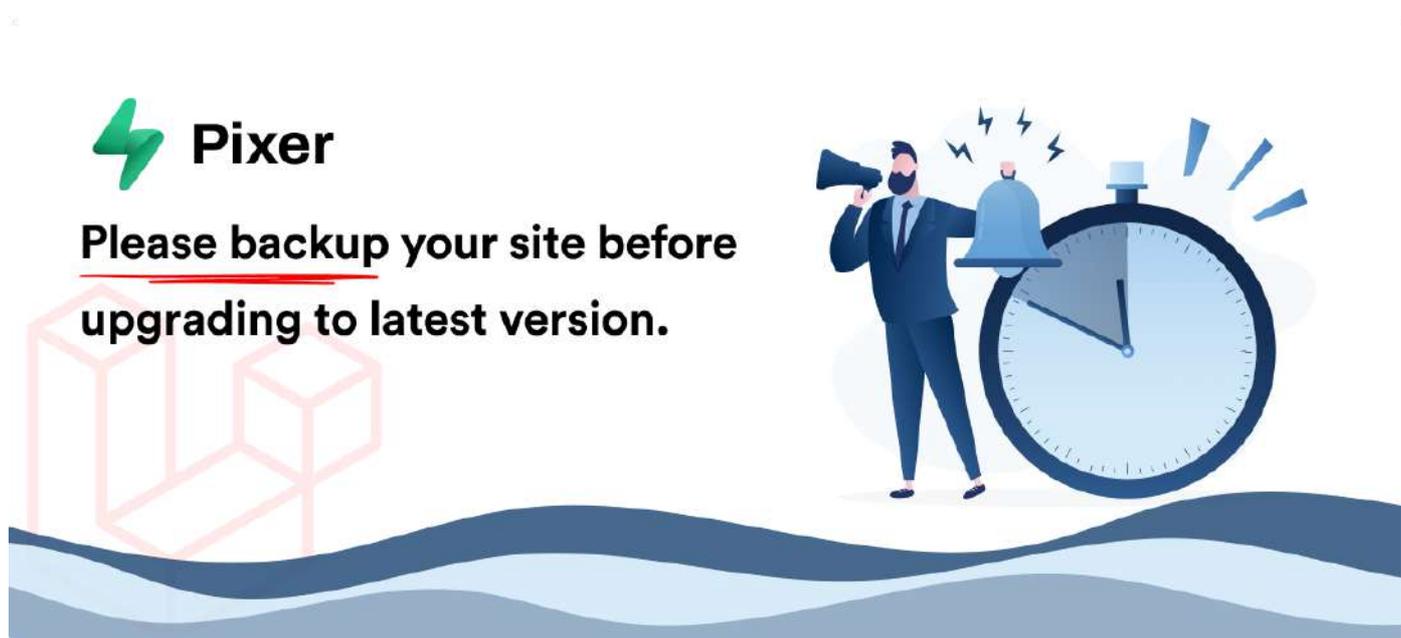
```
pm2 --name shop-rest start yarn -- run start
```

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.

## Update



### Pixier Update - Virtual Private Server

If you follow this [Virtual Private Server](#) docs to host your site, then follow this documentation to update Pixier 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 `pixier` directory on your server,

```
cd /var/www/pixier
```

2. Then initialize a git on that folder,

```
git init
```

3. 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 .
```

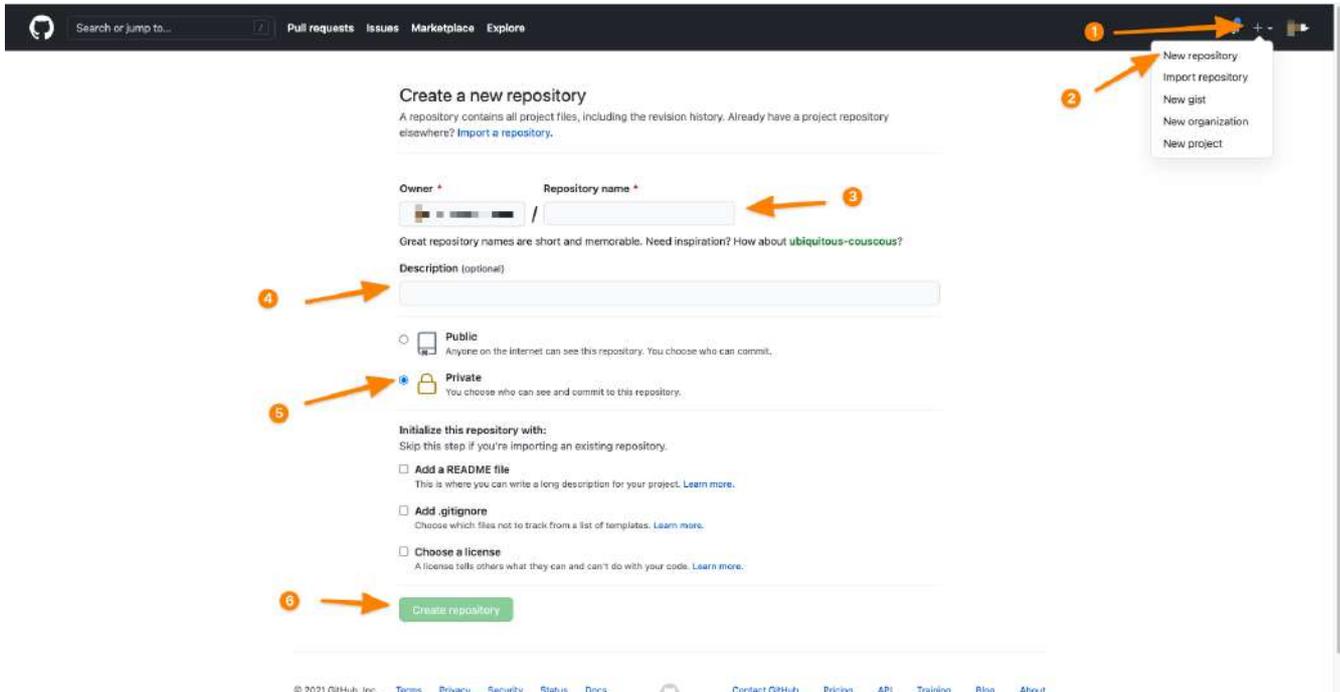
```
git commit -m "Initial commit"
```

Create a separate branch to maintain the updated code.

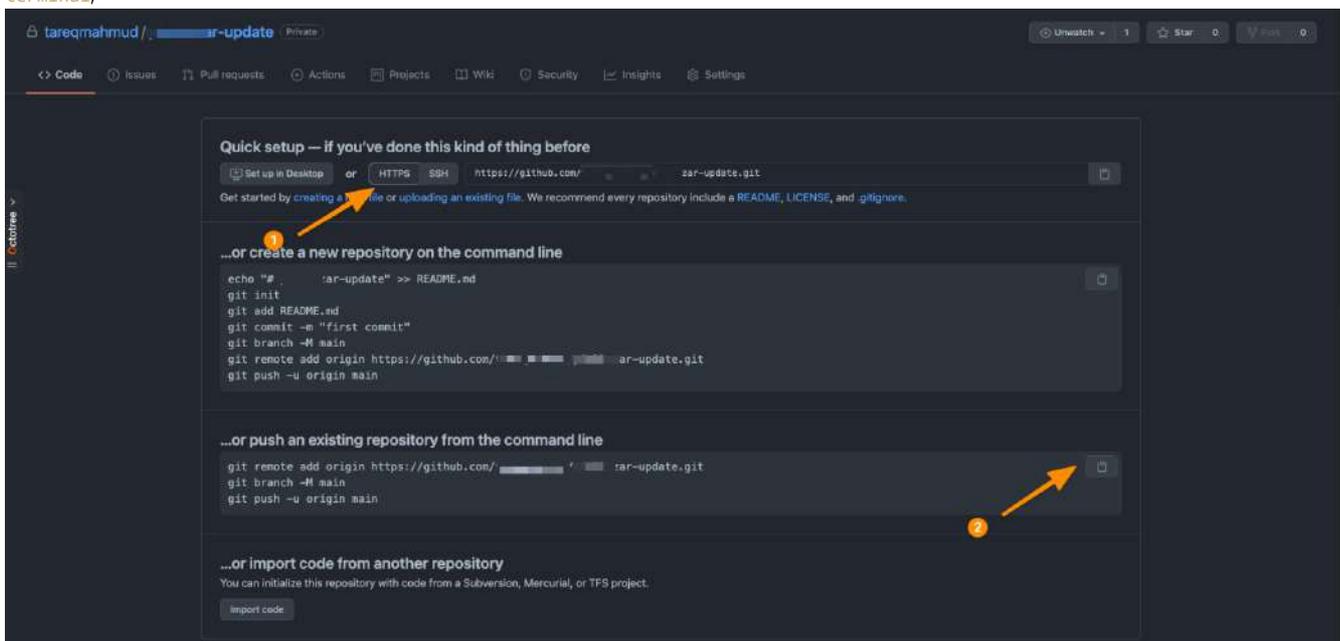
```
git branch pixer
```

## Git & Github

1. At first, go to <https://github.com/> and create an account first. If you already have an account then **Sign In** on that account.
2. Then create a new repository,



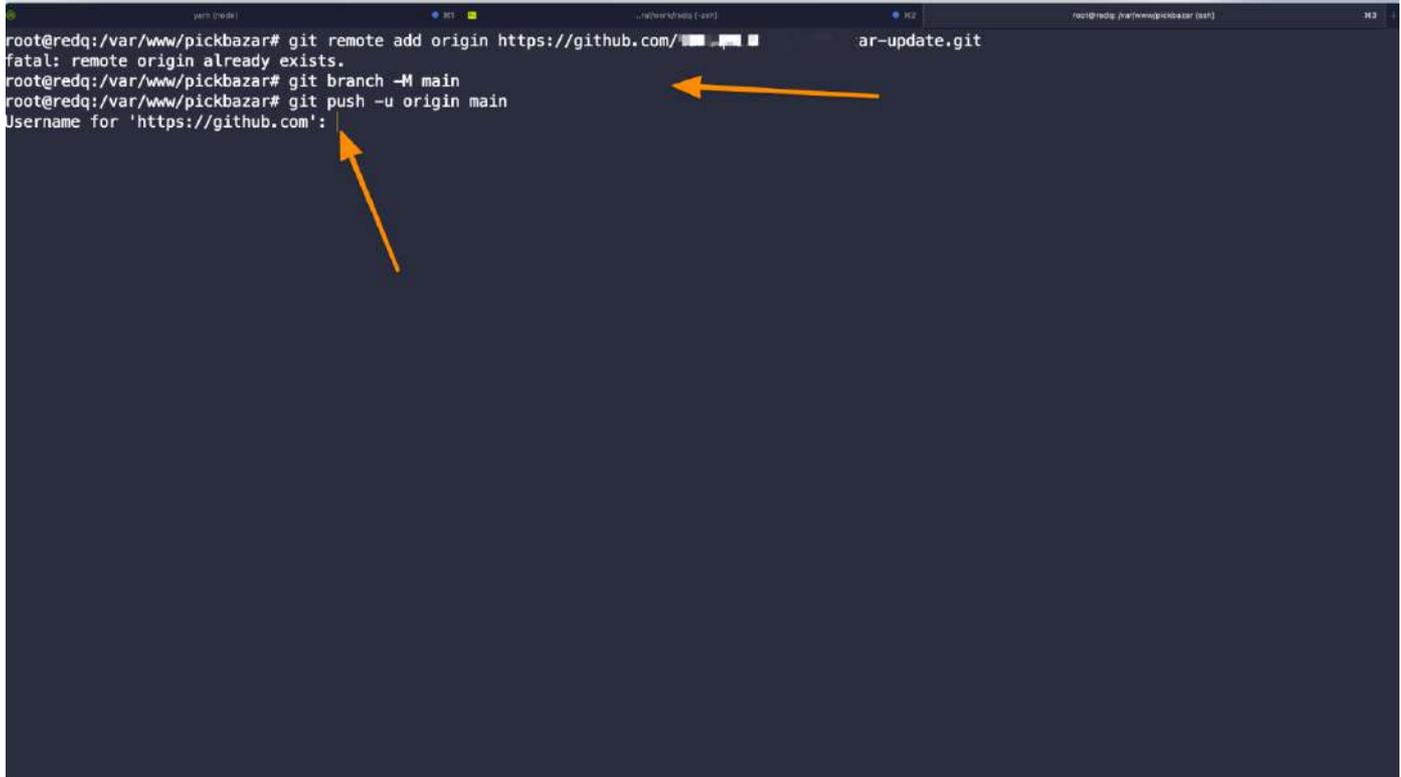
3. After creating the repository you'll get a page like this and from this page copy the **second command block** and go to your **server using SSH & terminal**,



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,

1. Clone that [GitHub repository](#) to your [local computer](#)
2. Download [update package](#) from [CodeCanyon](#)
3. Open your terminal and clone that repository to your computer
4. Checkout to [pixer](#) branch

```
git checkout -b pixer
```

5. replace [repository](#) file with downloaded [pixer-laravel](#) folder content.
6. Then use this command to add all files to git

```
git add .
```

```
git commit -m "Update Pixier API"
```

7. 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:clear
```

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.

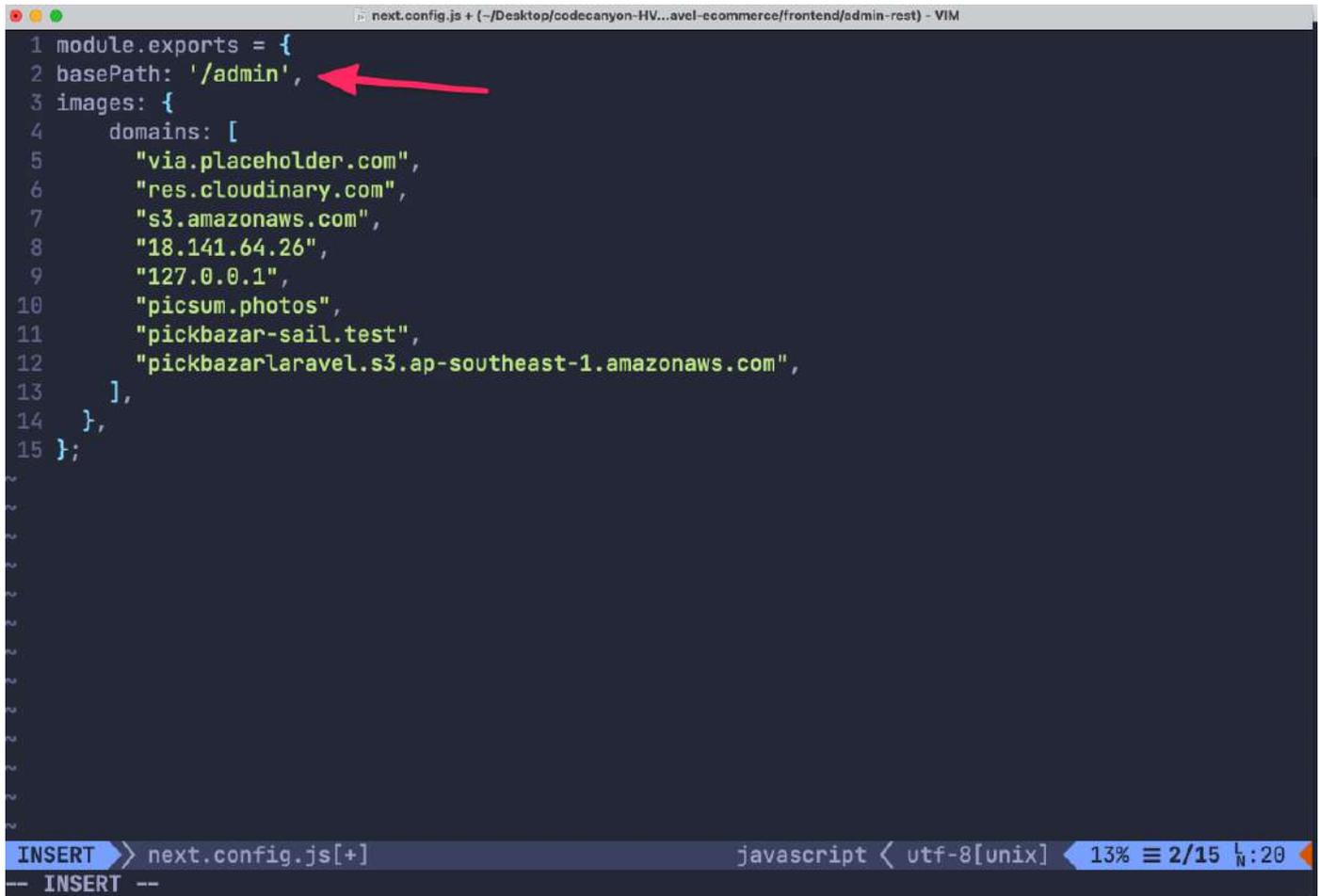
We'll suggest you build the frontend part on your computer and then move the build file to the server using git and github.

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       "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 };
```

## Install & Build

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

To install all the npm packages run this command,

```
yarn
```

Again,

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

To install all the npm packages run this command,

```
yarn
```

## Build the project

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

To install all the npm packages run this command,

```
yarn build
```

Again,

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

To install all the npm packages run this command,

```
yarn
```

Again,

go to your `pixier-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 and Analytics

### SEO

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

```
shop -> src -> components -> seo
```

With that boilerplate, check `Next SEO` docs to use SEO on your pixier 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](#).

---

## Laravel API

## 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.

### Installation Windows

#### Prerequisites

- PHP 8.1
- 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

#### Resources you might need

1. <https://laravel.com/docs/8.x>
2. <https://lighthouse-php.com/5/getting-started/installation.html>
3. <https://github.com/spatie/laravel-medialibrary>
4. <https://github.com/andersao/l5-repository>
5. <https://spatie.be/docs/laravel-permission/v3/introduction>

#### Packages we have used

```
"laravel/socialite": "5.5.2",
"laravel/tinker": "2.7.2",
"messagebird/php-rest-api": "3.1.2",
"symfony/http-client": "6.0.9",
"psr/log": "2.0.0",
"symfony/mailgun-mailer": "6.0.7",
"twilio/sdk": "6.40.0",
"srmklive/paypal": "3.0",
"mll-lab/graphql-php-scalars": "5.4.0",
"nuwave/lighthouse": "5.57.0",
"laravel/legacy-factories": "1.3.0",
"cviebrock/eloquent-sluggable": "9.0.0",
"laravel/sanctum": "2.15.1",
"mll-lab/laravel-graphql-playground": "2.6.0",
"prettus/l5-repository": "2.8.0",
"spatie/laravel-medialibrary": "10.4.1",
"spatie/laravel-permission": "5.5.5",
"php-http/guzzle7-adapter": "1.0.0",
"bensampo/laravel-enum": "5.3.1",
"league/flysystem-aws-s3-v3": "3.1.1",
"spatie/laravel-newsletter": "4.11.0",
"spatie/period": "2.3.3",
"kodeine/laravel-meta": "2.1.0",
"maatwebsite/excel": "3.1.44",
"niklasravnsborg/laravel-pdf": "4.1.0",
"cknow/laravel-money": "7.0.0",
"mollie/laravel-mollie": "2.19.1",
"razorpay/razorpay": "2.8.4",
"unicodeveloper/laravel-paystack": "1.0.8",
"stripe/stripe-php": "8.11.0",
"stevebauman/purify": "5.1.1",
```

## 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=pixer_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

#### 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 `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 publishes 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>

## 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 `15-repository` (<https://github.com/andersao/15-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` folder.

## 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 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.

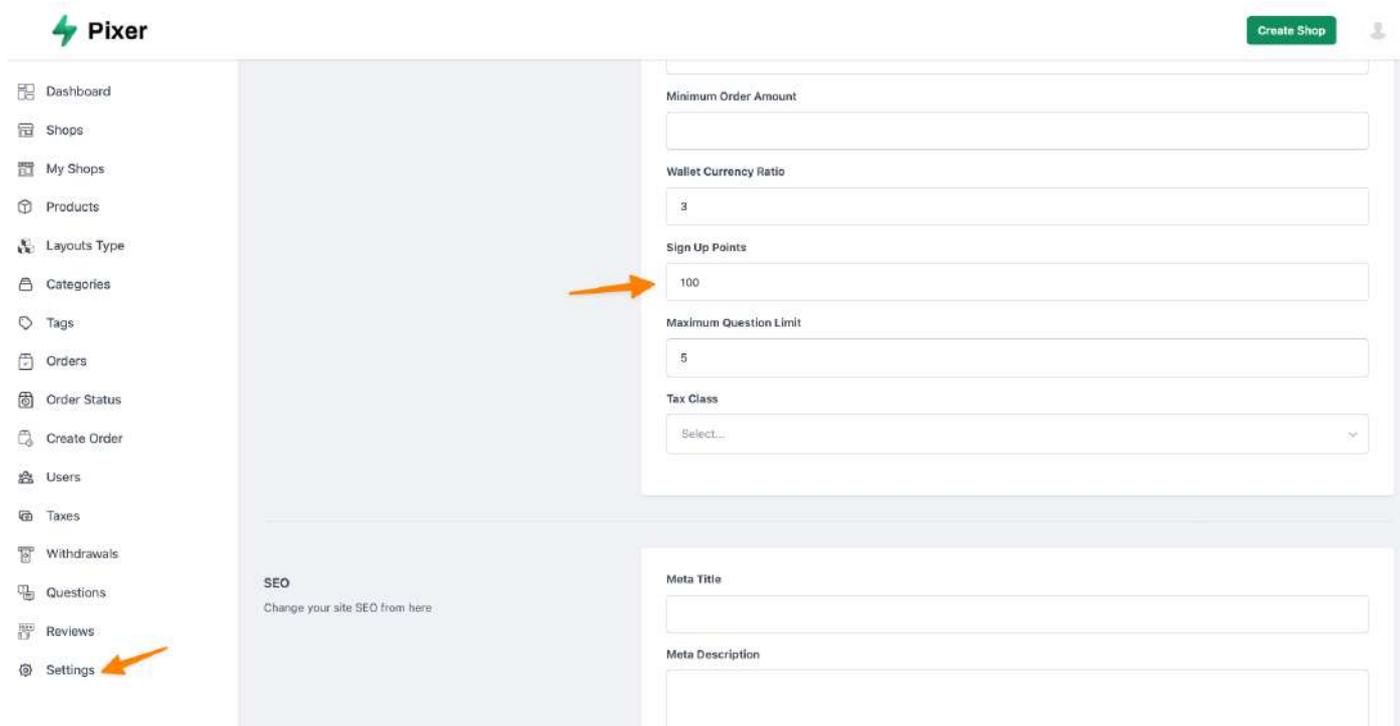
## Wallet

The wallet is a virtual currency that a customer can use for purchase items. On pixer, there are two ways that can be used to generate wallet points for a customer.

1. Sign up points.
2. Manually by admin

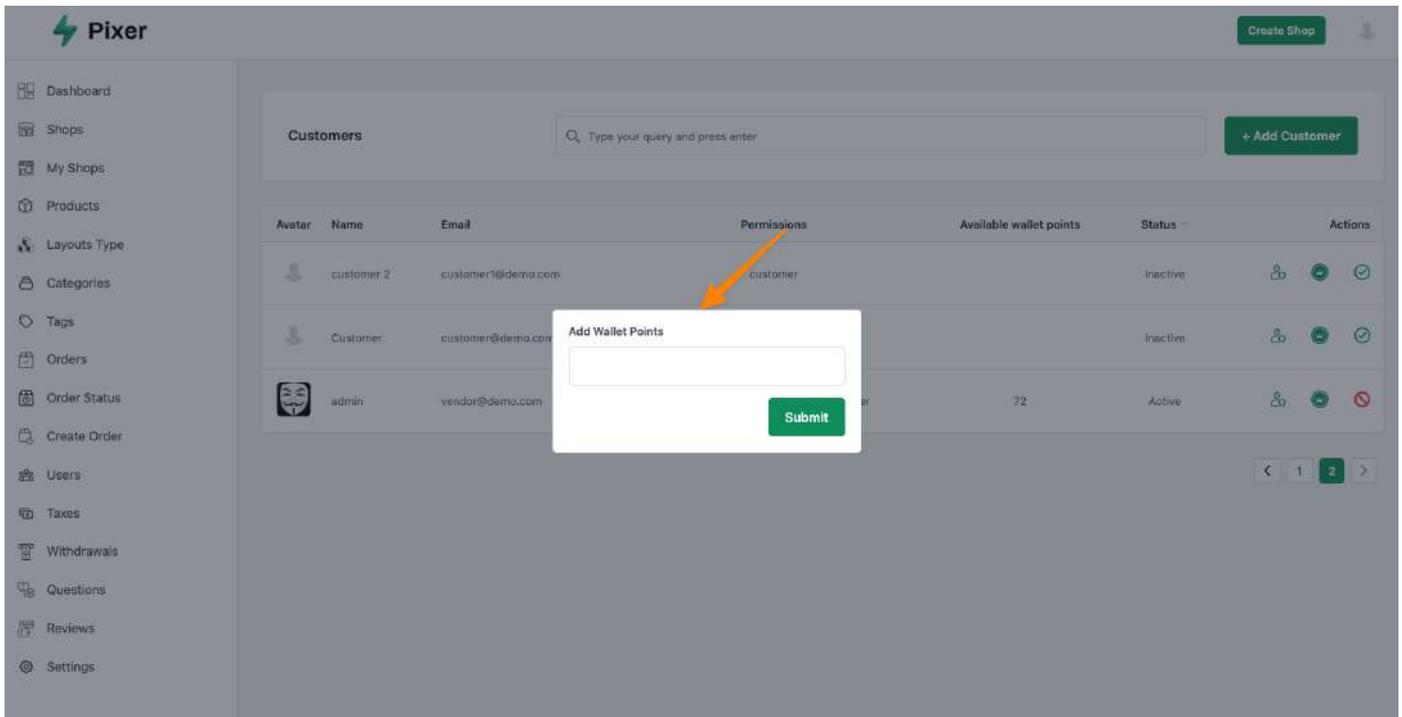
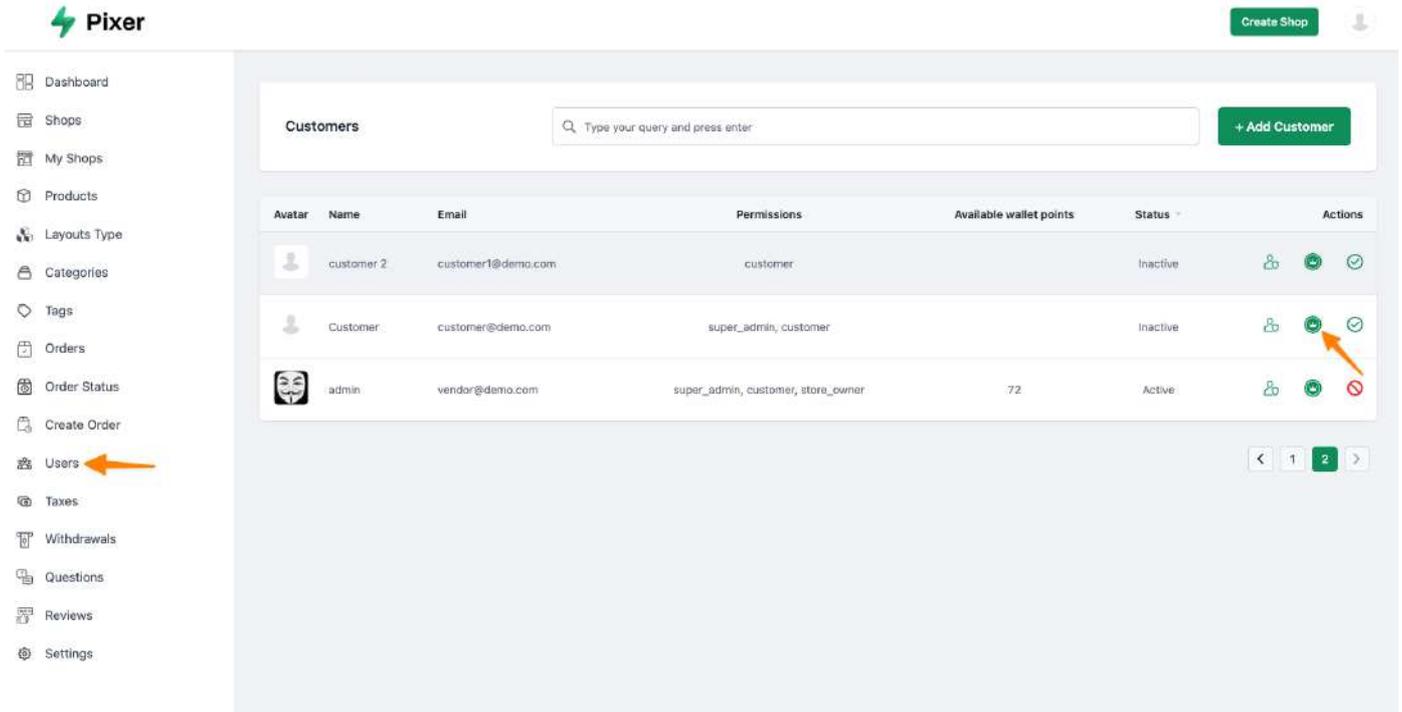
### Sign Up Points:

When new users sign up, they will get signup rewards points for the signup. You can set that signup points from [admin -> settings](#)



### Manually By Admin:

You can give wallet points to a specific user from [admin -> users](#)



Wallet Ratio:

You can set the conversion ratio for currency amount with wallet from `admin -> settings`

The screenshot shows the Pixier admin dashboard. On the left is a sidebar menu with options: Dashboard, Shops, My Shops, Products, Layouts Type, Categories, Tags, Orders, Order Status, Create Order, Users, Taxes, Withdrawals, Questions, Reviews, and Settings. The main content area is divided into two sections. The top section is for 'Wallet Currency Ratio' with a text input field containing '3', an arrow pointing to it, and other settings: 'Sign Up Points' (100), 'Maximum Question Limit' (5), and 'Tax Class' (Select...). The bottom section is for 'SEO' with a sub-header 'Change your site SEO from here' and fields for 'Meta Title', 'Meta Description', and 'Meta Tags'.

For example, you set 3 for the currency ratio. Now, if the customer asks for a refund that is 8\$ price and you approve it, then the customer will get  $8 \times 3 = 24$  wallet points. Similarly, if the customer tries to use this wallet to purchase an 8\$ item, then the wallet deduction will be like this,  $24 / 3 = 8\$$ .

## Payment

### Introduction

We have introduced the new **Payment** architecture feature on Pixier v2.3.0. So if you want to use **Payment** with **Pixier**, then make sure your Pixier is v2.3.0 or later.

We have plan to enrich this feature area by integrating more payment gateways in future. List of available payment gateways now.

- **Stripe**
- **PayPal**
- **RazorPay**
- **Mollie**
- **Paystack**
- **Bitpay**
- **Coinbase**

Let's discuss those sequentially.

### Stripe

Stripe is a financial service & Software as a service (SaaS) company. It offers payment processing software & API for e-commerce applications. In Pixier we have integrated Stripe API for payment system. Though currently **Card** based features are available only, **Stripe Element** for other payment options will be integrate in future updates.

Stripe integrate inside Pixier.

Please follow & complete this steps for stripe integration for your e-commerce system.

- Inside `.env` file copy & paste this line of codes. We will discuss later about how to create those API keys in stripe.

```
STRIPE_API_KEY=[YOUR_STRIPE_API_SECRET_KEY]
STRIPE_WEBHOOK_SECRET_KEY=[YOUR_STRIPE_API_WEBHOOK_SECRET_KEY]
```

- Activate Stripe from Pixier admin dashboard. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

Payment  
Configure Payment Option

Enable Cash On Delivery

Select Payment Gateway

Stripe

Webhook URL

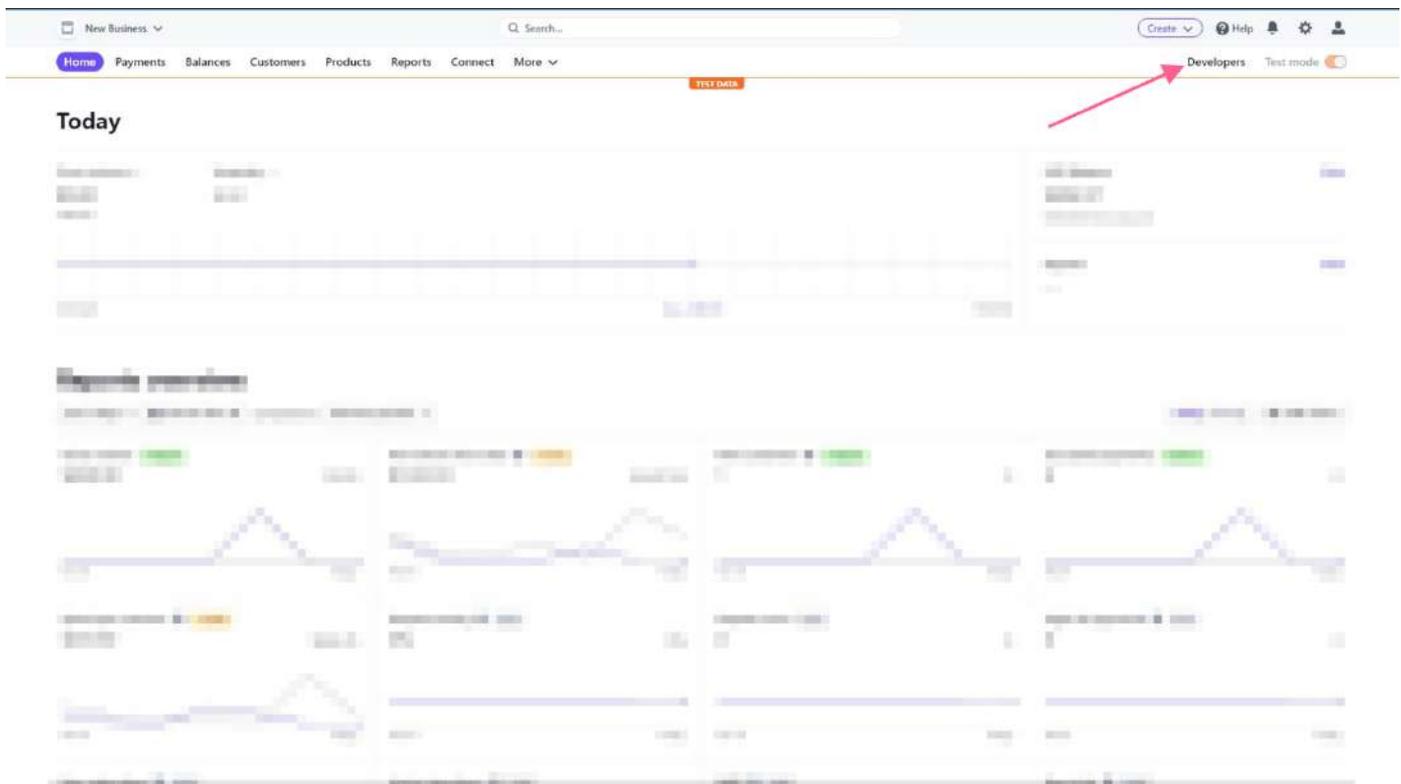
http://127.0.0.1:8000/webhooks/stripe

- Add Stripe publishable key inside Pixier shop.

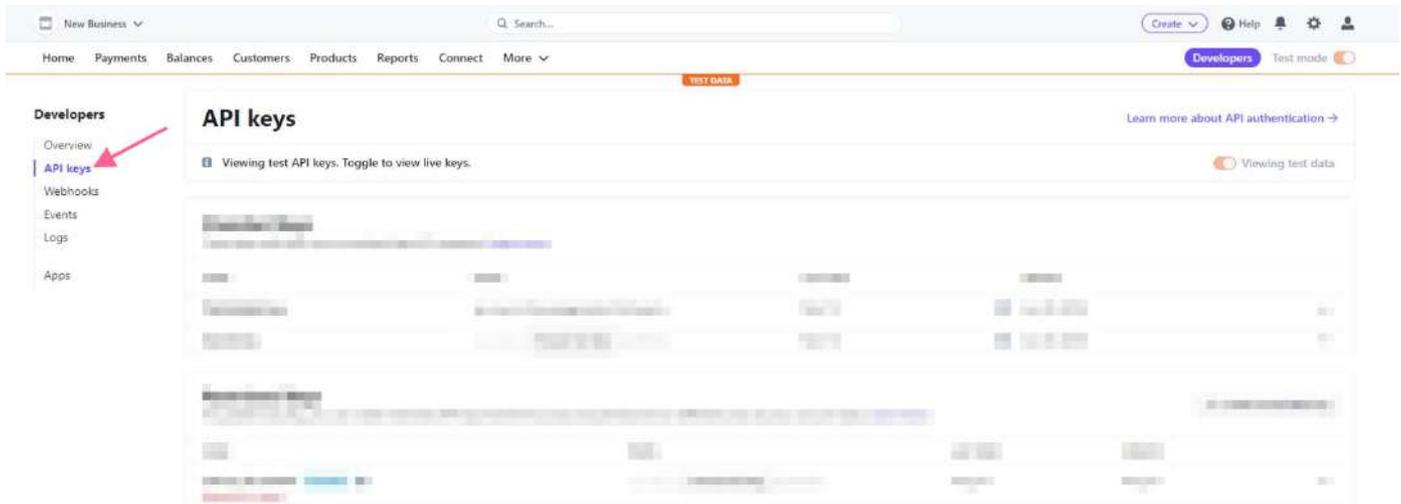
```
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=[YOUR_STRIPE_PUBLISHABLE_KEY]
```

How to create & setup Stripe information properly?

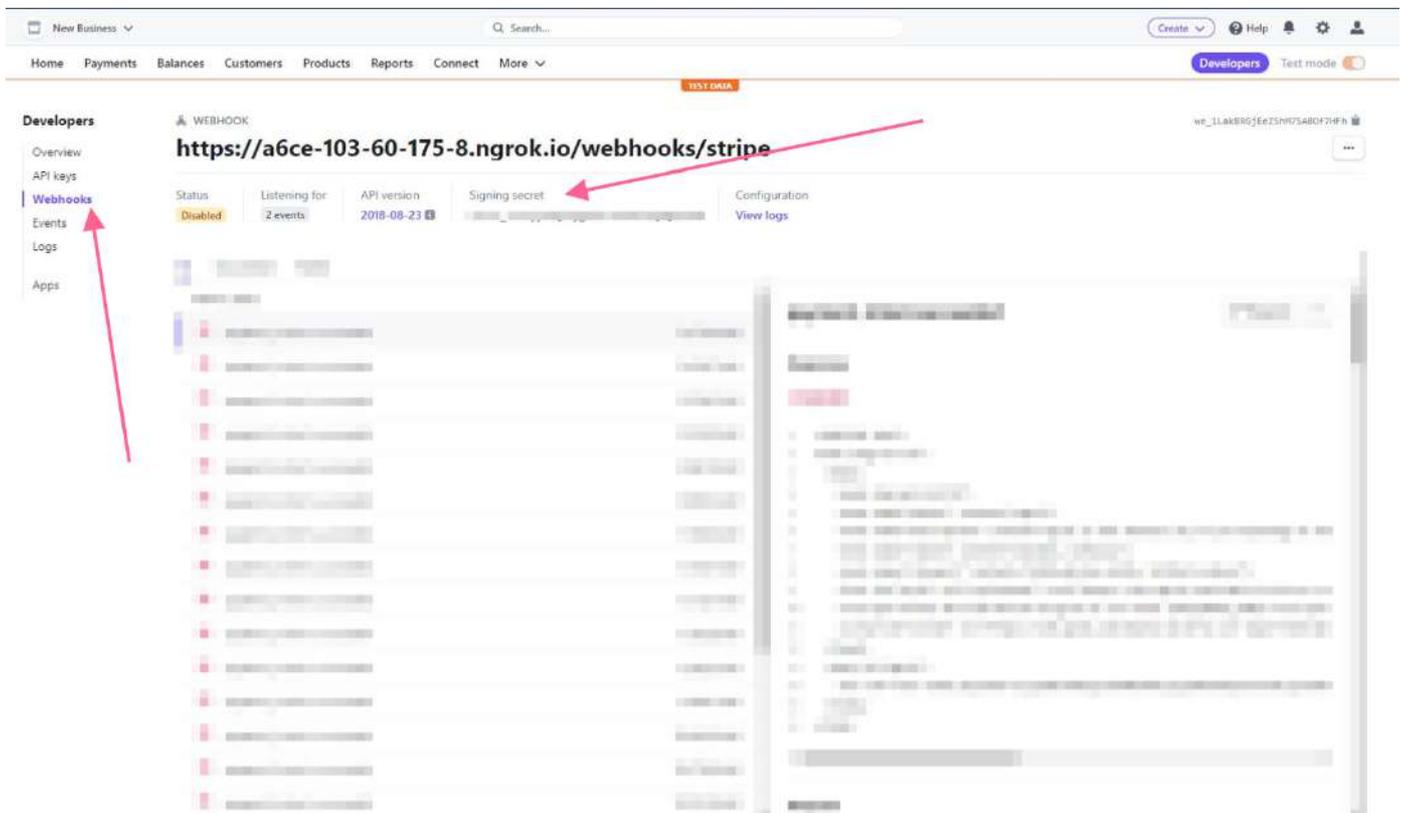
- Go to the stripe website and login <https://stripe.com> If you aren't a registered user, the complete the stripe registration first.
- After logged in into stripe dashboard, follow the developer link to create the API keys.



- Then create API keys from there. for more details follow up this official documentation <https://stripe.com/docs/keys>



- Create Webhook secret key if you decide to up & running webhooks in your App.



- Create this two webhook events for monitoring the payment flow.
  - **payment\_intent.succeeded**
  - **payment\_intent.payment\_failed**

### Special Notes for Stripe users.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

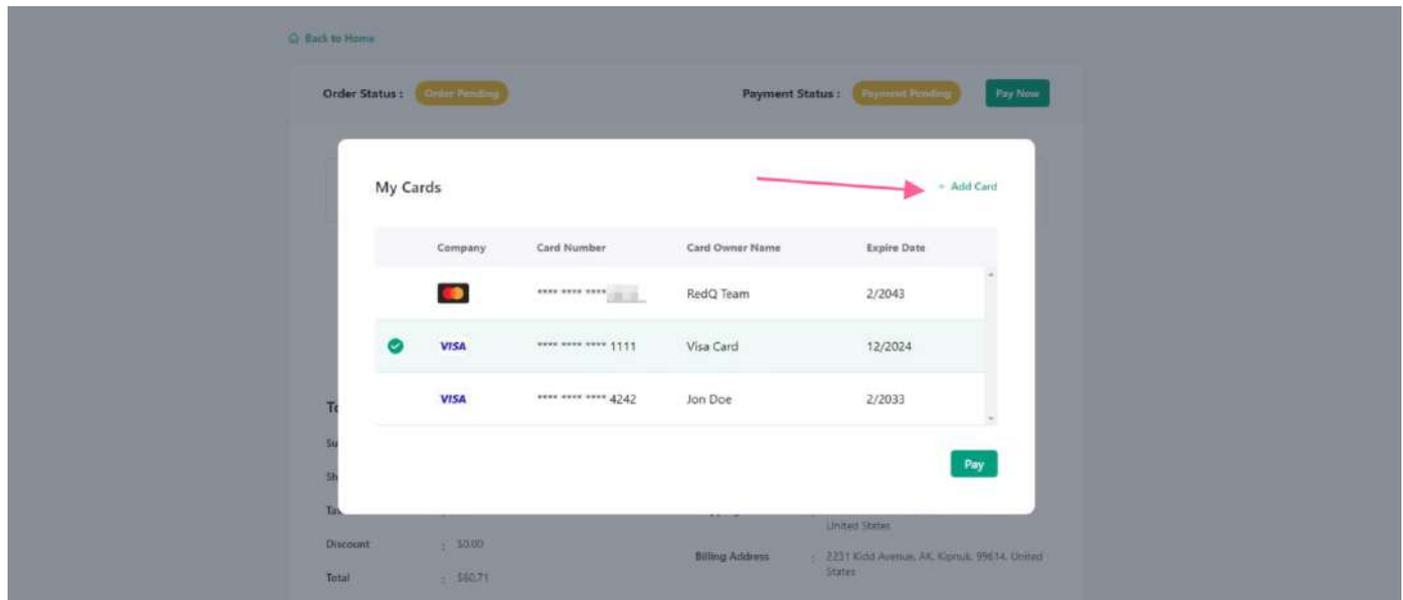
- Stripe DOCS [home page] -- <https://stripe.com/docs>
- Stripe API Docs -- <https://stripe.com/docs/api>
- Stripe JavaScript Docs -- <https://stripe.com/docs/js>

How can I add card in my user profile for future payments in Stripe?

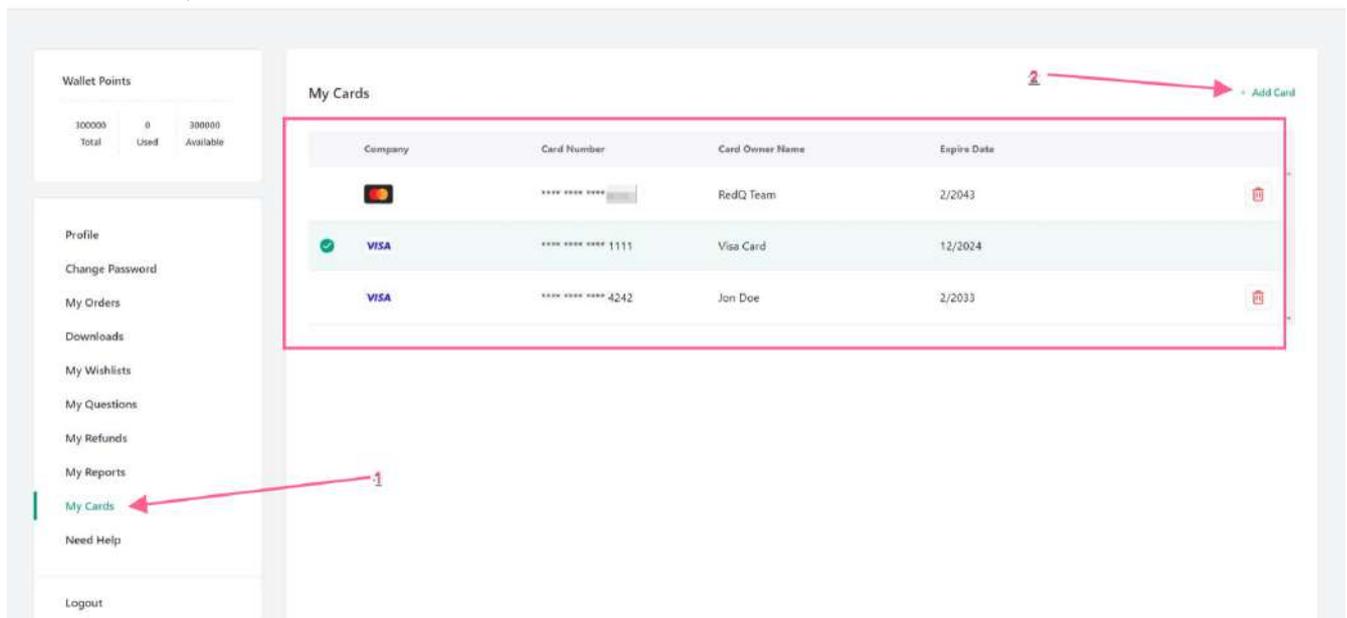
In Pixier we have provided an feature for saving card in case of future usages. This is an **on-session** process. So, if a customer wants to pay via Cards, s/he must have present in the application lively. No **off-session** payment was applied here.

A customer can save a card via two process.

- Save card during checkout process.



- Save card in his user profile **My Card** section.



There are something needs to keep in mind.

#### Please note

- No confidential information was saved in this features. By maintaing the guideline only available information which are permitted to save via Stripe is implemented here.
- Guest user can't save card for future payment.

## PayPal

PayPal is an American multinational financial technology company operating an online payments system. In Pixier we have integrated PayPal APIs which may cover a vast area of PayPal supported region. It will help your business to grow and reach to a wide region.

PayPal integrate inside Pixier.

Please follow & complete this steps for PayPal integration for your e-commerce system.

- Inside `.env` file copy & paste this line of codes. We will discuss later about how to create those API keys in PayPal.

```
# Values: sandbox or live (Default: live)
PAYPAL_MODE=sandbox

# Add currency like USD
PAYPAL_CURRENCY=USD

# Change this accordingly for your application.
PAYPAL_NOTIFY_URL=

# force gateway language i.e. it_IT, es_ES, en_US ... (for express checkout only)
PAYPAL_LOCALE=en

#Validate SSL when creating api client.
PAYPAL_VALIDATE_SSL=

# PayPal Setting & API Credentials -> sandbox

PAYPAL_SANDBOX_CLIENT_ID=[YOUR_PAYPAL_SANDBOX_CLIENT_ID]
PAYPAL_SANDBOX_CLIENT_SECRET=[YOUR_PAYPAL_SANDBOX_CLIENT_SECRET_KEY]

# PayPal Setting & API Credentials -> live

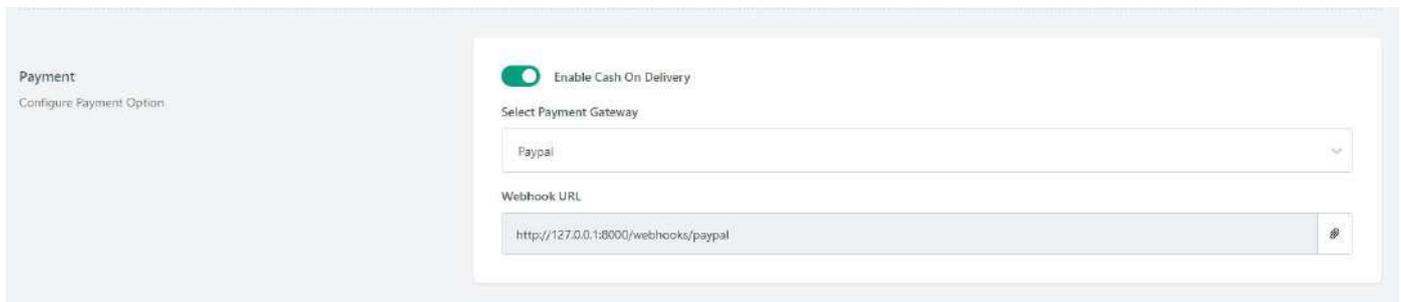
PAYPAL_LIVE_CLIENT_ID=[YOUR_PAYPAL_LIVE_CLIENT_ID]
PAYPAL_LIVE_CLIENT_SECRET=[YOUR_PAYPAL_LIVE_CLIENT_SECRET_KEY]

# PayPal Webhook settings

PAYPAL_WEBHOOK_ID=[YOUR_PAYPAL_WEBHOOK_URL]
SHOP_URL=[YOUR_SHOP_URL]
```

**SHOP\_URL=[YOUR\_SHOP\_URL]** This parameter is must have in .env file when PayPal is using. Otherwise the payment redirection will be broken.

- Activate PayPal from Pixier admin dashboard. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)



Payment  
Configure Payment Option

Enable Cash On Delivery

Select Payment Gateway

Paypal

Webhook URL

http://127.0.0.1:8000/webhooks/paypal

How to create & setup PayPal information properly?

- Create an account in <https://developer.paypal.com>
- Choose sandbox if you are testing your development environment.
- Click on **Default Applicaiton**. or create a new App.
- You will get your Sandbox API credentials. Copy & paste those inside Pixier app .env file like mentioned above.

PayPal Developer

Search

Docs APIs & SDKs Tools Support

DASHBOARD

**My Apps & Credentials**

My Account

SANDBOX

Accounts

Notifications

API Calls

IPN Simulator

Webhooks Events

MOCK

Webhooks Simulator

Credit Card Generator

Negative Testing

LIVE

API Calls

Webhooks Events

### Default Application

App display name: Default Application

#### SANDBOX API CREDENTIALS

Sandbox Account

Client ID

Secret [show](#)

#### SANDBOX APP SETTINGS

Return URL - Users are redirected to this URL after live transactions. Allow up to three hours for the change to take effect. [Show](#)

App feature options

Accept payments. Enable one-time and subscription payments. [Advanced options](#)

- You will find the Webhook ID and link too.

PayPal Developer

Search

Docs APIs & SDKs Tools Support

DASHBOARD

**My Apps & Credentials**

My Account

SANDBOX

Accounts

Notifications

API Calls

IPN Simulator

Webhooks Events

MOCK

Webhooks Simulator

Credit Card Generator

Negative Testing

LIVE

API Calls

Webhooks Events

Save Cancel

### SANDBOX WEBHOOKS

Configure webhooks to notify your app when certain events occur. To configure a webhook, define your webhook listener URL and a list of events for which to listen. You can configure up to ten webhooks. Each webhook can subscribe to either specific events or all events. To learn more about webhooks, see [webhooks notifications](#).

Search by Webhook id, event name

Webhook ↑	Webhook id	Event Tracked
		Payment capture completed, Payment capture denied, Payment capture pending, Payment capture refunded, Payment capture reversed

[Add Webhook](#)

Developer Support Tools Family

- Then Add the webhook events if you want to up & running the services.
  - PAYMENT.CAPTURE.COMPLETED**
  - PAYMENT.CAPTURE.PENDING**
  - PAYMENT.CAPTURE.CANCELLED**
  - PAYMENT.CAPTURE.REVERSED**
- At last, For going live with your application please follow this official documentation. <https://developer.paypal.com/docs/archive/paypal-here/sdk-dev/going-live/>

## RazorPay

Razorpay is the only payments solution in India that allows businesses to accept, process and disburse payments with its product suite. It gives you access to all payment modes including credit card, debit card, netbanking, UPI and popular wallets including JioMoney, Mobikwik, Airtel Money, FreeCharge, Ola Money and PayZapp.

## RazorPay integration inside Pixier.

Please follow & complete this steps for RazorPay integration with your e-commerce system.

- Inside `.env` file copy & paste this line of codes. We will discuss later about how to create those API keys in RazorPay.

```
RAZORPAY_KEY_ID=[YOUR_RAZORPAY_KEY_ID]
RAZORPAY_KEY_SECRET=[YOUR_RAZORPAY_KEY_SECRET]
RAZORPAY_WEBHOOK_SECRET_KEY=[YOUR_RAZORPAY__WEBHOOK_URL]
```

- Activate RazorPay from Pixier admin dashboard. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

**Payment**  
Configure Payment Option

Enable Cash On Delivery

Select Payment Gateway

RazorPay

Webhook URL

http://localhost:8000/webhooks/razorpay

- Add RazorPay publishable key inside pixier shop.

```
NEXT_PUBLIC_RAZORPAY_KEY=[YOUR_RAZORPAY_KEY_ID]
```

## How to create &amp; setup RazorPay information properly?

- Go to the RazorPay website & login <https://razorpay.com> If you aren't a registered user, Complete the RazorPay registration first.
- After logged in into RazorPay dashboard, Click on Settings

**Razorpay**

YOU'RE IN TEST MODE

Test Mode

Payment Button

Route

Subscriptions

QR Codes **NEW**

Smart Collect

Customers

Offers

Checkout Rewards

Reports

My Account **NEW**

**Settings**

App Store **NEW**

Theme Color

#528FFD Save

Choose a theme color for your brand. The default theme color will be used if none is specified.

Your Logo

Choose File

Max file size: 1MB

Choose a square image of minimum dimensions 256x256 px.

Default Language

English Save

Default language will be used on the Checkout page if customer doesn't specify a language.

Changes will reflect on Checkout page, Payment Links, Invoices & Payment pages.

Mahmud  
Order ID  
₹ 1

Phone

Email

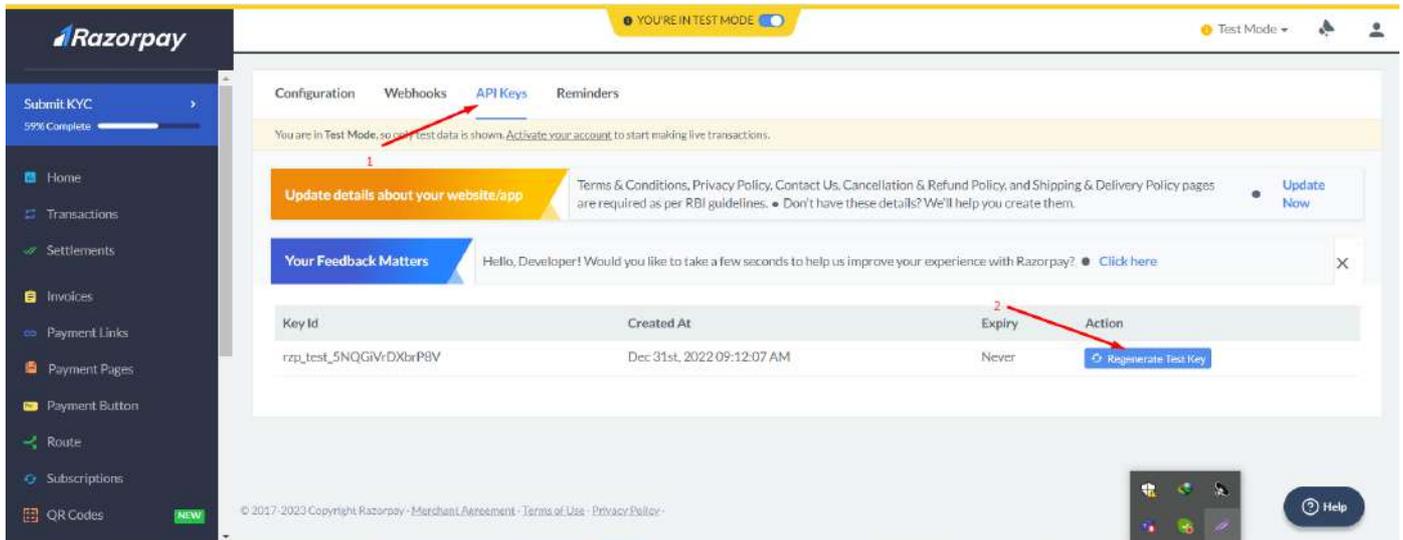
SELECT A PAYMENT METHOD

Card Netbanking Wallet

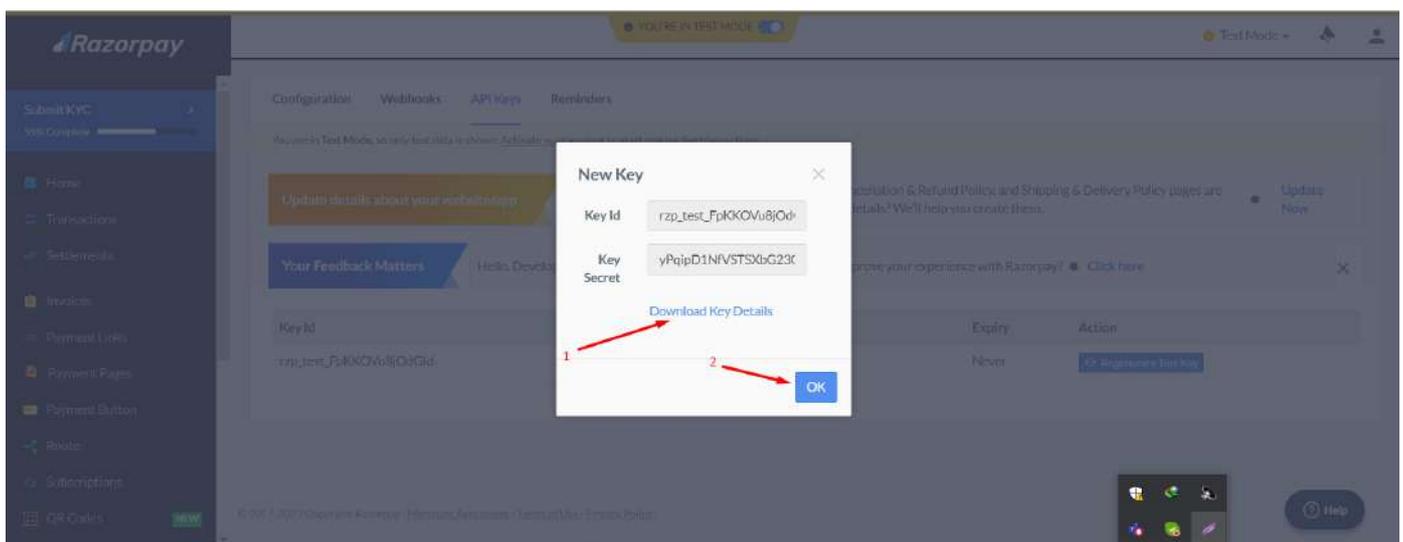
UPI Emi QR

Help

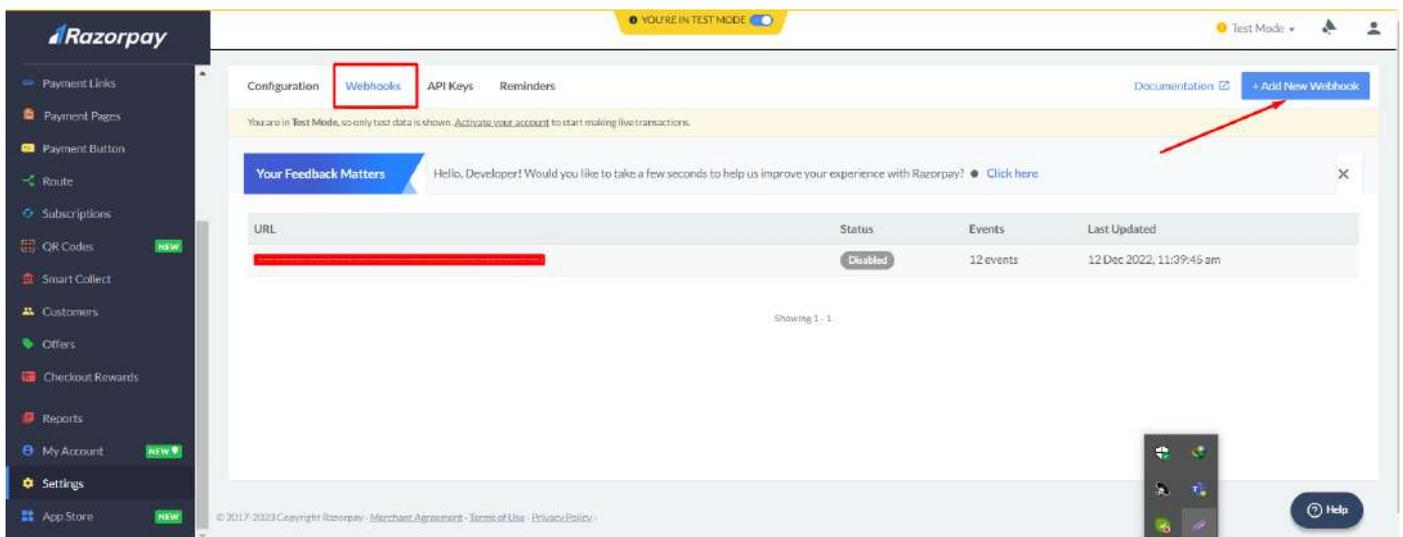
- Select API Keys option & click on Generate Test Key.



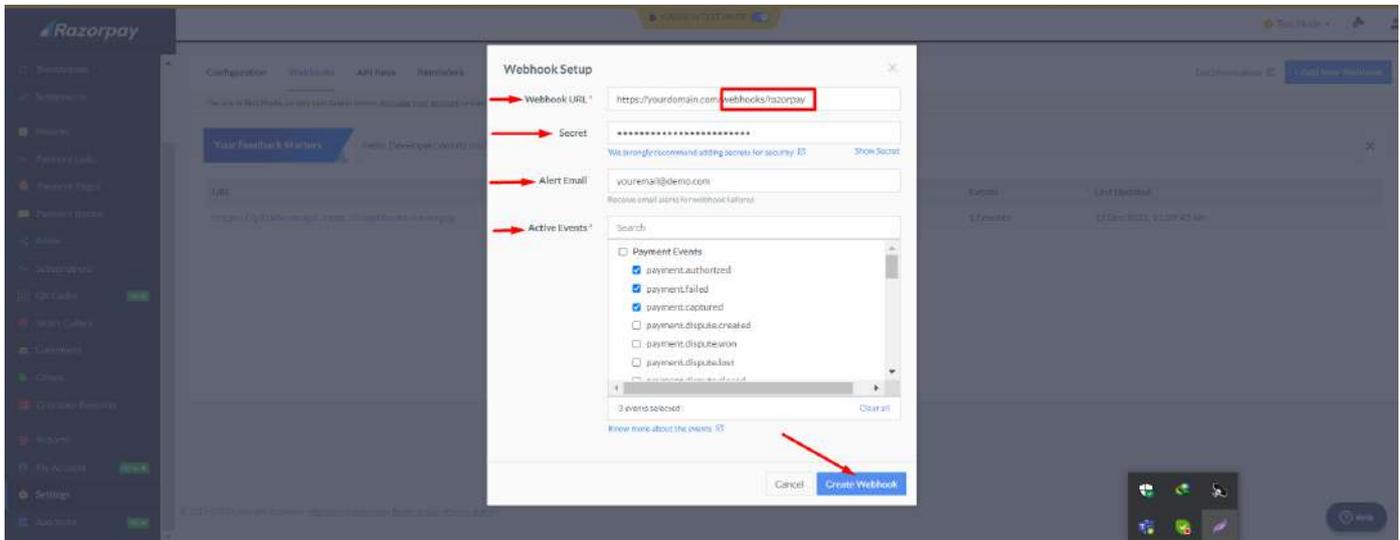
- The Client Id and Secret Key will appear. Download these keys and click on OK



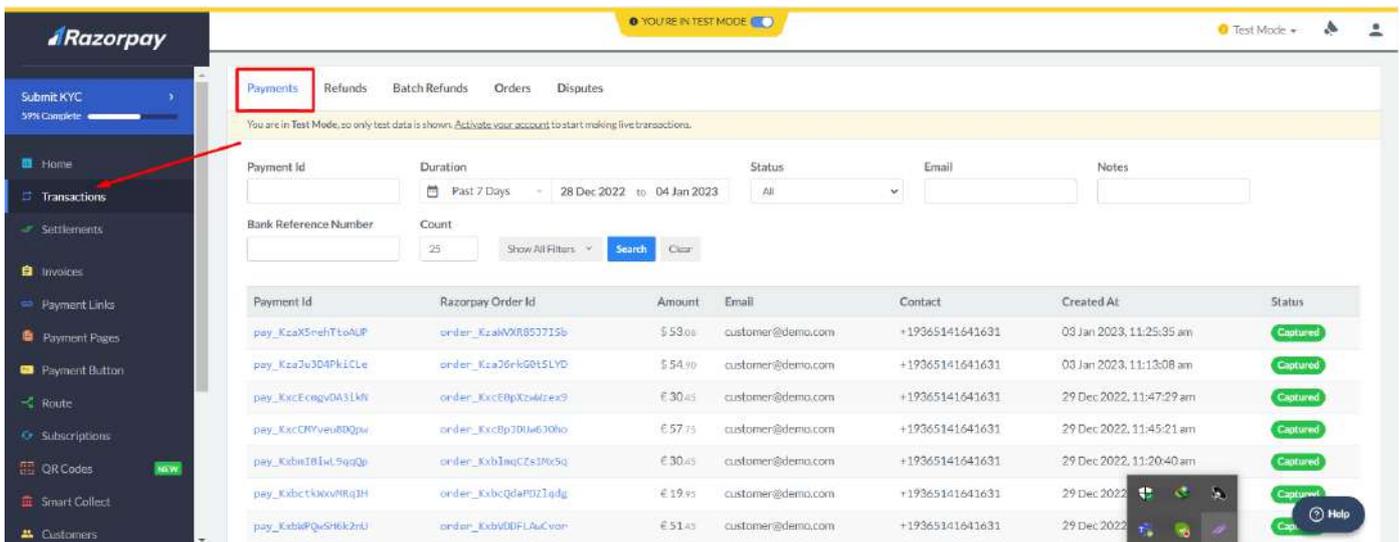
- Select Webhooks option & click on Add New Webhooks.



- Enter Webhook URL , Secret key , Alert Email and tick mark the checkbox in Active Event then click on create Webhook



- Copy & paste those inside Pixier app .env file like mentioned above. after that test your RazorPay Payment Status



### Special Notes for RazorPay.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

- RazorPay DOCS -- <https://razorpay.com/docs/#home-payments>
- RazorPay Webhooks -- <https://razorpay.com/docs/webhooks/setup-edit-payments>

### Mollie

Mollie is a Payment Service Provider (PSP) that processes online payments for companies. If you buy something online from one of our merchants, we make sure that your money is transferred safely from your bank to the merchant's bank. Since we arrange the payment process, you may see Mollie or Stg Mollie Payments on your bank statement.

Mollie integrate inside Pixier.

Please follow & complete this steps for Mollie integration for your e-commerce system.

- Inside .env file copy & paste this line of codes. We will discuss later about how to create those API keys in Mollie.

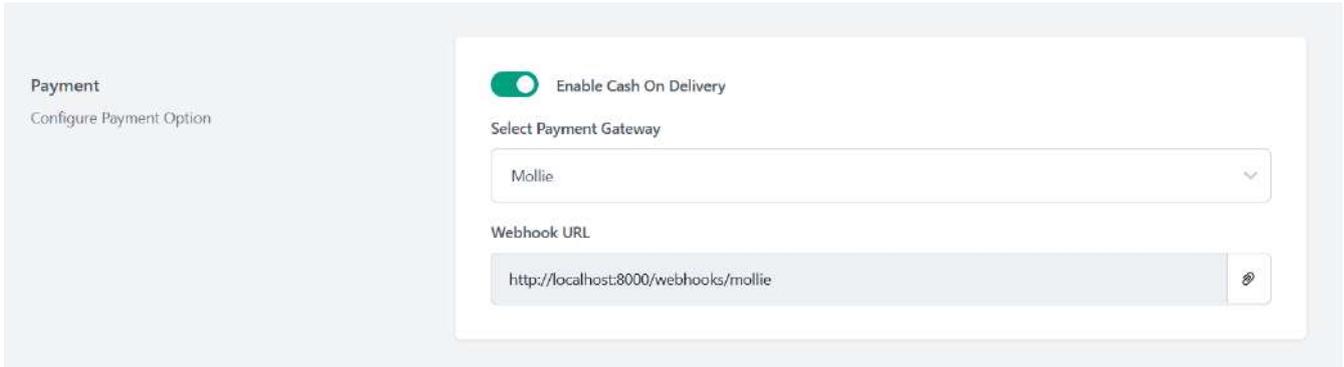
```
MOLLIE_KEY=[YOUR_MOLLIE_API_KEY]
```

### Mollie Webhook settings

```
SHOP_URL=[YOUR_SHOP_URL]
MOLLIE_WEBHOOK_URL=[YOUR_MOLLIE_WEBHOOK_URL]
```

**SHOP\_URL=[YOUR\_SHOP\_URL]** This parameter is must have in .env file when Mollie is using. Otherwise the payment redirection will be broken.

- Activate Mollie from Pixier admin dashboard. (e.g. Mollie Webhook URL is coming from local development. This static link will dynamically generated in live environment)



- Copy & paste those inside Pixier app .env file like mentioned above.
- If you want to use webhook during development on localhost, you must use a tool like ngrok to have the webhooks delivered to your local machine.

```

ngrok
- ngrok http http://localhost:8000
ngrok (Ctrl+C to quit)
Check which logged users are accessing your tunnels in real time https://ngrok.com/s/app-users
Session Status      online
Account             ████████████████████
Version             3.1.0
Region              India (in)
Latency             116ms
Web Interface       http://127.0.0.1:4040
Forwarding           https://c852-37-111-243-195.in.ngrok.io -> http://localhost:8000
Connections
  ttl   opn   rt1   rt5   p50   p90
   0    0    0.00 0.00 0.00 0.00

```

- Copy the forwarding https link & paste .env file after link add line- /webhooks/mollie

```

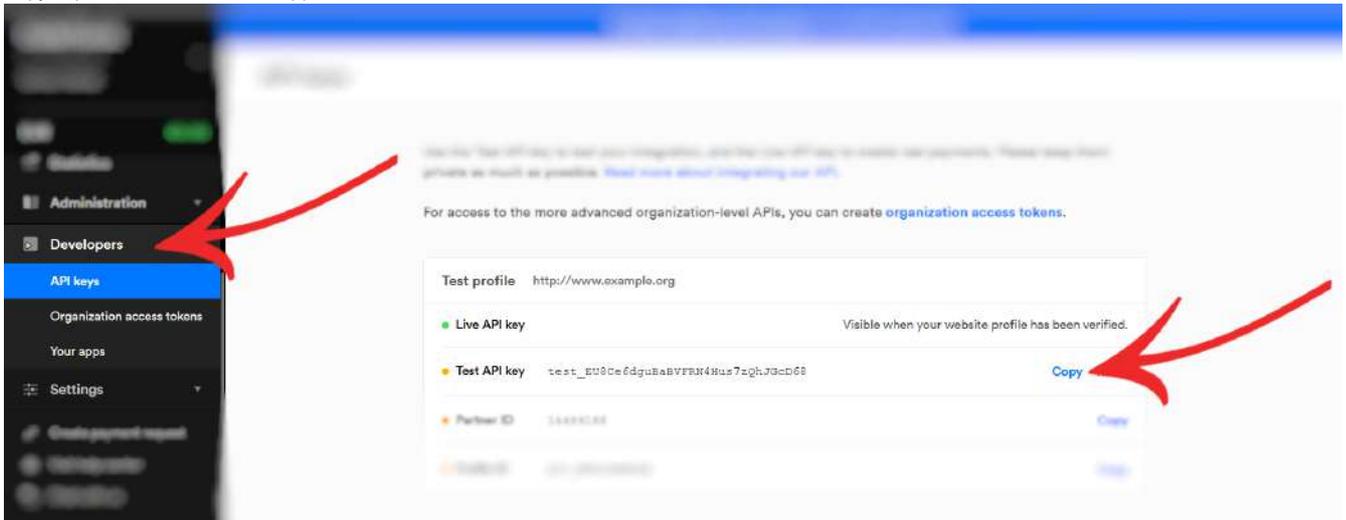
84 # Payment -> Mollie
85 MOLLIE_KEY=test_EU8Ce6dguBaBVFRN4Hus7zQhJGcD68
86 MOLLIE_WEBHOOK_URL=https://c852-37-111-243-195.in.ngrok.io/webhooks/mollie

```

How to create & setup Mollie information properly?

- Go to the Mollie website & login <https://www.mollie.com> If you aren't a registered user, the complete the Mollie registration first.
- After logged in into Mollie dashboard, follow the developer link to copy the API keys.
- Choose Test API key your are testing your development environment.

- Copy & paste those inside Pixier app .env file like mentioned above.



### Special Notes for Mollie.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

- Mollie DOCS -- <https://docs.mollie.com/>
- Mollie Webhooks -- <https://docs.mollie.com/overview/webhooks>

### Paystack

Paystack is a Nigerian fintech company that provides online payment solutions for businesses. It offers a range of services, including online and offline payments, subscriptions, and invoicing. Paystack's platform enables businesses to easily and securely accept payments from customers via credit card, debit card, and mobile money.

Paystack integrate inside Pixier.

Please follow & complete this steps for Paystack integration for your e-commerce system.

- First go to settings from Pixier admin dashboard. Inside settings you will find Currency option. Select your Currency. If you want to Test development on localhost. then Select South African Rand(ZAR).

The screenshot shows the Pixier dashboard. On the left is a sidebar with navigation items: Layouts Type, Categories, Tags, Orders, Create Order, Users, Taxes, Withdrawals, Questions, Reviews, and Settings (highlighted with a red box and arrow). The main area displays four summary cards: Total Revenue (€76.99), Total Order (10), Today's Revenue (€0.00), and Total Shops (60). Below these is a 'Sale History' bar chart with a single bar at approximately 75.00. The bottom section is the 'Information' settings page, which includes fields for Site Title (Pixier), Site Subtitle, Currency (South African Rand), Minimum Order Amount, and Wallet Currency Ratio (100). A red arrow points to the 'Currency' dropdown menu.

Here is a breakdown of countries and the currencies they can accept payments in.

Country	Available Currencies
Ghana	GHS
Nigeria	NGN, USD
South Africa	ZAR

For more information: [Paystack currency is available.](#)

- Inside api there is `.env` file. You have to copy & paste this line of codes inside the `.env` file. We will discuss later about how to create those API keys in [Paystack official website](#).

```
PAYSTACK_PUBLIC_KEY=[YOUR_PAYSTACK_PUBLIC_KEY]
PAYSTACK_SECRET_KEY=[YOUR_PAYSTACK_SECRET_KEY]
PAYSTACK_PAYMENT_URL=https://api.paystack.co
MERCHANT_EMAIL=[YOUR_PAYSTACK_MERCHANT_EMAIL]
```

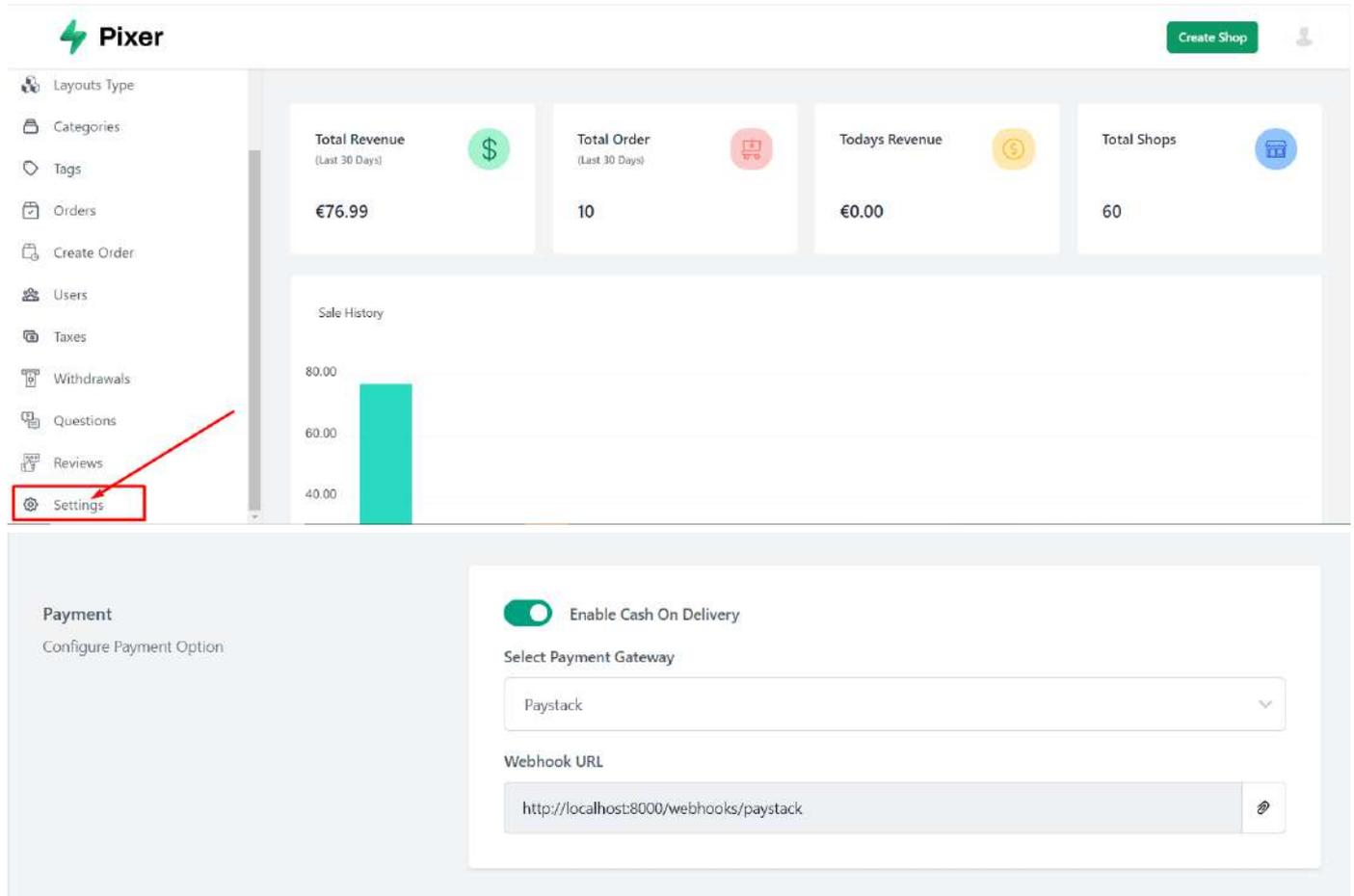
- Add Paystack publishable key inside Pixier shop.

```
NEXT_PAYSTACK_PUBLIC_KEY=[YOUR_YOUR_PAYSTACK_PUBLIC_KEY]
```

**SHOP\_URL=[YOUR\_SHOP\_URL]** This parameter is must have in .env file when Paystack is using. Otherwise the payment redirection will be broken.

### Paystack Webhook settings

- To activate Paystack go to settings from Pixier admin dashboard. Inside settings you will find configure payment option. (e.g. Paystack Webhook URL is coming from local development. This static link will dynamically generated in live environment)



- Copy the webhook url & registered in Paystack Dashboard.



- If you want to use webhook during development on localhost, you must use a tools like ngrok to have the webhooks delivered to your local machine.

```

ngrok
- ngrok http http://localhost:8000
Check which logged users are accessing your tunnels in real time https://ngrok.com/s/app-users
Session Status      online
Account              ████████████████████
Version              3.1.0
Region              India (in)
Latency              116ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://c852-37-111-243-195.in.ngrok.io -> http://localhost:8000

Connections
  ttl   opn   rt1   rt5   p50   p90
    0     0    0.00  0.00  0.00  0.00

```

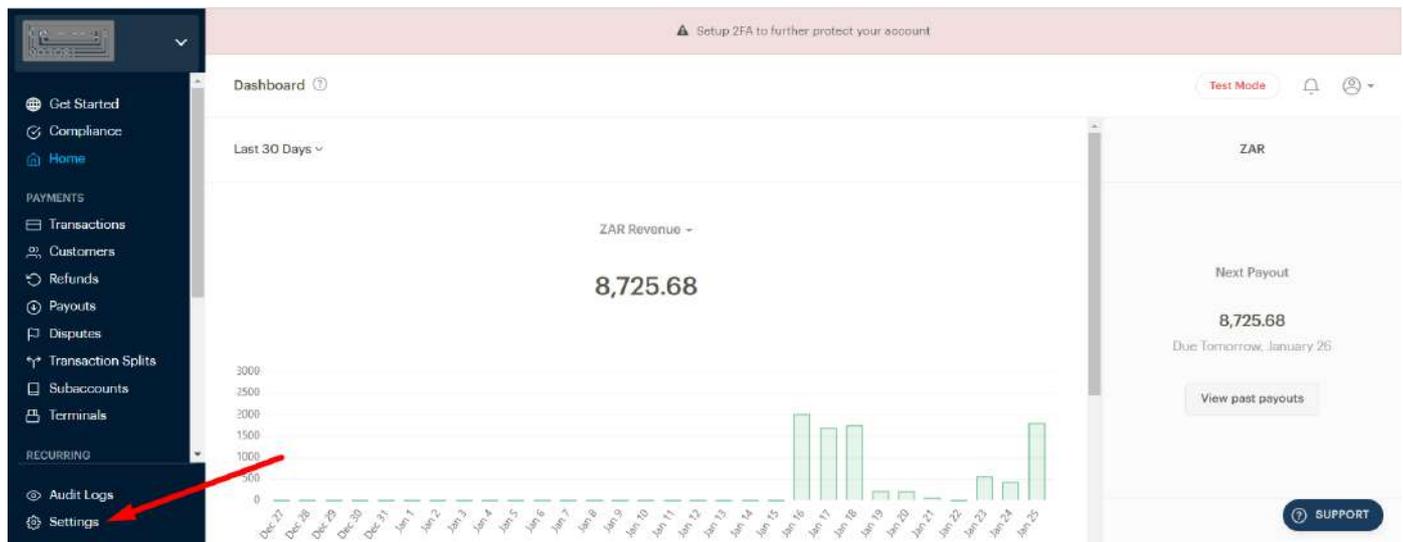
- Copy the forwarding https link & paste on Test Webhook URL, after paste forwarding https link, add line- /webhooks/paystack.

Test Callback URL

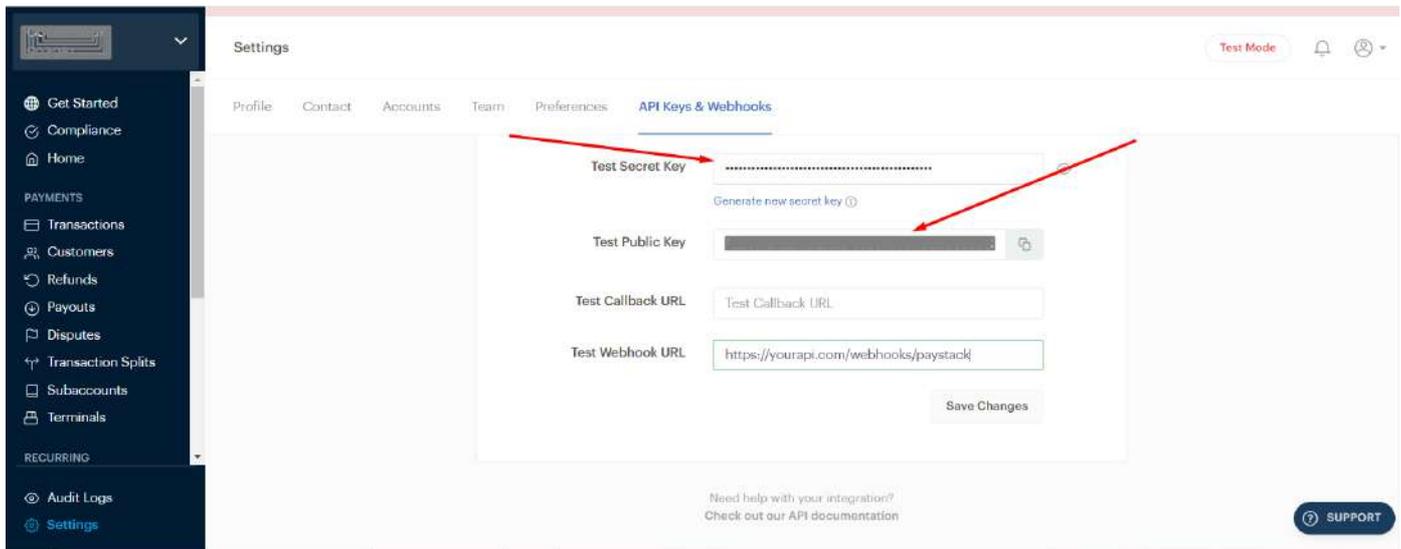
Test Webhook URL

### How to create & setup Paystack information properly?

- Go to the [Paystack official website](#) & login If you aren't a registered user, Complete the Paystack registration first.
- After logged in into Paystack dashboard, Click on Settings.



- After That Click on API Keys & Webhooks for Paystack Secret Key & Paystack Public Key.



- Copy & paste those inside Pixier API and Pixier Shop `.env` file like mentioned above.

### Special Notes for Paystack.

If we have used any third party system/plugin/packages, then we have always encouraged our respected customers to follow the official documentation for detailed & in-depth knowledge.

- [Paystack official documentation](#)
- [Paystack Webhooks documentation](#)
- [Accept Payments in US Dollars \(USD\)](#)

### BitPay

BitPay is a bitcoin payment service provider headquartered in Atlanta, Georgia, United States. It was founded in May 2011 by Tony Gallippi and Stephen Pair. BitPay provides Bitcoin and Bitcoin Cash payment processing services for merchants.

Before integrate BitPay check the list that your country is not restricted. [Restricted Country List](#)

BitPay integrate inside Pixier.

Please follow & complete this steps for BitPay integration for your e-commerce system.

- Inside `.env` file copy & paste this line of codes. We will discuss later about how to create those API keys in BitPay.

```
# `BITPAY_IS_PRODUCTION` value either true or false
BITPAY_IS_PRODUCTION=false

# Set any password you want that will use for encrypting your private key
BITPAY_KEY_STORAGE_PASSWORD=[LONG_SECRET]

# This is used for generating your token.
BITPAY_ENABLE_MERCHANT=true
BITPAY_ENABLE_PAYOUT=false

# This is important generating your json or yaml file
BITPAY_GENERATE_JSON_FILE=true
BITPAY_GENERATE_YML_FILE=false

# All Files are located `storage/app/private/`. Which will use for initiating you BITPAY client.

SHOP_URL=[YOUR_SHOP_URL]
```

**SHOP\_URL=[YOUR\_SHOP\_URL]** This parameter is must have in `.env` file when BitPay is using. Otherwise the payment redirection will be broken.

- Activate BitPay from Pixier admin dashboard. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

**Payment**

Configure Payment Option

Select Payment Gateway

BitPay

Webhook URL

http://localhost:8000/webhooks/bitpay

### How to create & setup BitPay information properly?

- Create an account in <https://bitpay.com/authenticate/signup?business>
- Choose sandbox if your are testing your development environment signup in here <https://test.bitpay.com/authenticate/signup>.
- Now go to the terminal run `php artisan marvel:generate_bitpay_config` to generate your private key and config file
- You will find an output like below, With some instructions.

```

→ php artisan marvel:generate_bitpay_config
Generating configuration file for BitPay.....

+-----+
| key   | Value |
+-----+
| Label | Marvel_1684812919 |
| Facade | merchant |
| Token | ERY2fXDUq3fUytCxxr4ZTSp73M2U98J99PNfzB7scZJF |
| PairingCode | xhxXHto |
+-----+

[INFO] Please, Go to the following link to approve your BitPay API Token.
[INFO] https://test.bitpay.com/api-access-request?pairingCode=xhxXHto.
[INFO] Once you have this Pairing Code approved you can start using the Client.
[INFO] BitPay config file has been generated Successfully to: storage/app/private.

```

- You can approve the API Token by logon to that url, or just copy The ParingCode and goto your bitpay merchant account's dashboard. Go to Payment Tools find API TOKEN section paste Paring Code there. Then find and approve it.

The screenshot shows the BitPay merchant dashboard. On the left is a navigation sidebar with 'Payment Tools' selected. The main content area is titled 'Payment Tools' and contains several cards. In the 'Integration Resources' section, a red arrow points to the 'API Tokens' card, which has the description 'Add, revoke, or modify BitPay API access tokens.' Other cards include 'Quick Checkout for Web' and 'Legacy API Tokens (Deprecated)'.

test.bitpay.com/api-access-request?pairingCode=xhxXHto

Dashboard > my-account > API Token Request

## API Token Request

A client is requesting access to use the **BitPay API** for your BitPay merchant. By confirming this request you are creating a Token with the listed Facade and Client ID.

**Capabilities**

A facade defines a set of limited capabilities. The "Point-of-Sale" facade has the capability to create and get invoices, while the "Merchant" facade has the capability to do most of the things you see in the Dashboard. A token provides access to the capabilities of a facade on a specific resource (a merchant organization in this case).

Client ID:

Tf57a8Fh1BQQiyoesBr9HPiAz5yhBnCF8q4

Facade: Merchant

Information: Created: Now

Webhook is most important thing to manage payments status with bitpay setup.

- Available Webhooks Status.
  - new
  - paid
  - confirmed
  - complete
  - expired
  - invalid
- At last, For going live with your application please follow this official documentation. <https://github.com/bitpay/php-bitpay-client-v2> and <https://test.bitpay.com/api/>

## Coinbase

Coinbase Global, Inc., branded Coinbase, is an American publicly traded company that operates a cryptocurrency exchange platform. Coinbase is a distributed company; all employees operate via remote work. It is the largest cryptocurrency exchange in the United States by trading volume.

In Pixier Laravel we integrate Coinbase Commerce API. Coinbase Commerce does not provide test or sandbox environment, So be careful. It will cost you money even in testing.

Coinbase integrate inside Pixier.

Please follow & complete these steps for Coinbase integration for your e-commerce system.

- Inside `.env` file copy & paste this line of codes. We will discuss later about how to create those API keys in Coinbase.

```
# Payment -> Coinbase
COINBASE_API_KEY=

SHOP_URL=[YOUR_SHOP_URL]
```

**SHOP\_URL=[YOUR\_SHOP\_URL]** This parameter is must have in `.env` file when Coinbase is using. Otherwise the payment redirection will be broken.

- In Pixier **Laravel** **Http** Client has been used to implement Coinbase Commerce.
- Activate Coinbase from Pixier admin dashboard. (e.g. webhook URL is coming from local development. This static link will dynamically generated in live environment)

Payment

Configure Payment Option

Select Payment Gateway

Coinbase

Webhook URL

http://localhost:8000/webhooks/coinbase

How to create & setup Coinbase information properly?

- Create an account in <https://beta.commerce.coinbase.com/signup>
- Get API key from your dashboard. Copy the API\_KEY then set it to `COINBASE_API_KEY` env variable.

COMMERCE Settings

Your Business Payments Notifications Security

API keys

72db0253-----9049

Whitelisted domains

If you prefer to control where your payment buttons are allowed to be embedded, add your domains and subdomains here.

- Coinbase Commerce do not provide any kind of test or sandbox environment.

Webhook is most import thing to manage payments status with coinbase setup.

- To Set Webhook notification URL go to coinbase commerce dashboard -> settings -> notification set url

COMMERCE

New Webhook Subscription

https://yourDomain.com/webhooks/coinbase

Save

Webhook subscriptions

Add a URL to start receiving payment notifications.

Show shared secret Add an endpoint

- Available Webhooks Status.
  - **charge:created** New charge is created
  - **charge:confirmed** Charge has been confirmed and the associated payment is completed
  - **charge:failed** Charge failed to complete
  - **charge:delayed** Charge received a payment after it had been expired

- **charge:pending** Charge has been detected but has not been confirmed yet
- **charge:resolved** Charge has been resolved
- At last, For going live with your application please follow this official API Guide <https://docs.cloud.coinbase.com/commerce/reference> and documentation <https://docs.cloud.coinbase.com/commerce/docs/welcome>

## Integration of new payment gateway

To integrate a payment gateway in Pixier-Laravel, you will need to follow these general steps:

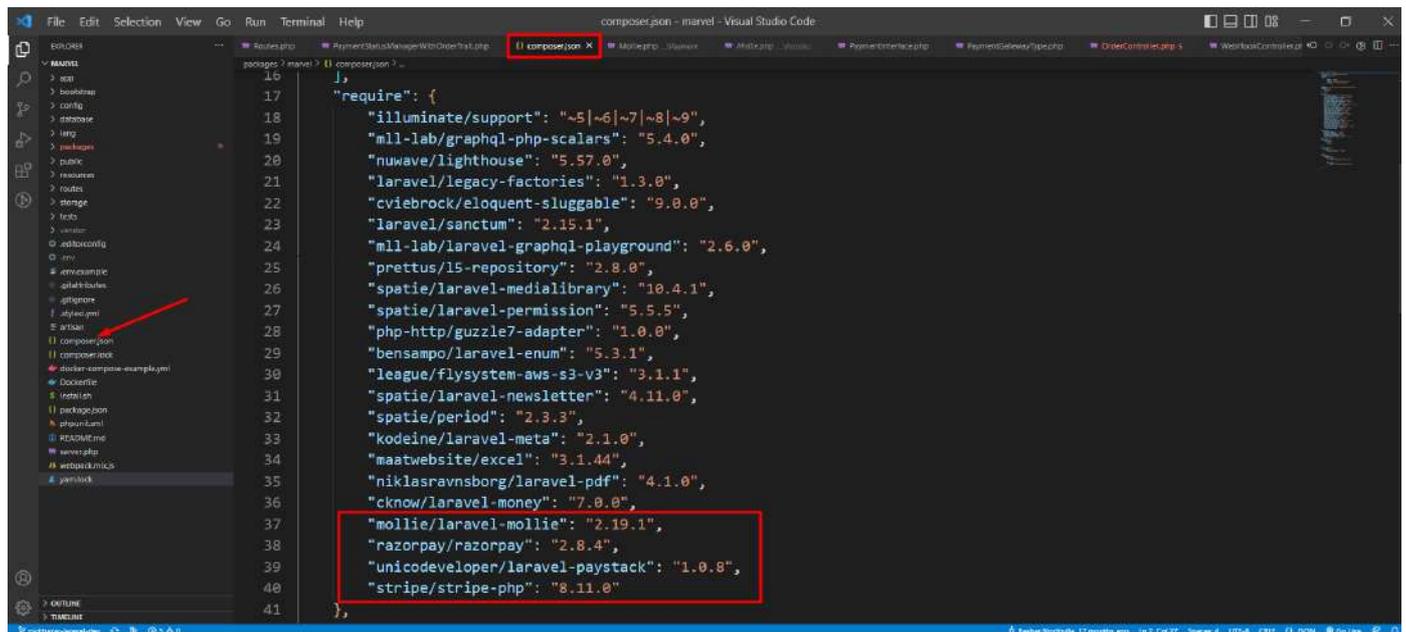
### Getting Started with API

#### Step 1: Install and configure the payment gateway package

First, you will need to install the payment gateway package for Laravel (if there are any available package). There are several packages available that provide integration with various payment gateways. Example is given in the screenshot.

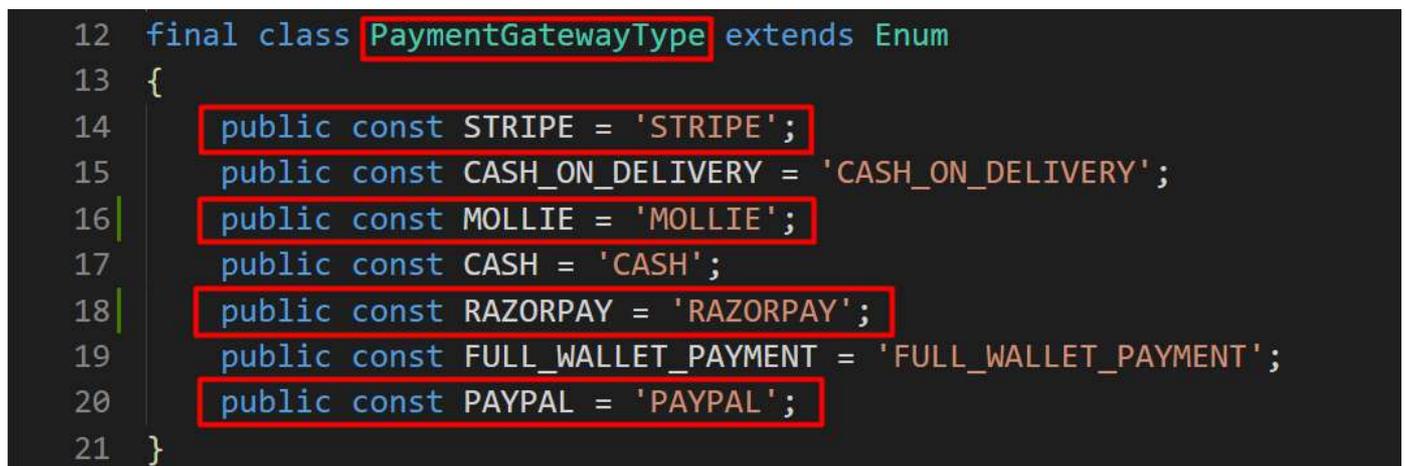
Once you have chosen and installed a package, you will need to configure it by adding your payment gateway credentials and any other required settings.

To check the installed dependencies of payment gateway in Pixier-Laravel, you can open the composer.json file in a text editor and find that.



#### Step 2: Add payment gateway name in the Enum.

Add PaymentGateway Enum API -> package -> marvel -> src -> Enums -> PaymentGatewayType.php



#### Step 3: Configure the Payment Facade for the new payment gateway.

Now go to API -> package -> marvel -> src -> Payment then create new payment Class (e.g. Stripe, PayPal, Razorpay, Mollie) for implements the PaymentInterface. The Class must implements all of the methods defined in the PaymentInterface.

```

17 class Razorpay extends Base implements PaymentInterface
18 {
19     use PaymentTrait;
20
21     public Api $api;
22
23     public function __construct()
24     {
25         parent::__construct();
26         $this->api = new Api(
27             config(
28                 'shop.razorpay.key_id'),
29                 config(
30                     'shop.razorpay.secret_key')
31             );
32     }
33
34     /**
35      * Get payment intent for payment
36      *
37      * @param $data
38      * @return array
39      * @throws MarvelException
40      */
41     public function getIntent($data): array
42     {
43     }
44 }

```

Methods defined in the PaymentInterface

```

5 interface PaymentInterface
6 {
7     public function getIntent(array $data): array;
8
9     public function verify(string $id): mixed;
10
11     public function handleWebHooks(object $request): void;
12
13     public function createCustomer(object $request): array;
14
15     public function attachPaymentMethodToCustomer(
16         string $retrieved_payment_method,
17         object $request): object;
18
19     public function detachPaymentMethodToCustomer(
20         string $retrieved_payment_method): object;
21
22     public function retrievePaymentIntent(
23         string $payment_intent_id): object;
24
25     public function confirmPaymentIntent(
26         string $payment_intent_id,
27         array $data): object;
28
29     public function setIntent(array $data): array;
30
31     public function retrievePaymentMethod(
32         string $method_key): object;
33 }

```

Here's an example of a class that implements that PaymentInterface:

```

1 <?php
2
3 namespace Marvel\Payments;
4
5 use Exception;
6 use Marvel\Database\Models\Order;
7 use Marvel\Database\Models\PaymentIntent;
8 use Marvel\Exceptions\MarvelException;
9 use Marvel\Traits\PaymentTrait;
10 use Razorpay\Api\Api;
11 use Marvel\Enums\OrderStatus;
12 use Marvel\Enums\PaymentStatus;
13 use Razorpay\Api\Errors\SignatureVerificationError;
14 use Str;
15 use Throwable;
16
17 class Razorpay extends Base implements PaymentInterface
18 {
19     use PaymentTrait;
20
21     public Api $api;
22
23     public function __construct()
24     {

```

```
25     parent::__construct();
26     $this->api = new Api(config('shop.razorpay.key_id'), config('shop.razorpay.key_secret'));
27 }
28
29 /**
30  * Get payment intent for payment
31  *
32  * @param $data
33  * @return array
34  * @throws MarvelException
35  */
36 public function getIntent($data): array
37 {
38     try {
39         extract($data);
40         $order = $this->api->order->create([
41             'receipt' => $order_tracking_number,
42             'amount' => round($amount, 2) * 100,
43             'currency' => $this->currency,
44         ]);
45
46         return [
47             'payment_id' => $order->id,
48             'order_tracking_number' => $order->receipt,
49             'currency' => $order->currency,
50             'amount' => $order->amount,
51             'is_redirect' => false,
52         ];
53     } catch (Exception $e) {
54         throw new MarvelException(SOMETHING_WENT_WRONG_WITH_PAYMENT);
55     }
56 }
57
58 /**
59  * Verify a payment
60  *
61  * @param $id
62  * @return false|mixed
63  * @throws MarvelException
64  */
65 public function verify($id): mixed
66 {
67     try {
68         $order = $this->api->order->fetch($id);
69         return isset($order->status) ? $order->status : false;
70     } catch (Exception $e) {
71         throw new MarvelException(SOMETHING_WENT_WRONG_WITH_PAYMENT);
72     }
73 }
74
75 /**
76  * handleWebHooks
77  *
78  * @param mixed $request
79  * @return void
80  * @throws Throwable
81  */
82 public function handleWebHooks($request): void
83 {
84     $webhookSecret = config('shop.razorpay.webhook_secret');
85     $webhookBody = @file_get_contents('php://input');
86     $webhookSignature = $request->header('X-Razorpay-Signature');
87
88     try {
89         if ($webhookBody && $webhookSignature && $webhookSecret) {
90             $this->api->utility->verifyWebhookSignature($webhookBody, $webhookSignature, $webhookSecret);
91         } else {
92             // Invalid request
93             http_response_code(400);
94             exit();
95         }
96     } catch (SignatureVerificationError $e) {
97         // Invalid signature
98         http_response_code(400);
99         exit();
100     }
101
102     $eventStatus = (string) Str::of($request->event)->replace('payment.', '', $request->event);
103
104     switch ($eventStatus) {
105         case 'dispute.won':
106         case 'dispute.created':
107         case 'authorized':
108             $this->updatePaymentOrderStatus($request, OrderStatus::PENDING, PaymentStatus::PROCESSING);
109             break;
110         case 'captured':
```

```

111         $this->updatePaymentOrderStatus($request, OrderStatus::PROCESSING, PaymentStatus::SUCCESS);
112         break;
113         case 'failed':
114             $this->updatePaymentOrderStatus($request, OrderStatus::PENDING, PaymentStatus::FAILED);
115     }
116
117     // To prevent loop for any case
118     http_response_code(200);
119     exit();
120 }
121
122 /**
123  * Update Payment and Order Status
124  *
125  * @param $request
126  * @param $orderStatus
127  * @param $paymentStatus
128  * @return void
129  */
130 public function updatePaymentOrderStatus($request, $orderStatus, $paymentStatus): void
131 {
132     $payload = $request->payload['payment']['entity'];
133     $paymentIntent = PaymentIntent::whereJsonContains('payment_intent_info', ['payment_id' => $payload['order_id']])->first();
134     $trackingId = $paymentIntent->tracking_number;
135     $order = Order::where('tracking_number', '=', $trackingId)->first();
136     $this->webhookSuccessResponse($order, $orderStatus, $paymentStatus);
137 }
138
139 /**
140  * createCustomer
141  *
142  * @param mixed $request
143  * @return array
144  */
145 public function createCustomer($request): array
146 {
147     return [];
148 }
149
150 /**
151  * attachPaymentMethodToCustomer
152  *
153  * @param string $retrieved_payment_method
154  * @param object $request
155  * @return object
156  */
157 public function attachPaymentMethodToCustomer(string $retrieved_payment_method, object $request): object
158 {
159     return (object) [];
160 }
161
162 /**
163  * detachPaymentMethodToCustomer
164  *
165  * @param string $retrieved_payment_method
166  * @return object
167  */
168 public function detachPaymentMethodToCustomer(string $retrieved_payment_method): object
169 {
170     return (object) [];
171 }
172
173 public function retrievePaymentIntent($payment_intent_id): object
174 {
175     return (object) [];
176 }
177
178 /**
179  * confirmPaymentIntent
180  *
181  * @param string $payment_intent_id
182  * @param array $data
183  * @return object
184  */
185 public function confirmPaymentIntent(string $payment_intent_id, array $data): object
186 {
187     return (object) [];
188 }
189
190 /**
191  * setIntent
192  *
193  * @param array $data
194  * @return array
195  */

```

```

196 public function setIntent(array $data): array
197 {
198     return [];
199 }
200
201 /**
202  * retrievePaymentMethod
203  *
204  * @param string $method_key
205  * @return object
206  */
207 public function retrievePaymentMethod(string $method_key): object
208 {
209     return (object) [];
210 }
211 }
212

```

\*\*\* Note : \*\*\* It is important to note that each payment gateway has its own set of requirements and may have different methods for processing payments. You will need to follow the official documentation of that specific payment gateway.

Step 4: Using that payment gateway for submitting the order.

Go to API -> package -> marvel -> src -> Http -> Controllers -> OrderController.php -> submitPayment then you've to add your PaymentGatewayType and function name in switch case.

```

359  */
360  */
361 public function submitPayment(Request $request): void
362 {
363     $tracking_number = $request->tracking_number ?? null;
364     try {
365         $order = $this->repository->with(['products', 'children.shop', 'wallet_point', 'payment_intent'])
366             ->findOneByFieldOrFail('tracking_number', $tracking_number);
367
368         switch ($order->payment_gateway) {
369             Gateway Type
370             case PaymentGatewayType::STRIPE:
371                 $this->stripe($order, $request, $this->settings);
372                 break;
373             function
374             case PaymentGatewayType::PAYPAL:
375                 $this->paypal($order, $request, $this->settings);
376                 break;
377
378             case PaymentGatewayType::MOLLIE:
379                 $this->mollie($order, $request, $this->settings);
380                 break;
381
382             case PaymentGatewayType::RAZORPAY:
383                 $this->razorpay($order, $request, $this->settings);
384                 break;

```

After That go to API -> package -> marvel -> src -> Traits -> PaymentStatusManagerWithOrderTrait.php for verify your payment status update, you've to add function like Stripe, PayPal, Razorpay or Mollie.

```

PaymentStatusManagerWithOrderTrait.php - marvel - Visual Studio Code
107
108 public function Razorpay(Order $order, Request $request, Settings $settings): void
109 {
110     try {
111         $chosen_intent = '';
112         // for single gateway options
113         if (isset($order->payment_intent)) {
114             foreach ($order->payment_intent as $key => $intent) {
115                 if (strtoupper($settings->options['payment_gateway']) === $order->payment_gateway) {
116                     $chosen_intent = $intent;
117                 }
118             }
119         }
120
121         $paymentId = isset($chosen_intent->payment_intent_info) ? $chosen_intent->payment_intent_info['payment_id'] : null;
122         if (isset($paymentId)) {
123             $paymentStatus = Payment::verify($paymentId);
124             if ($paymentStatus) {
125                 switch (strtolower($paymentStatus)) {
126                     case "paid":
127                         $this->paymentSuccess($order);
128                         break;
129                     case "attempted":
130                         $this->paymentProcessing($order);
131                         break;
132                     case "failed":
133                         $this->paymentFailed($order);
134                 }
135             }
136         }
137     } catch (Exception $e) {
138         throw new Exception('SOMETHING WENT WRONG WITH PAYMENT');

```

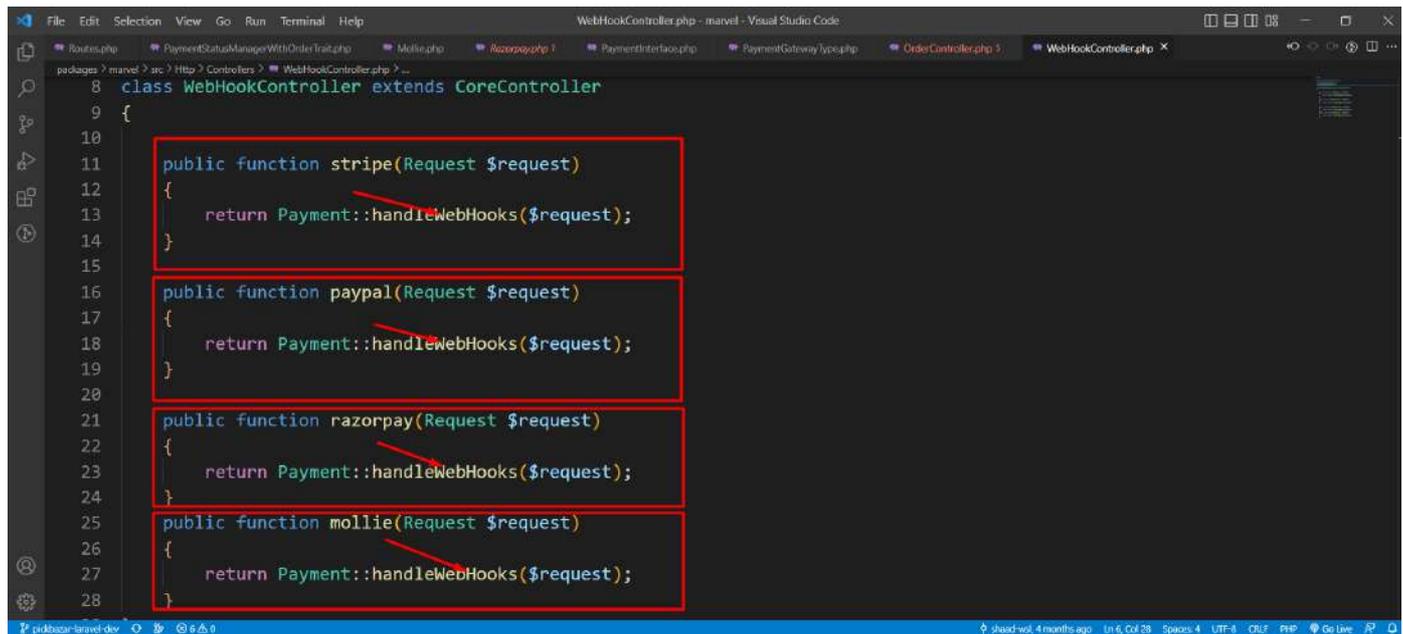
Step 5: How to Setup webhook in Pixier follow the steps

To use a payment gateway webhook, you would first configure the webhook URL in your payment gateway account. Then, whenever the specified events occur, the payment gateway will send an HTTP POST request to the webhook URL with a payload of data about the event.

First go to `API -> package -> marvel -> src -> Routes` then add a post route.

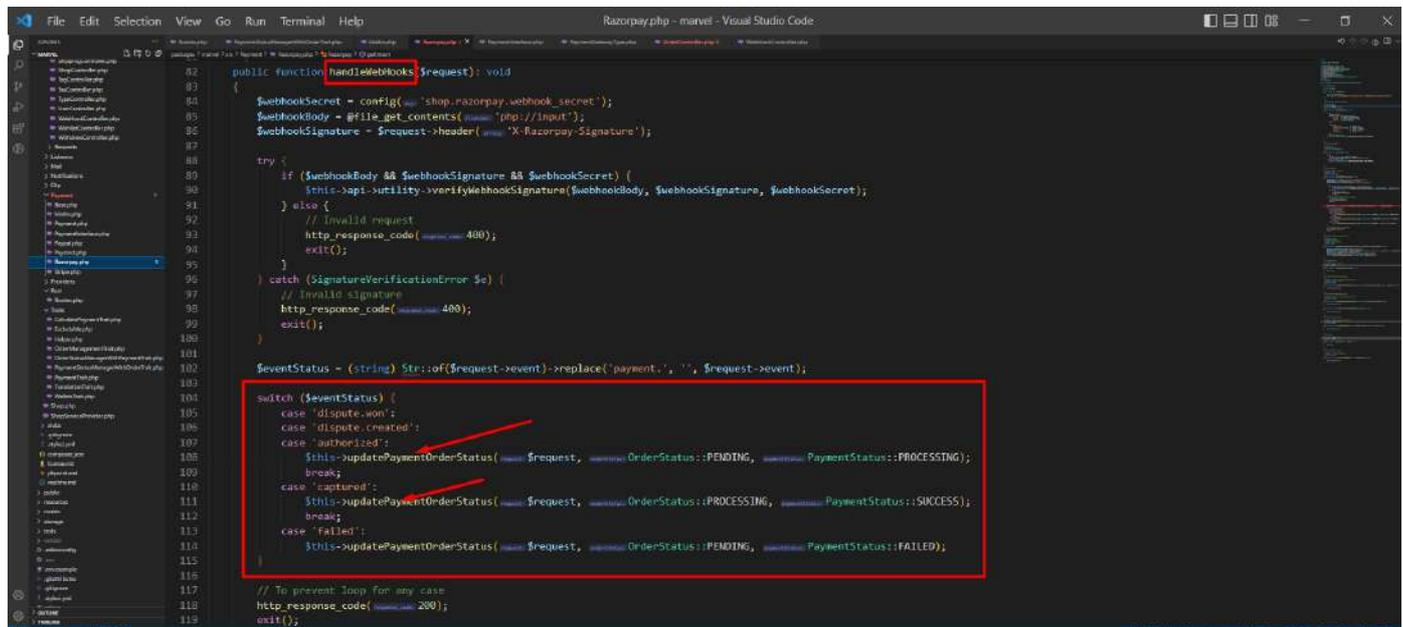
```
73 Route::post(uri: 'webhooks/razorpay', action: [WebHookController::class, 'razorpay']);
74 Route::post(uri: 'webhooks/stripe', action: [WebHookController::class, 'stripe']);
75 Route::post(uri: 'webhooks/paypal', action: [WebHookController::class, 'paypal']);
76 Route::post(uri: 'webhooks/mollie', action: [WebHookController::class, 'mollie']);
```

Add your function in WebHookController



```
8 class WebHookController extends CoreController
9 {
10
11     public function stripe(Request $request)
12     {
13         return Payment::handleWebHooks($request);
14     }
15
16     public function paypal(Request $request)
17     {
18         return Payment::handleWebHooks($request);
19     }
20
21     public function razorpay(Request $request)
22     {
23         return Payment::handleWebHooks($request);
24     }
25
26     public function mollie(Request $request)
27     {
28         return Payment::handleWebHooks($request);
29     }
30 }
```

To handle **webhook events** follow your payment gateway official webhook documentation



```
52 public function handleWebHooks($request): void
53 {
54     $webhookSecret = config('shop.razorpay.webhook_secret');
55     $webhookBody = @file_get_contents('php://input');
56     $webhookSignature = $request->header('X-Razorpay-Signature');
57
58     try {
59         if ($webhookBody && $webhookSignature && $webhookSecret) {
60             $this->api->utility->verifyWebhookSignature($webhookBody, $webhookSignature, $webhookSecret);
61         } else {
62             // Invalid request
63             http_response_code(400);
64             exit();
65         }
66     } catch (SignatureVerificationError $e) {
67         // Invalid signature
68         http_response_code(400);
69         exit();
70     }
71
72     $eventStatus = (string) Str::of($request->event)->replace('payment.', '', $request->event);
73
74     switch ($eventStatus) {
75         case 'dispute-won':
76         case 'dispute-created':
77         case 'authorized':
78             $this->updatePaymentOrderStatus($request, OrderStatus::PENDING, PaymentStatus::PROCESSING);
79             break;
80         case 'captured':
81             $this->updatePaymentOrderStatus($request, OrderStatus::PROCESSING, PaymentStatus::SUCCESS);
82             break;
83         case 'failed':
84             $this->updatePaymentOrderStatus($request, OrderStatus::PENDING, PaymentStatus::FAILED);
85             break;
86     }
87
88     // To prevent loop for any case
89     http_response_code(200);
90     exit();
91 }
```

\*\*\* Note: \*\*\* For locally webhook testing you can use ngrok tools for that. Please follow their official documentation.

Getting Started with admin dashboard.

First go to `marvel-admin -> rest -> types -> index.ts -> export enum PaymentGateway` then add PaymentGateway Name.

```
24 export enum PaymentGateway {
25   STRIPE = 'STRIPE',
26   COD = 'CASH_ON_DELIVERY',
27   CASH = 'CASH',
28   FULL_WALLET_PAYMENT = 'FULL_WALLET_PAYMENT',
29   PAYPAL = 'PAYPAL',
30   MOLLIE = 'MOLLIE',
31   RAZORPAY = 'RAZORPAY',
32 }
```

after that go to `marvel-admin -> rest -> src -> components -> settings -> payment.ts` add name & Title.

```
1 export const PAYMENT_GATEWAY = [
2   { name: 'stripe', title: 'Stripe' },
3   { name: 'paypal', title: 'Paypal' },
4   { name: 'razorpay', title: 'RazorPay' },
5   { name: 'mollie', title: 'Mollie' },
6 ];
```

Then it will automatically add that payment gateway in the marvel-admin settings.

Getting Started with shop front.

There are two types of payment gateway system can be integrated here.

- Redirect based payment gateway (e.g PayPal). Where the customer will redirect to that payment gateway site during order checkout. Complete the payment there. And then comeback to the application.
- Non redirect based payment gateway. Where the customer will stay on the application and complete the whole payment process here. Here we consider Stripe as a non-redirect based payment gateway. Though Stripe has features too similar to redirect based payment gateway.

Redirect-base Payment Gateway

if you want to integrate redirect based payment gateway (e.g. PayPal, Mollie). follow the steps

First go to `shop -> src -> types -> index.ts -> export enum PaymentGateway` then add PaymentGateway Name.

```
24 export enum PaymentGateway {
25   STRIPE = 'STRIPE',
26   COD = 'CASH_ON_DELIVERY',
27   CASH = 'CASH',
28   FULL_WALLET_PAYMENT = 'FULL_WALLET_PAYMENT',
29   PAYPAL = 'PAYPAL',
30   MOLLIE = 'MOLLIE',
31   RAZORPAY = 'RAZORPAY',
32 }
```

after that go to `shop -> src -> components -> checkout -> payment -> Payment-grid.tsx` then add your PaymentGateway object.

```

77 > = {
78   STRIPE: {
79     name: 'Stripe',
80     value: PaymentGateway.STRIPE,
81     icon: '/payment/stripe.png',
82     component: PaymentOnline,
83   },
84   PAYPAL: {
85     name: 'Paypal',
86     value: PaymentGateway.PAYPAL,
87     icon: '',
88     // icon: '/payment/paypal.png',
89     component: PaymentOnline,
90   },
91   RAZORPAY: {
92     name: 'RazorPay',
93     value: PaymentGateway.RAZORPAY,
94     icon: '/payment/razorpay.png',
95     component: PaymentOnline,
96   },
97   MOLLIE: {
98     name: 'Mollie',
99     value: PaymentGateway.MOLLIE,
100    icon: '/payment/mollie.png',
101    component: PaymentOnline,
102  },

```

### Non-redirect based payment gateway

First, complete the redirect-based payment gateway steps mentioned above. Because that two steps is universal for all payment gateway to apply.

After That go to `shop -> src -> components -> payment` then add a folder like Stripe, Razorpay. Inside your payment folder you can create your required typescript files with related functionalities for your Payment Gateway. For example, you can checkout the Stripe folder. You can find all the necessary indication, components guide, payment-method (card) saving options etc in the Stripe folder.

```

16 interface Props {
17   paymentIntentInfo: PaymentIntentInfo;
18   trackingNumber: string;
19   paymentGateway: PaymentGateway;
20 }
21
22 const PaymentForm: React.FC<Props> = ({
23   paymentIntentInfo,
24   trackingNumber,
25   paymentGateway,
26 }) => {
27   const { t } = useTranslation('common');
28   const stripe = useStripe();
29   const elements = useElements();

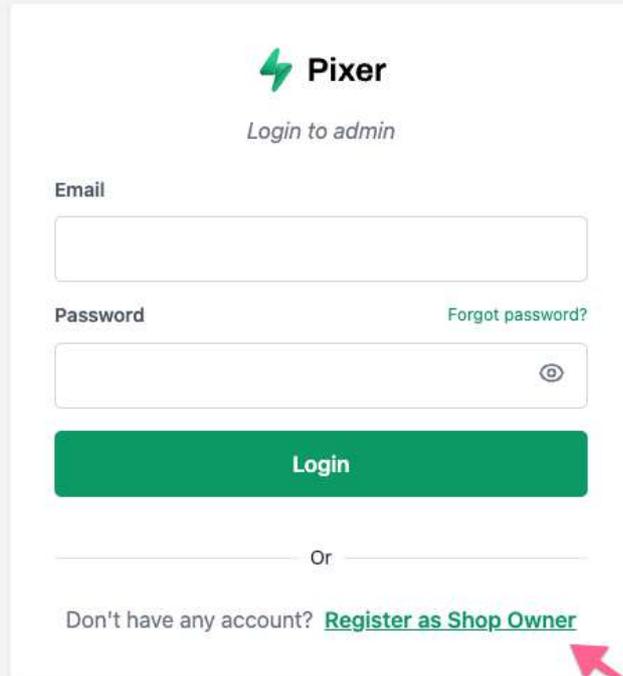
```

That's all for today. If you need any more help you can always contact with our support agents in our support portal.

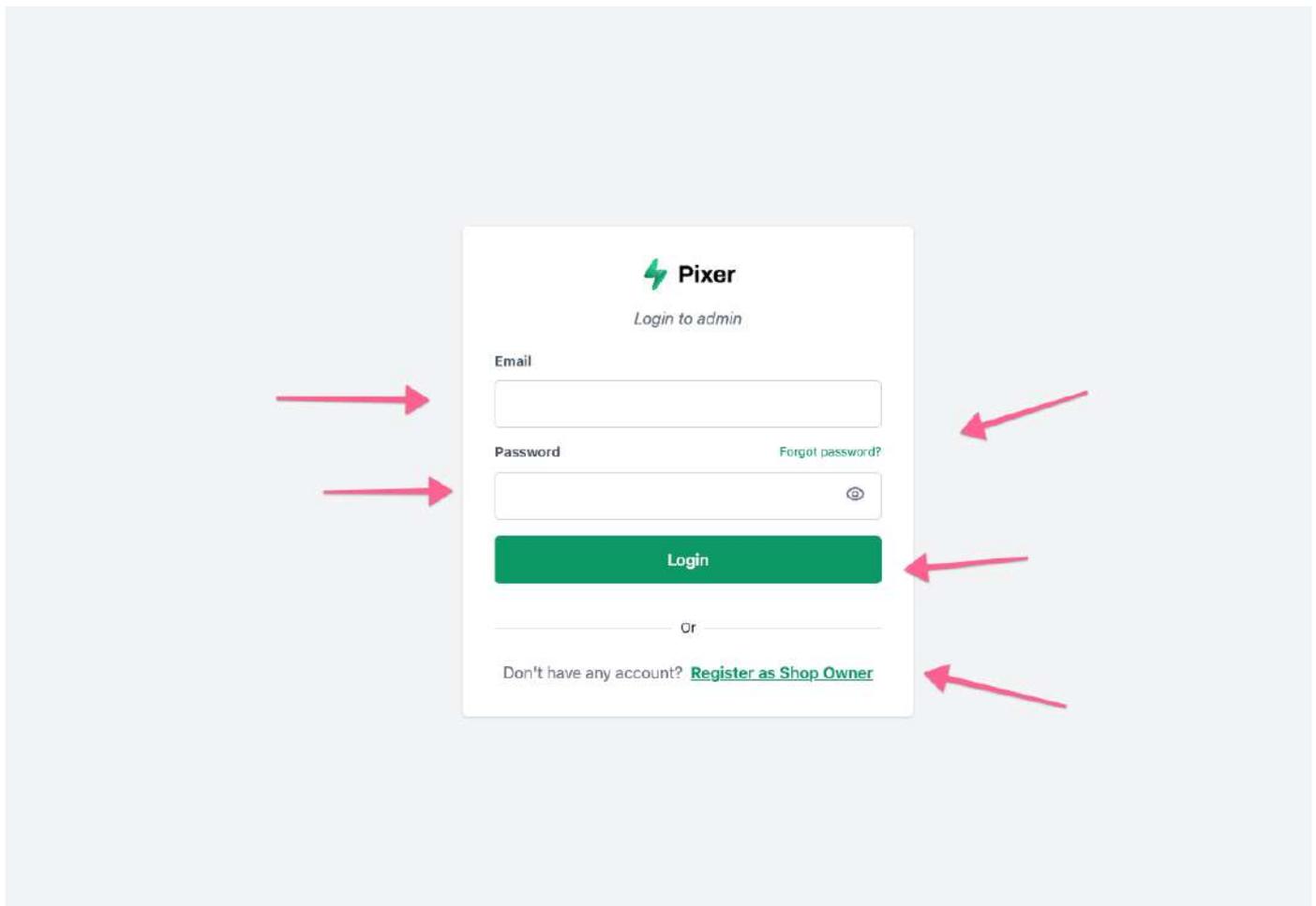
## Multivendor

### Create New Shop

To create new shop login as an **administrator** or create a new account for creating **shop**



The image shows a login form for the Pixier admin interface. At the top, there is a green lightning bolt icon followed by the text "Pixier". Below this is the subtitle "Login to admin". The form contains two input fields: "Email" and "Password". The "Password" field has a toggle icon on the right. A green "Login" button is positioned below the password field. Below the button is a horizontal line with the word "Or" in the center. At the bottom of the form, there is a link that says "Don't have any account? [Register as Shop Owner](#)". A red arrow points to this link.



After creating the account you'll be redirected to this page,

**Summary**

- Total Revenue: \$0.00
- Total refunds: \$0.00
- Total Shops: 10
- Todays Revenue: \$0.00

**Order Status**

- Pending Order: 0
- Processing Order: 0
- Completed Order: 0
- Cancelled Order: 0

**Sale History**

Line chart showing sales from January to December. The y-axis ranges from 0.0 to 2.0. No data is visible on the chart.

**Top 10 Most Rated Products**

No data found. Sorry we couldn't find any data.

**Top 10 Category with most products**

Category ID	Category Name	Shop	Product Count
#ID: 9	Wireframe Kits	BentaSoft	5
#ID: 9	Wireframe Kits	Imagineco	5
#ID: 9	Wireframe Kits	BentaSoft	5
#ID: 9	Wireframe Kits	Imagineco	5
#ID: 7	Angular	Gubitron Solutions	4
#ID: 6	WordPress Theme	BentaSoft	5
#ID: 9	Wireframe Kits	BentaSoft	5
#ID: 5	WordPress Plugin	BentaSoft	4
#ID: 17	Shopify	BentaSoft	4
#ID: 7	Angular	BentaSoft	4
#ID: 11	Illustrations	BentaSoft	4
#ID: 8	CMS	Omatron	4
#ID: 16	Joomla	Maxicon Soft Tech	4
#ID: 15	Bootstrap	BentaSoft	4

After that, click [Create Shop](#)

**Create Shop**

**Logo**  
Upload your shop logo from here

Upload an image or drag and drop  
PNG, JPG

**Cover Image**  
Upload your shop cover image from here  
Dimension of the cover image should be 1170 x 435px

Upload an image or drag and drop  
PNG, JPG

**Basic Info**  
Add some basic info about your shop from here

Name \*

  
Slug  
  
Description

**Payment Info**  
Add your payment information from here

Account Holder Name \*

  
Account Holder Email \*  
Bank Name \*  
Account Number \*

**Shop Address**  
Add your physical shop address from here

Country  
  
City  
  
State  
  
ZIP  
  
Street Address

**Email Notification**  
Set your email notification for messaging feature

Notification email  
  
 Enable Notification

**Shop Settings**  
Add your shop settings information from here

Contact Number \*

  
Website \*

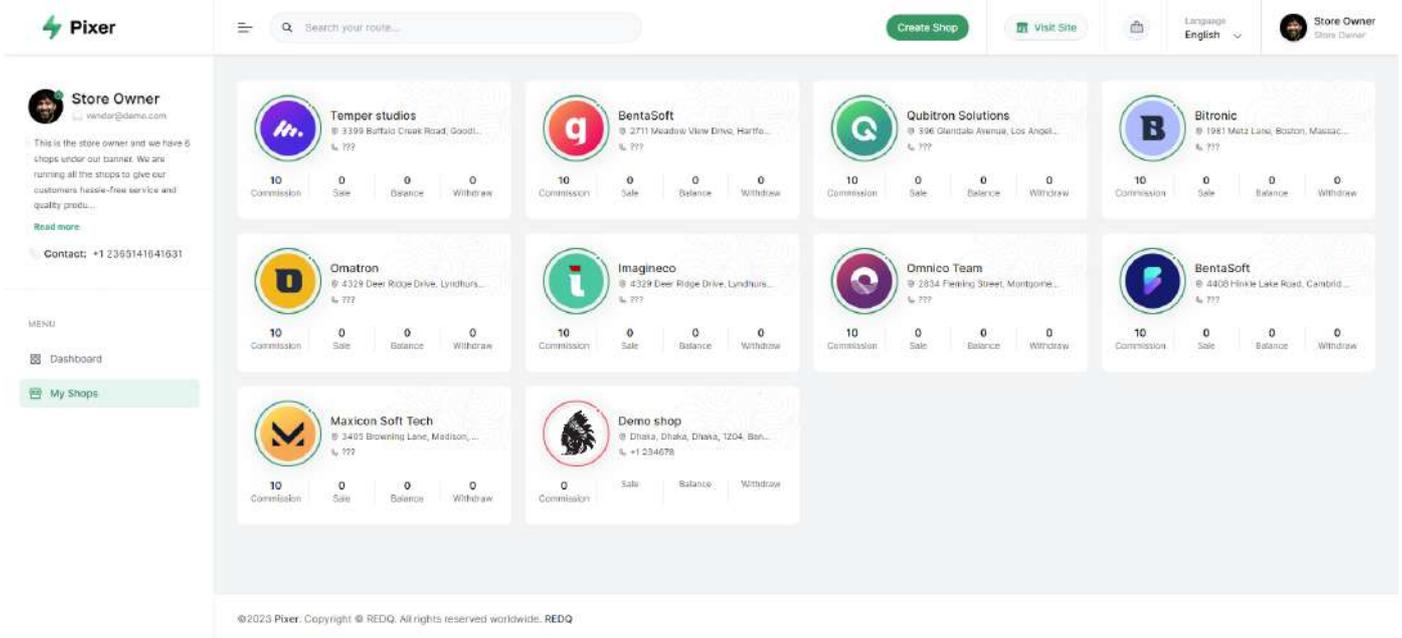
**Social Profile Settings**  
Add your social profile information from here

[Add New Social Profile](#)

[Save](#)

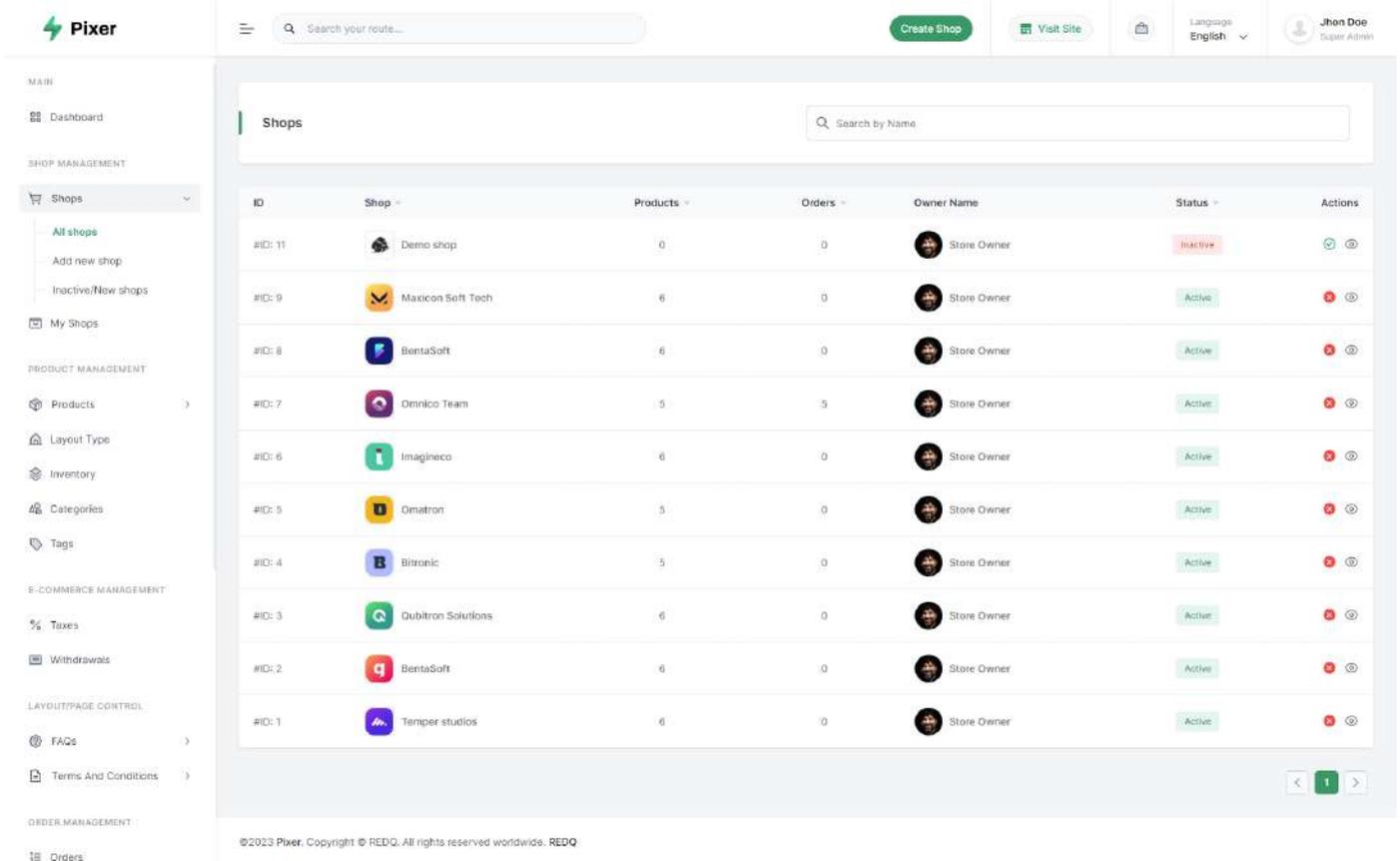
And provide all the information for the [store](#).

After creating the shop you'll redirect to this page,

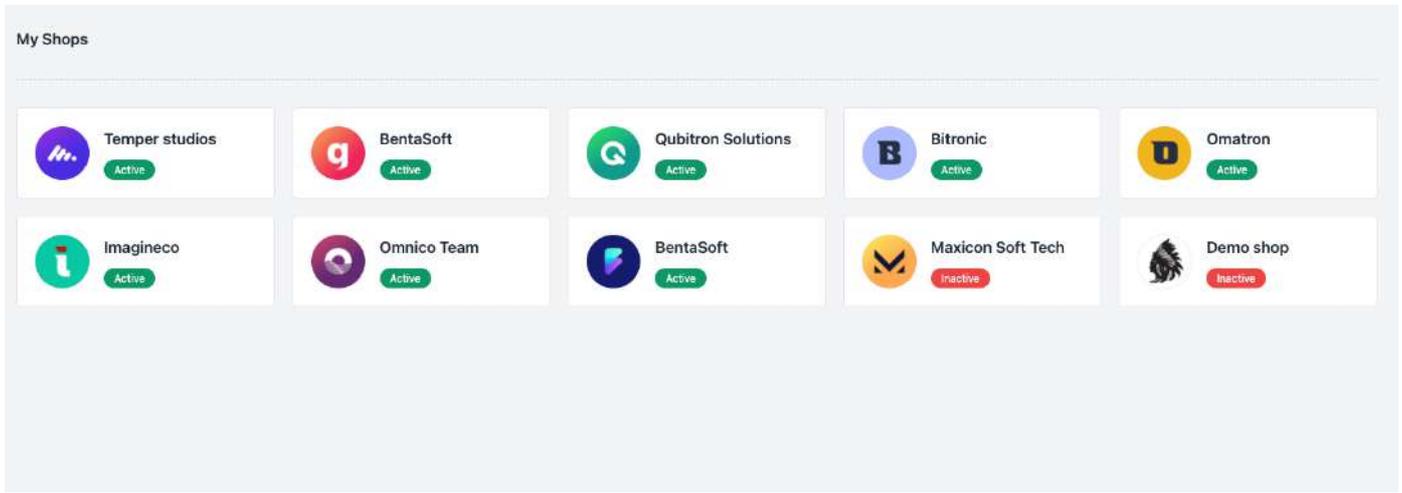


By default, the shop will be inactive. Only **administrator** can activate a shop.

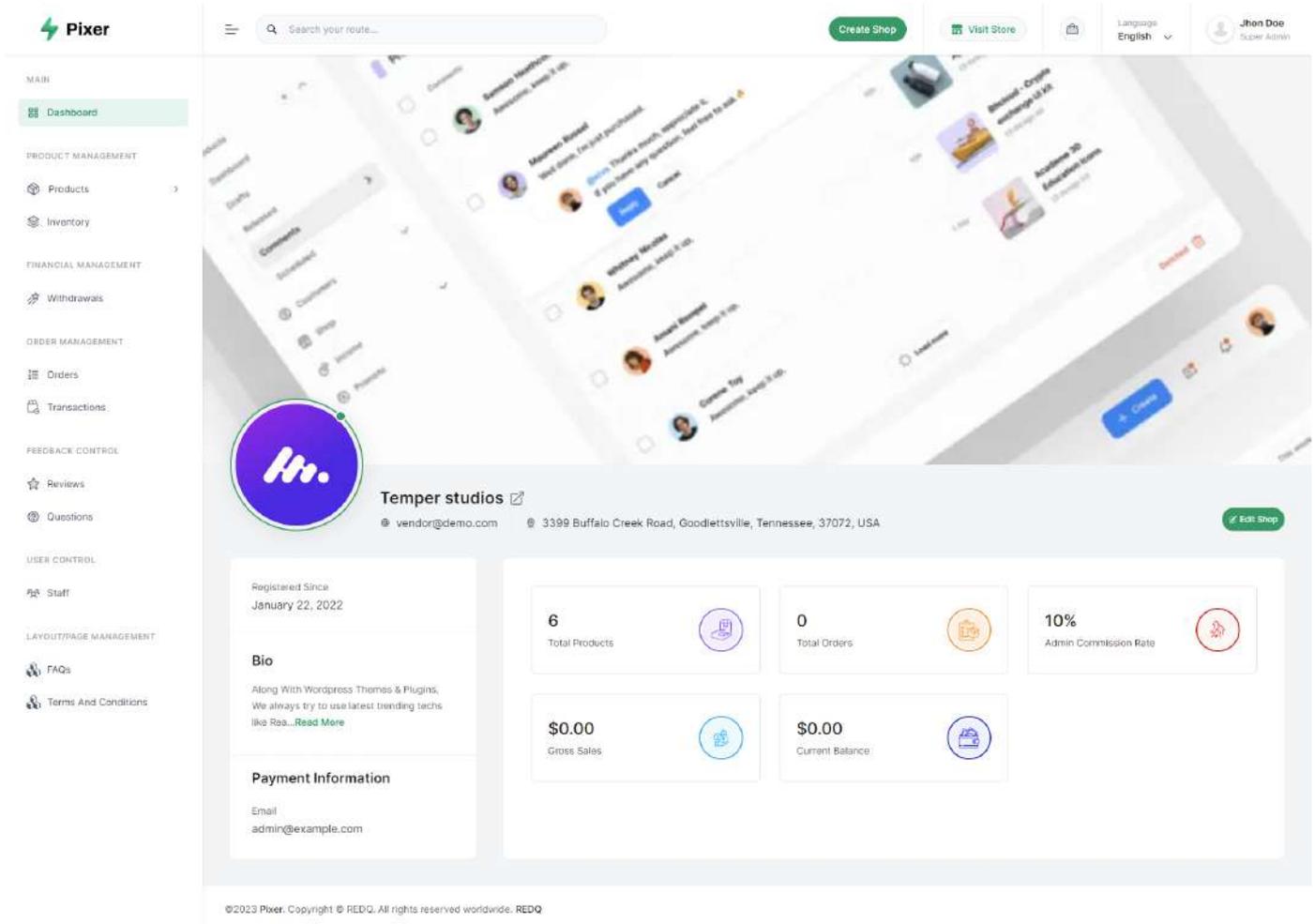
From **administrator** account go to shop and click tick mark to activate or deactivate a shop.



After activate the shop by **administrator** the **vendor** dashboard will be like this,



After click on [shop](#), you'll be redirected to dashboard page,



From this dashboard, you can maintain you shop,

**Dashboard:**

**Pixier** Create Shop

- Dashboard
- Products
- Orders
- Refunds
- Staffs
- Withdraws

**BentaSoft**  
Along With Wordpress Themes & Plugins, We always try to use latest tre...  
[Read more](#)

2711 Meadow View Drive, Hartford, Connecticut, 06103, USA  
+659033724396

Visit Shop

**Products**

6 Total Products

3 Total Orders

**Revenue**

\$79.20 Gross Sales

\$79.20 Current Balance

**Others**

10% Admin Commission Rate

Registered Since  
January 22, 2022

**Payment Information**

Name:  
Email: rnb@rnc.md  
Bank:  
Account No.:

**Products:**

**Pixier** Create Shop

- Dashboard
- Products
- Orders
- Refunds
- Staffs
- Withdraws

**Products**

Q Type your query and press enter + Add Product Filter ?

Image	Name	Group	Price/Unit	Quantity	Status	Actions
	Dashify WordPress Elementor Dashboard Theme	Liquid	\$29.00	500	publish	
	Superify	N/A	\$19.00	500	publish	
	Businessmind GraphQL React WordPress Business Theme	Responsive	\$25.00	500	publish	
	Portello React Gatsby Portfolio Template With Shopify	Responsive	\$0.00	500	publish	
	Aptily Swift Ecommerce Theme	Liquid	\$19.00	500	publish	
	COvily Bootstrap CMS Template	Fixed	\$49.00	500	publish	

< 1 >

**Order:**

**Pixer** Create Shop

- Dashboard
- Products
- Orders**
- Refunds
- Staffs
- Withdraws

### Orders

Type your query and press enter

Tracking Number	Total	Order Date	Status	Actions
z0YQ60QZ4uFn	\$195.00	2 days ago	Confirmed	
LzCpzkuJz0B	\$65.00	3 days ago	Confirmed	
ooIZNgwARbz	\$65.00	3 days ago	Confirmed	
emHREducFZVq	\$90.00	5 days ago	Confirmed	
07BYulHcUC9	\$30.00	5 days ago	Confirmed	
ZXGm5EAuQWq	\$65.00	6 days ago	Confirmed	
kqRfdNwRO06	\$65.00	12 days ago	Confirmed	
iDnefVNRkkmH	\$45.00	12 days ago	Confirmed	
FP8lyZvoc5Qd	\$325.00	13 days ago	Confirmed	
wZiVNgZZlGe	\$195.00	14 days ago	Confirmed	

< 1 2 >

**Staff:**

**Pixer** Create Shop

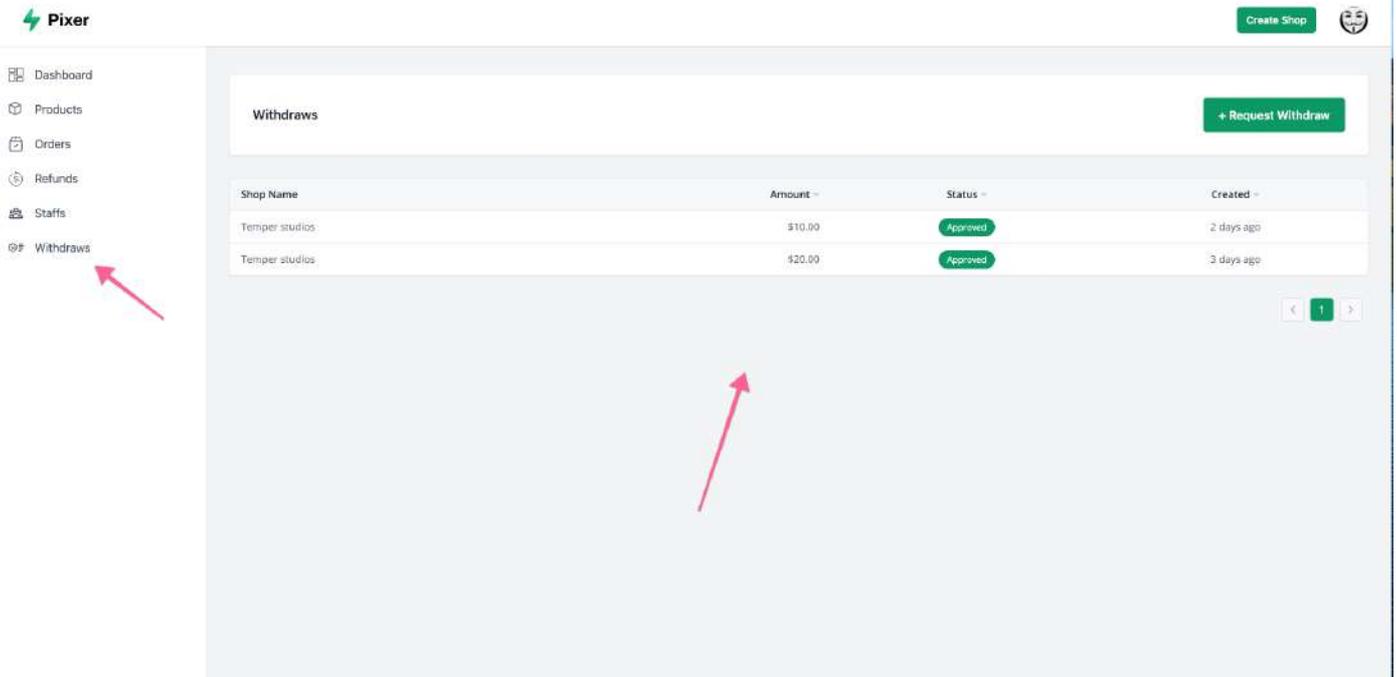
- Dashboard
- Products
- Orders
- Refunds
- Staffs**
- Withdraws

### Staff

+ Add Staff

Name	Email	Status	Actions
No data found			

**WithDraw:**



### User Roles:

#### Super Admin:

Super admin can do everything. the admin can maintain and edit every store on the site.

#### Store Owner:

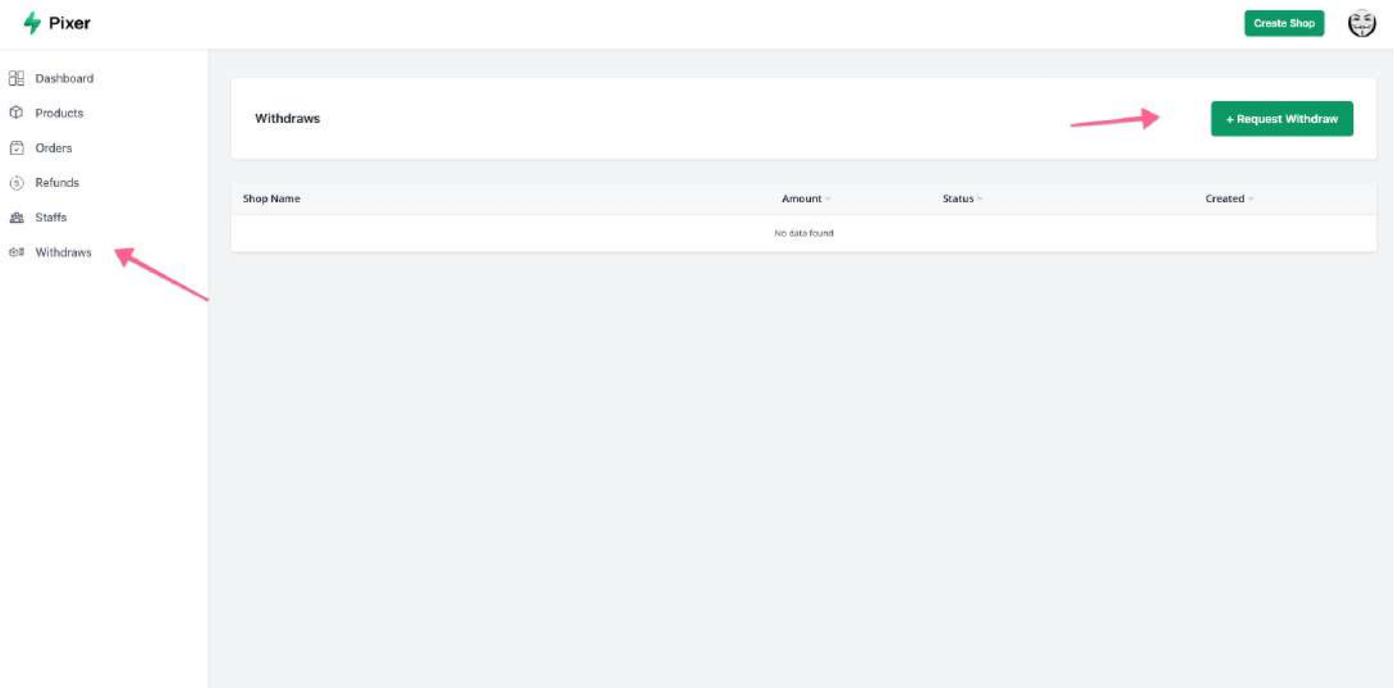
Store owner can edit or main it's store, staff or payment.

#### Staff:

The staff of a store has similar permission as store owner, but the staff can't update the store or withdraw payment.

### Withdraw Payment:

Only the store owner can withdraw its payment. To do that go to your shop dashboard -> withdraws -> Request a withdraw,



- Dashboard
- Products
- Orders
- Refunds
- Staffs
- Withdraws

### Create Withdraw

**Description**  
Add withdraw request from here

**Amount** (Available Balance: \$94.20)

**Payment Method**

**Details**

**Note**

[Request Withdraw](#)

After request payment, the dashboard will be like this,

### Withdraws

[+ Request Withdraw](#)

Shop Name	Amount	Status	Created
Shop 2	\$500.00	Pending	a few seconds ago

[<](#) [1](#) [>](#)

After request, the admin has to be approved the admin.

**Pixier** Create Shop

- Dashboard
- Shops
- My Shops
- Products
- Layouts Type
- Categories
- Tags
- Orders
- Order Status
- Create Order
- Users
- Taxes
- Withdraws
- Refunds
- Settings

### Withdraws

Shop Name	Amount	Status	Created	Actions
Temper studios	\$10.00	Approved	2 days ago	
Maxicon Soft Tech	\$200.00	Rejected	2 days ago	
Temper studios	\$20.00	Approved	3 days ago	

< 1 >

**Pixier** Create Shop

- Dashboard
- Shops
- My Shops
- Products
- Layouts Type
- Categories
- Tags
- Orders
- Order Status
- Create Order
- Users
- Taxes
- Withdraws
- Refunds
- Settings

### Withdrawal Information

Amount :

Payment Method : **Bank**

Status : **Rejected**

Details :  
Bank details

Rejected Change Status

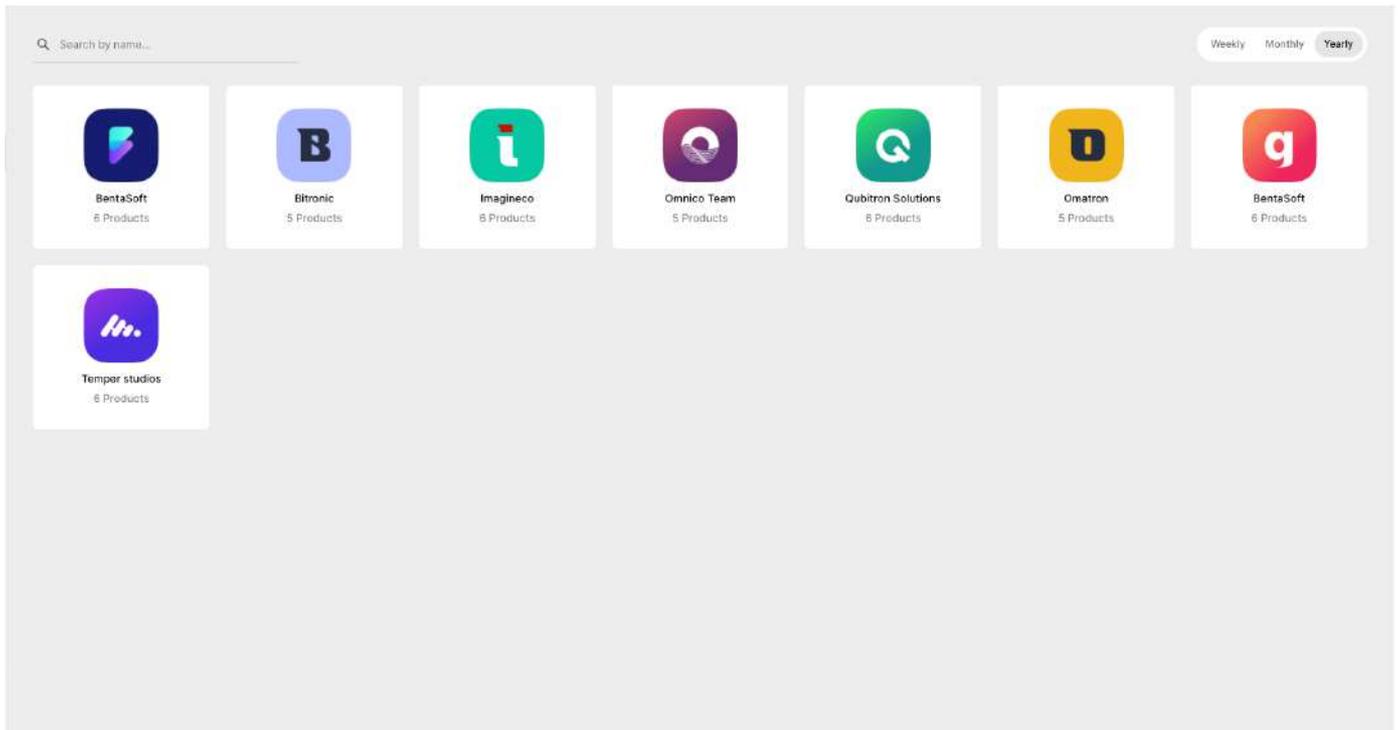
- Approved
- On Hold
- Processing
- Pending
- Rejected

After approved,

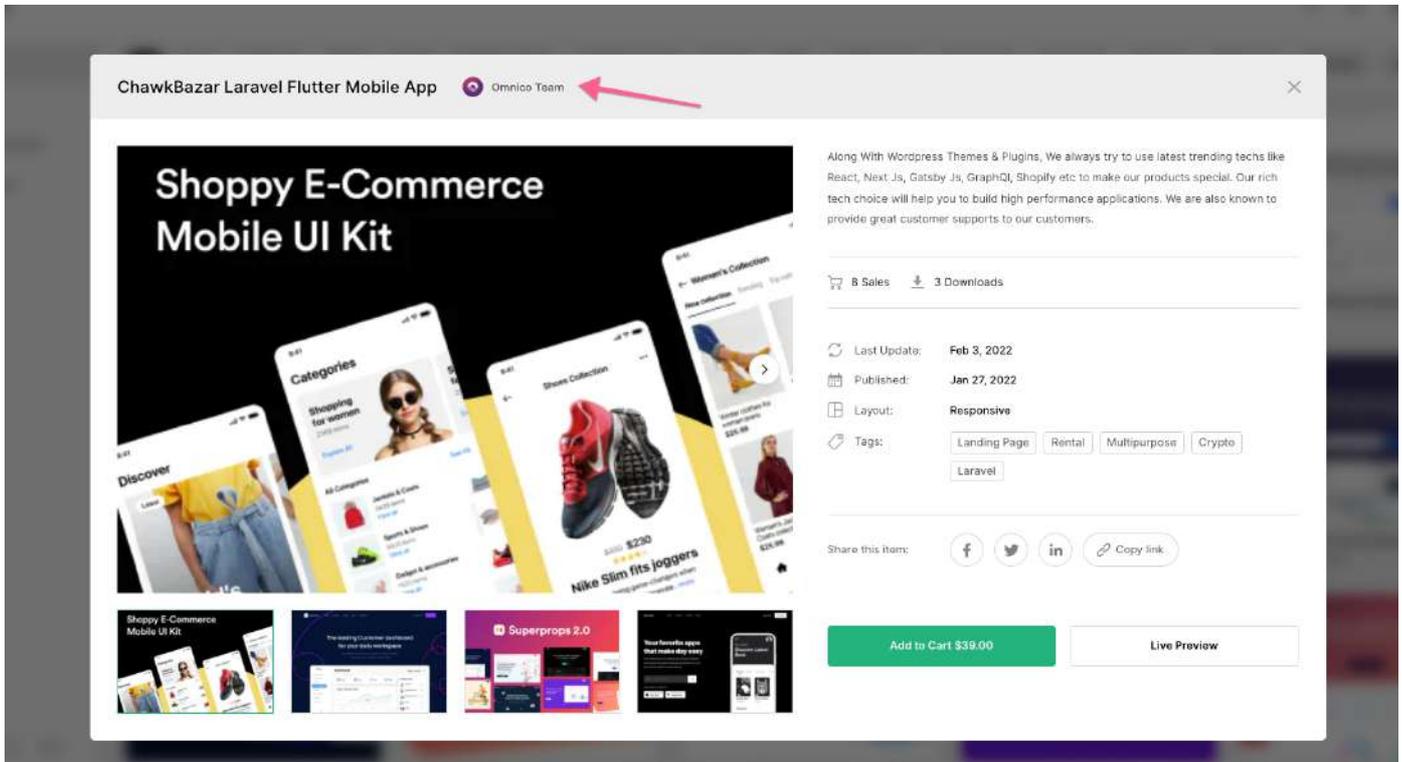
Shop Name	Amount	Status	Created
Shop 2	\$500.00	Approved	3 minutes ago

### FrontEnd Shop

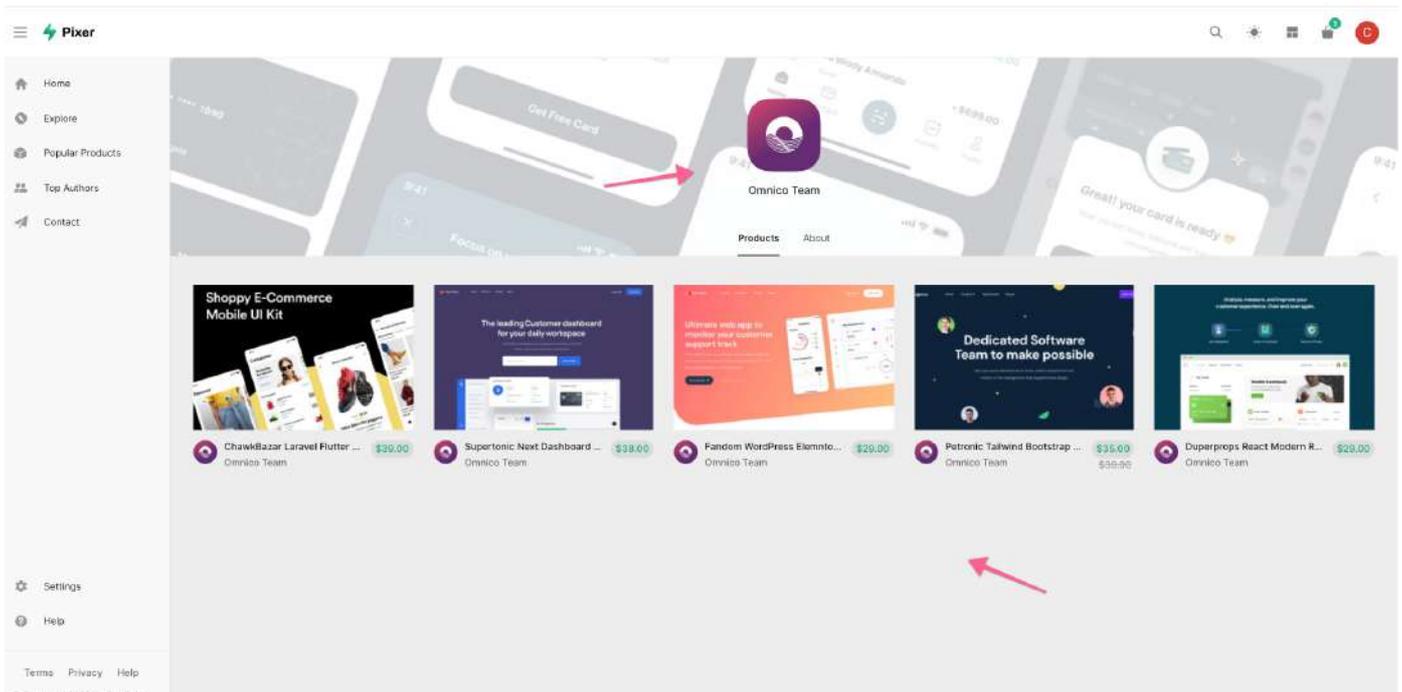
From frontend when customers click on the [shop](#) page they'll get all the shops as a card list,



Also when customers click on a product they will get the seller link,



After clicking the **seller** link they will redirect to the seller shop page,



## Multilingual

If your business or site doesn't require multiple languages, then we don't recommend enabling the multilanguage feature. It'll increase the complexity of maintaining all the language simultaneously. Only enable it if you know what multilanguage is and your business or site needs that feature.

### Step 1: Enable Multilingual Feature

API:

At first open `api` -> `.env` and make true of ``

```
TRANSLATION_ENABLED=true
```

```

1 APP_NAME=Marvel
2 APP_ENV=local
3 APP_KEY=base64:3dEjhUvaWdcDL3MYe4V5QRXWm0yqsdiyqkS1V9v8Bg=
4 APP_DEBUG=true
5 APP_URL=http://localhost:8000
6 APP_SERVICE=marvel.test
7 APP_NOTICE_DOMAIN=PIXER_
8
9 DUMMY_DATA_PATH=pixer
10
11
12 # Multilang
13 # If you want to enable multilang then follow this doc -> https://pixer-doc.vercel.app/multilingual
14 TRANSLATION_ENABLED=true
15 DEFAULT_LANGUAGE=en
16
17 LOG_CHANNEL=stack
18 LOG_LEVEL=debug
19
20 DB_CONNECTION=mysql
21 DB_HOST=localhost
22 DB_PORT=3306
23 DB_DATABASE=laravel_pixer_doc
24 DB_USERNAME=root
25 DB_PASSWORD=
26
27 BROADCAST_DRIVER=log
28 CACHE_DRIVER=file
29 QUEUE_CONNECTION=sync
30 SESSION_DRIVER=file
31 SESSION_LIFETIME=120

```

## Admin

For admin,

update `admin` -> `.env` and update,

```
NEXT_PUBLIC_ENABLE_MULTI_LANG=true
```

And add available language to `NEXT_PUBLIC_AVAILABLE_LANGUAGES` with a comma separator.

For example, at your site, you want to support three languages, one is English, and the rest of the two will be german and Arabic. Then `NEXT_PUBLIC_AVAILABLE_LANGUAGES` will be like this,

```
NEXT_PUBLIC_AVAILABLE_LANGUAGES=en,de
```

```

1 # Don't add '/' after the URL
2 NEXT_PUBLIC_REST_API_ENDPOINT="http://127.0.0.1:8000"
3 NEXT_PUBLIC_SHOP_URL="http://localhost:3000"
4
5 # Application Mode and Authentication key
6 APPLICATION_MODE=production
7 NEXT_PUBLIC_AUTH_TOKEN_KEY=AUTH_CRED
8
9 # Default Language
10 NEXT_PUBLIC_DEFAULT_LANGUAGE=en
11
12 # Multilang
13 # If you want to enable multilang then follow this doc -> https://pixer-doc.vercel.app/multilingual
14 NEXT_PUBLIC_ENABLE_MULTI_LANG=true
15 NEXT_PUBLIC_AVAILABLE_LANGUAGES=en,de
16
17 # API key for third party service
18 NEXT_PUBLIC_GOOGLE_MAP_API_KEY=

```

## Shop

For shop, similarly,

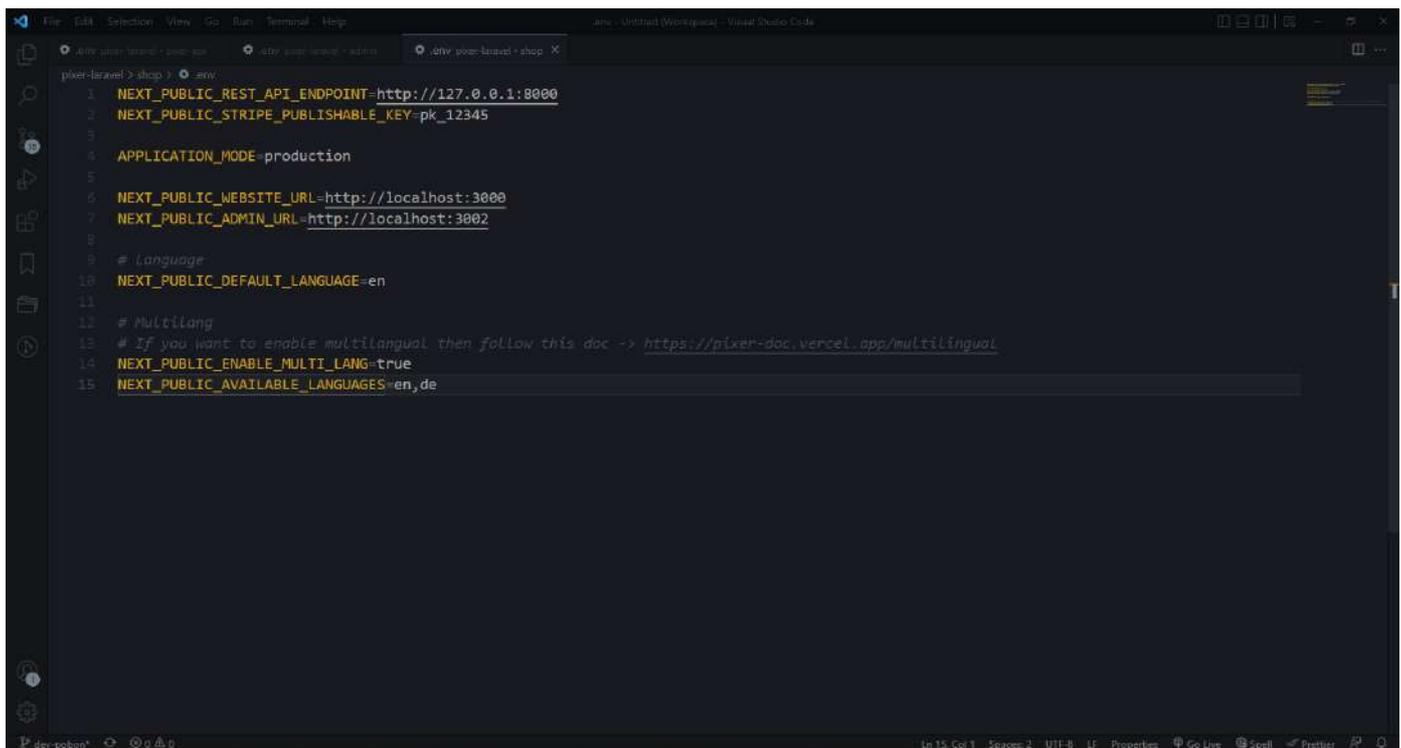
update `shop` -> `.env` and update,

```
NEXT_PUBLIC_ENABLE_MULTI_LANG=true
```

And add available language to `NEXT_PUBLIC_AVAILABLE_LANGUAGES` with a comma separator.

For example, at your site, you want to support three languages, one is English, and the rest of the two will be german and Arabic. Then `NEXT_PUBLIC_AVAILABLE_LANGUAGES` will be like this,

```
NEXT_PUBLIC_AVAILABLE_LANGUAGES=en,de
```

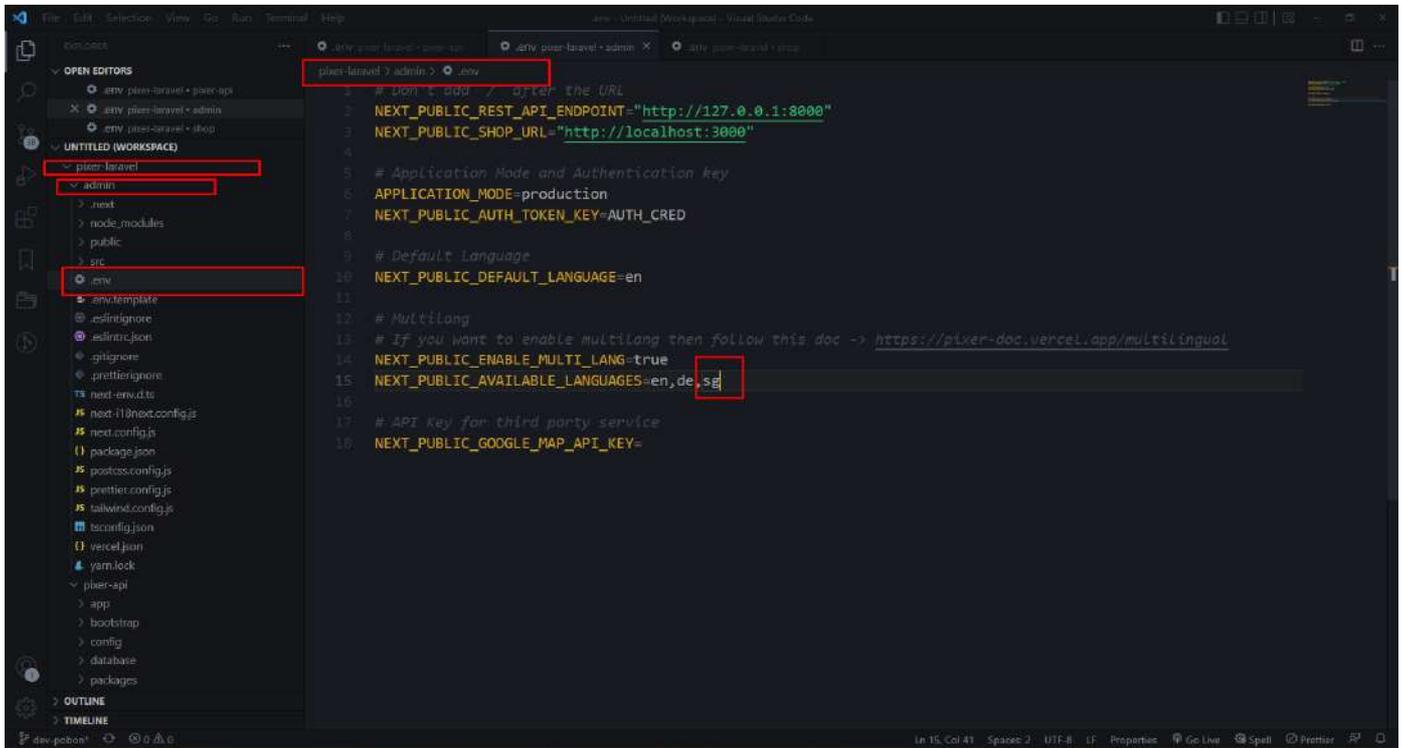


```
1 NEXT_PUBLIC_REST_API_ENDPOINT=http://127.0.0.1:8000
2 NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_12345
3
4 APPLICATION_MODE=production
5
6 NEXT_PUBLIC_WEBSITE_URL=http://localhost:3000
7 NEXT_PUBLIC_ADMIN_URL=http://localhost:3002
8
9 # Language
10 NEXT_PUBLIC_DEFAULT_LANGUAGE=en
11
12 # Multilang
13 # If you want to enable multilingual then follow this doc -> https://pixier-doc.vercel.app/multilingual
14 NEXT_PUBLIC_ENABLE_MULTI_LANG=true
15 NEXT_PUBLIC_AVAILABLE_LANGUAGES=en,de
```

After that, if you already host your site on server then make sure you redeploy the API and rebuild the frontend,

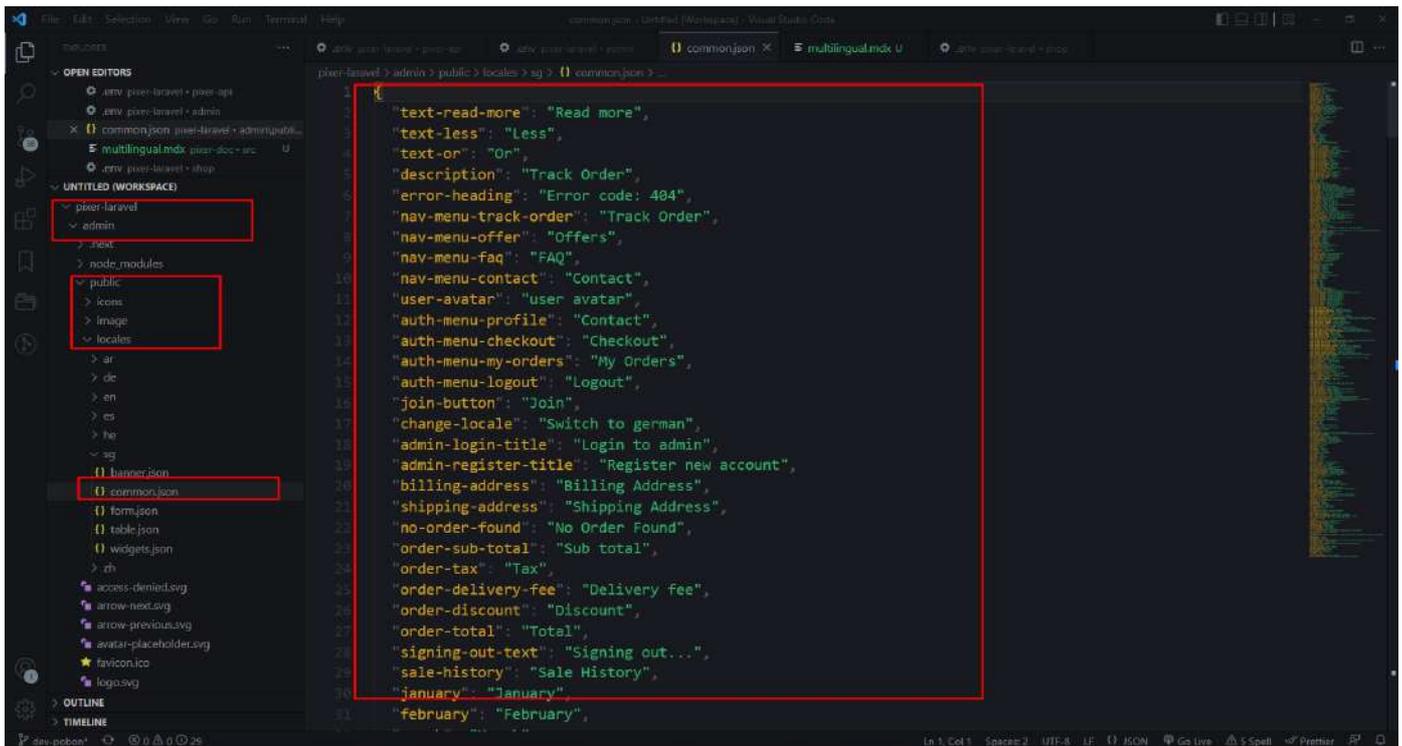
<https://pixier-doc.vercel.app/faq#how-to-rebuild-the-project>

## Step 2: How to add new language in admin?



File Location

admin\public\locales\sg\banner.json



File Location

admin\src\utils\locals.tsx

The screenshot shows the VS Code editor with the file `local.tsx` open. The file contains a configuration object for the localization menu. The entry for 'Sg' (Singapore) is highlighted with a red box. The configuration for 'Sg' is as follows:

```

    id: 'Sg',
    name: 'Singapore',
    value: 'sg',
    icon: <SGFlag width="20px" height="15px" />,
  },

```

### File Location

admin/src/components/icons/flags\

The screenshot shows the VS Code editor with the file `SGFlag.tsx` open. The file contains the definition of the `SGFlag` component. The component is defined as follows:

```

export const SGFlag = ({ width = '640px', height = '480px' }) => {
  return (
    <svg
      xmlns="http://www.w3.org/2000/svg"
      id="flag-icon-css-us"
      viewBox="0 0 640 480"
      width={width}
      height={height}
    >
      <g fillRule="evenodd">
        <g strokeWidth="1pt">
          <path
            fill="#bd3d44"
            d="M0 0h972.8v39.4H0z m0 78.8h972.8v197H0z m0 78.8h972.8v39.4H0z m0 78.8h972.8v39.4H0z m0 78.8h972.8v197H0z"
            transform="scale(.9375)"
          />
          <path
            fill="#fff"
            d="M0 39.4h972.8v39.4H0z m0 78.8h972.8v39.4H0z m0 78.8h972.8v39.4H0z m0 78.8h972.8v197H0z"
            transform="scale(.9375)"
          />
        </g>
        <path fill="#192f5d" d="M0 0h389.1v275.7H0z" transform="scale(.9375)" />
        <path
          fill="#fff"
          d="M32.4 11.8L36 22.7h11.4l-9.2 6.7 3.5 11-9.3-6.8-9.2 6.7 3.5-10.9-9.3-6.7H29zm64.9 0l3.5 10.9h11.5l-9.3 6.7 3.5 11-9.3-6.8-9.2 6.7 3.5-10.9-9.3-6.7H11.5zm64.9 0l3.5 10.9H242l-9.3 6.7 3.6 11-9.3-6.8-9.3 6.7 3.6-10.9-9.3-6.7H11.4zm64.8 0l3.6 10.9h11.4l-9.2 6.7 3.5 11-9.3-6.8-9.2 6.7 3.5-10.9-9.2 6.7h11.4"
        />
      </g>
    </svg>
  );
};

```

Step 3: How to add new language in shop?

```

NEXT_PUBLIC_REST_API_ENDPOINT=http://127.0.0.1:8000
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_12345
APPLICATION_MODE=production
NEXT_PUBLIC_WEBSITE_URL=http://localhost:3000
NEXT_PUBLIC_ADMIN_URL=http://localhost:3002
# Language
NEXT_PUBLIC_DEFAULT_LANGUAGE=en
# Multilang
# If you want to enable multilingual then follow this doc -> https://pixer-doc.vercel.app/multilingual
NEXT_PUBLIC_ENABLE_MULTI_LANG=true
NEXT_PUBLIC_AVAILABLE_LANGUAGES=en,de,sg

```

## File Location

shop\public\locales\sg\common.json

```

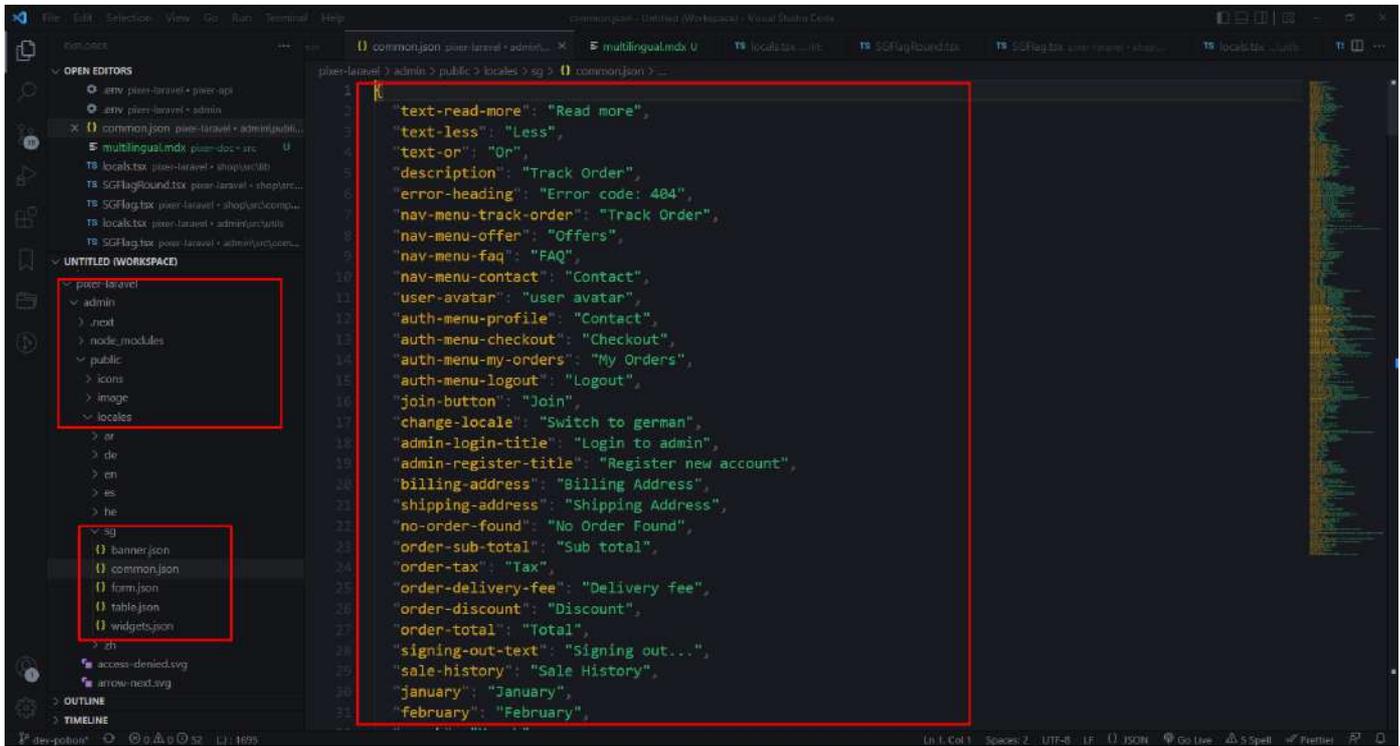
"404-heading": "No page found",
"404-sub-heading": "Oops! The page you were looking for isn't here.",
"404-back-home": "Back to home",
"contact-us-title": "Need help? Contact us",
"contact-us-subtitle": "Looking for help? Drop your contact details here",
"contact-us-info-title": "Contact Information",
"contact-us-info-subtitle": "Fill out the form and our team will get back to you within 24 hours.",
"contact-us-office-title": "Office Location",
"contact-us-office-message": "Add your office location from admin panel",
"contact-us-phone-title": "Call us anytime",
"contact-us-phone-message": "Add your contact info from admin panel",
"contact-us-site-title": "Visit Website",
"contact-us-site-message": "Add your website info from admin panel",
"contact-us-name-field": "Name",
"contact-us-email-field": "Email",
"contact-us-subject-field": "Subject",
"contact-us-message-field": "Message",
"contact-us-submit-button": "Contact Us",
"text-followed-authors": "Followed Authors",
"text-help-page-title": "Help",
"text-help-page-subtitle": "Frequently asked question",
"text-license-page-title": "All the legal bits",
"text-license-page-subtitle": "Last updated on January 20, 2022",
"text-terms-page-title": "Terms and Conditions",
"text-terms-page-subtitle": "Last updated on January 20, 2022",
"text-privacy-page-title": "Privacy Policy",
"text-privacy-page-subtitle": "Last updated on January 20, 2022",
"text-password-page-title": "Change Password",
"text-current-password": "Current Password",
"text-new-password": "New Password",

```

## File Location

shop\src\lib\locals.tsx





```

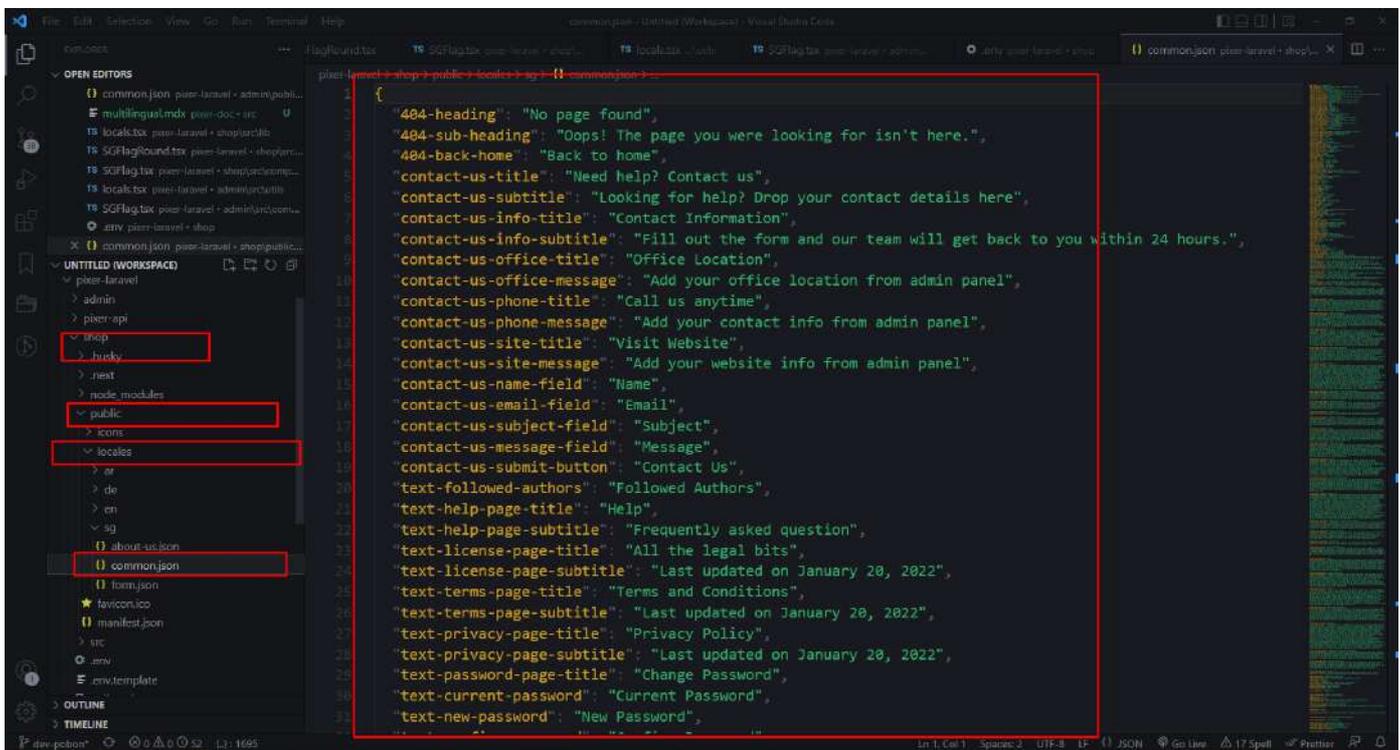
1  {
2    "text-read-more": "Read more",
3    "text-less": "Less",
4    "text-on": "On",
5    "description": "Track Order",
6    "error-heading": "Error code: 404",
7    "nav-menu-track-order": "Track Order",
8    "nav-menu-offer": "Offers",
9    "nav-menu-faq": "FAQ",
10   "nav-menu-contact": "Contact",
11   "user-avatar": "user avatar",
12   "auth-menu-profile": "Contact",
13   "auth-menu-checkout": "Checkout",
14   "auth-menu-my-orders": "My Orders",
15   "auth-menu-logout": "Logout",
16   "join-button": "Join",
17   "change-locale": "Switch to german",
18   "admin-login-title": "Login to admin",
19   "admin-register-title": "Register new account",
20   "billing-address": "Billing Address",
21   "shipping-address": "Shipping Address",
22   "no-order-found": "No Order Found",
23   "order-sub-total": "Sub total",
24   "order-tax": "Tax",
25   "order-delivery-fee": "Delivery fee",
26   "order-discount": "Discount",
27   "order-total": "Total",
28   "signing-out-text": "Signing out...",
29   "sale-history": "Sale History",
30   "january": "January",
31   "february": "February",

```

## Step 4: How to translate static content for shop?

### File Location

```
shop\public\locales\sg\common.json
```



```

1  {
2    "404-heading": "No page found",
3    "404-sub-heading": "Oops! The page you were looking for isn't here.",
4    "404-back-home": "Back to home",
5    "contact-us-title": "Need help? Contact us",
6    "contact-us-subtitle": "Looking for help? Drop your contact details here",
7    "contact-us-info-title": "Contact Information",
8    "contact-us-info-subtitle": "Fill out the form and our team will get back to you within 24 hours.",
9    "contact-us-office-title": "Office Location",
10   "contact-us-office-message": "Add your office location from admin panel",
11   "contact-us-phone-title": "Call us anytime",
12   "contact-us-phone-message": "Add your contact info from admin panel",
13   "contact-us-site-title": "Visit Website",
14   "contact-us-site-message": "Add your website info from admin panel",
15   "contact-us-name-field": "Name",
16   "contact-us-email-field": "Email",
17   "contact-us-subject-field": "Subject",
18   "contact-us-message-field": "Message",
19   "contact-us-submit-button": "Contact Us",
20   "text-followed-authors": "Followed Authors",
21   "text-help-page-title": "Help",
22   "text-help-page-subtitle": "Frequently asked question",
23   "text-license-page-title": "All the legal bits",
24   "text-license-page-subtitle": "Last updated on January 20, 2022",
25   "text-terms-page-title": "Terms and Conditions",
26   "text-terms-page-subtitle": "Last updated on January 20, 2022",
27   "text-privacy-page-title": "Privacy Policy",
28   "text-privacy-page-subtitle": "Last updated on January 20, 2022",
29   "text-password-page-title": "Change Password",
30   "text-current-password": "Current Password",
31   "text-new-password": "New Password",

```

## Step 5: How to translate content?

Now after successfully configuring the Multilingual follow this procedure to translate your existing content to another one.

### Data Type

Now before starting the translation, you've to understand about dependent and non-dependent data at pixer.

At pixer, some data are required to create another data, and it's called dependent data, and the independent data is called non-dependent data.

For example,

To create product types, categories, tags, author, manufacture, attributes, etc are required.

And for coupons or settings, no prior data is required.

Now, to translate a product, you must have to translate types, categories, tags, authors, manufacturers, and attributes at first. Without that, when you translate a product, you can't add your categories or types as those data are not translated yet.

To simplify the process, we made a serial for you. Just follow this serial, and it'll make the process relatively easy.

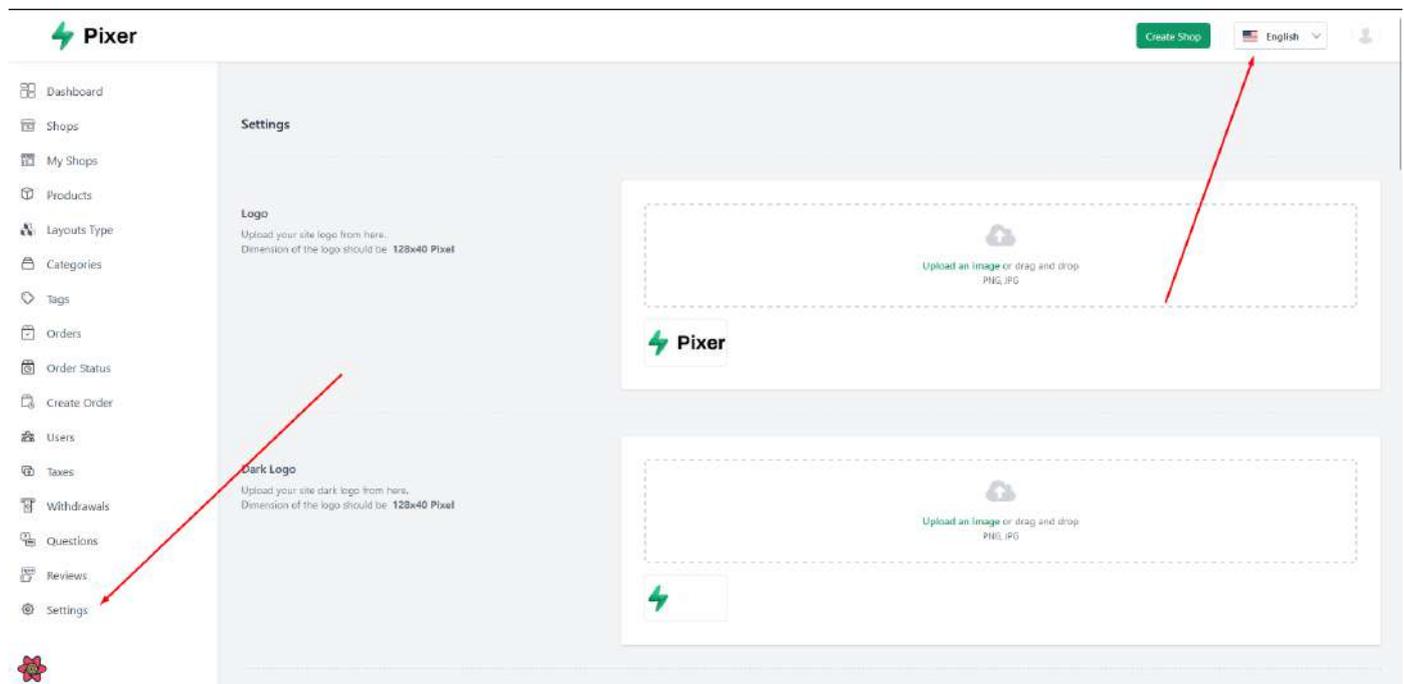
1. Settings
2. Order Status
3. Layouts Type
4. Tags
5. Categories
6. Products

## Translate Settings

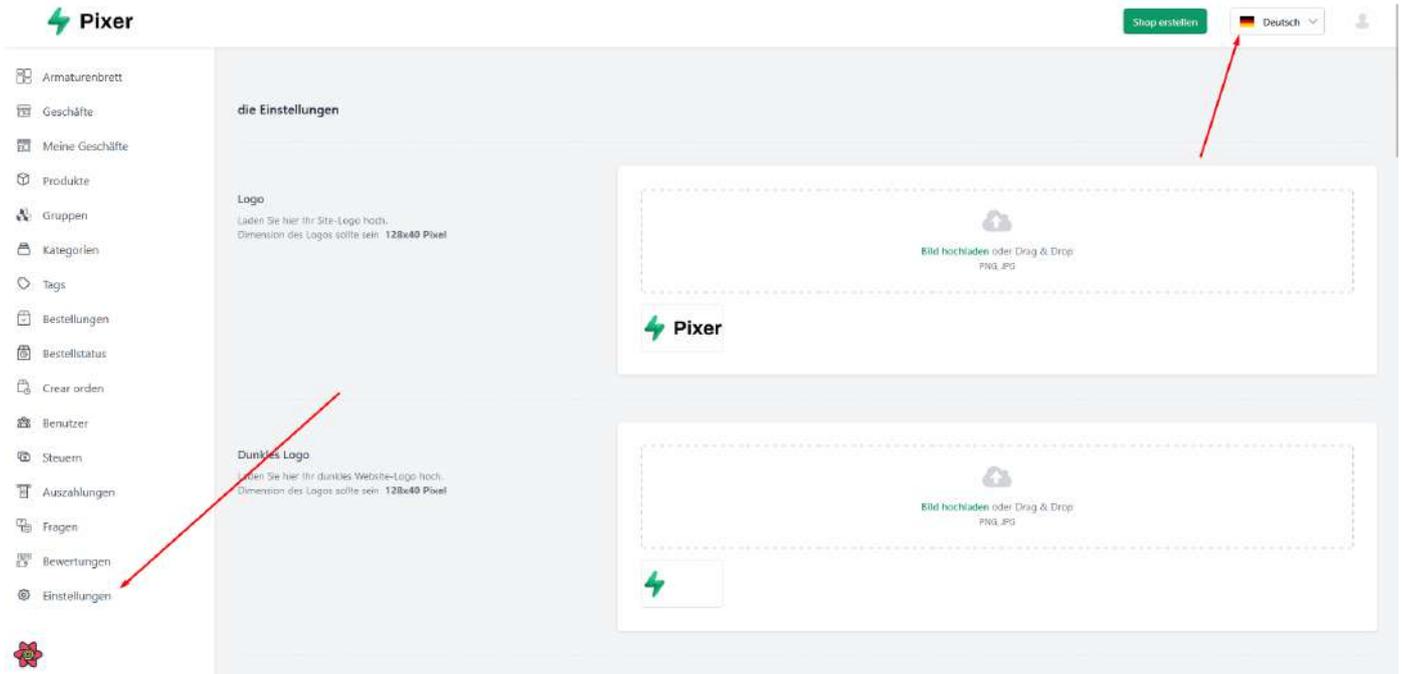
If you want to translate the settings,

just go to your admin -> settings

It'll open default language settings.



Then change the language from the navbar, and the settings page will be redirected,

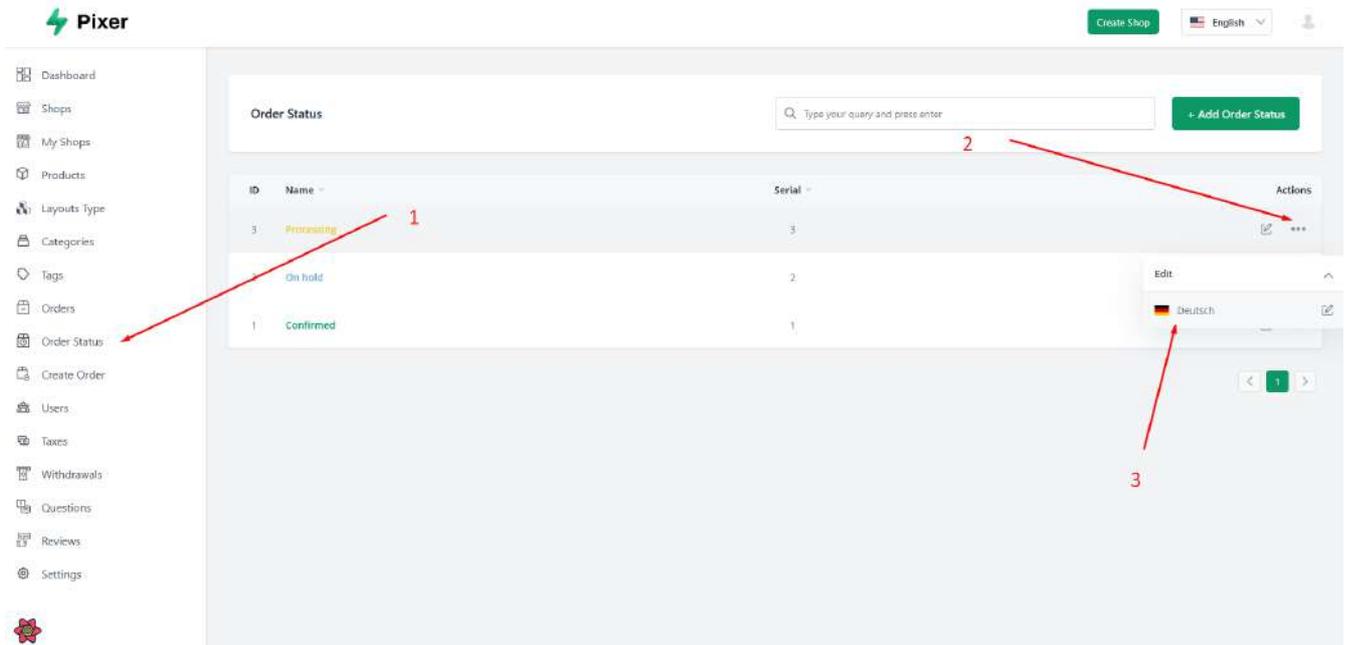


Then update the text of the settings page and click save.

### Translate Order Status

To translate the order status,

1. Go to the order status page from the admin
2. Then click **Three Dot** and Create translate,



3. Then translate the text and click Save.

Similarly, translate all the order status.

Translate Layouts Type, Tags, Categories:

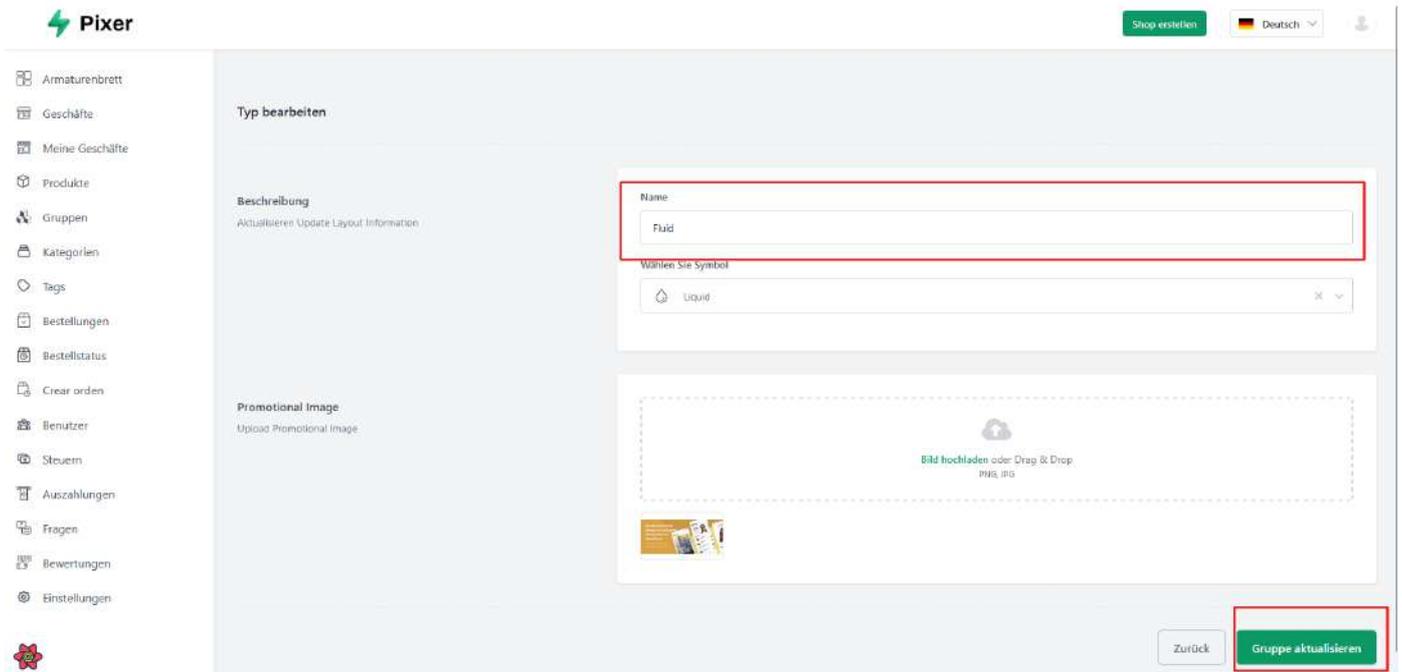
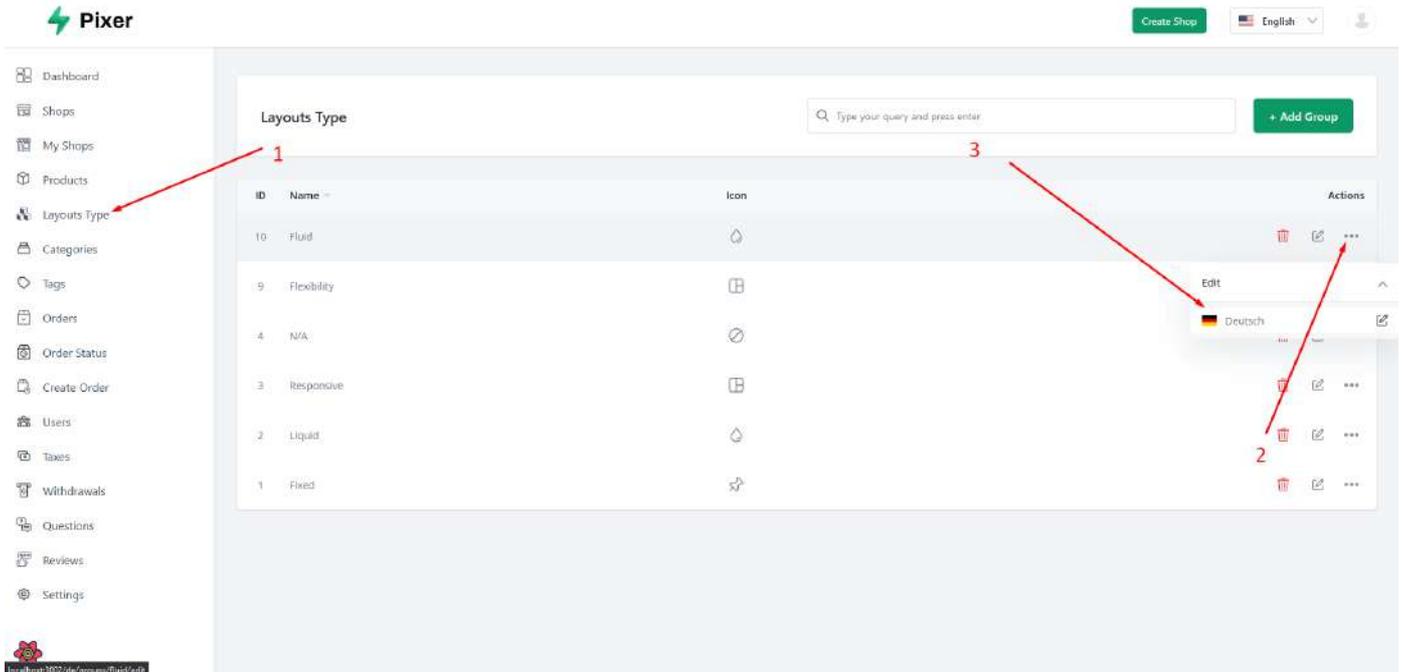
Follow order status procedure to translate **Layouts Type, Tags, and Categories**.

Translate Products

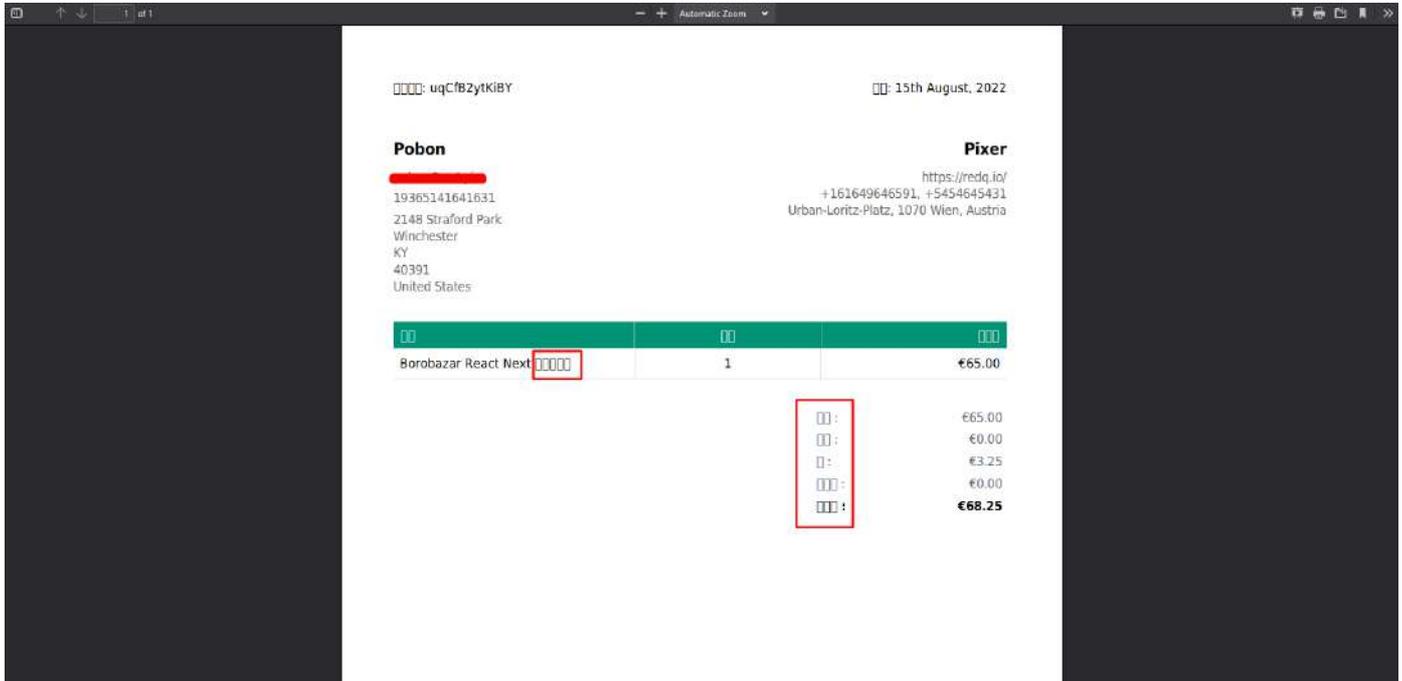
Similarly, just edit three dots and translate the product.

But during the translation, don't change the **SKU**. SKU for a product has to be similar for all languages. We calculate the quantity based on that field.

Translate Layouts Type



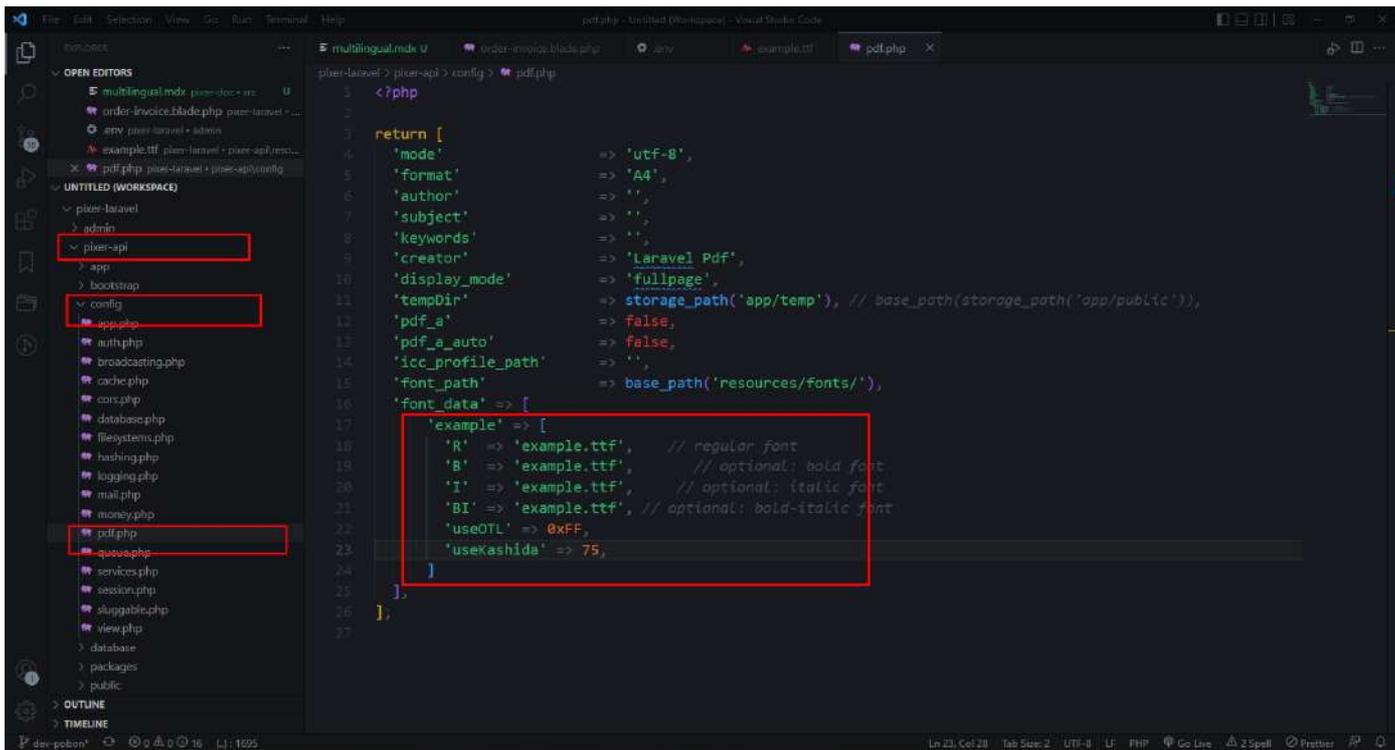
Step 6: How to add custom or solved broken font issue?



File Location

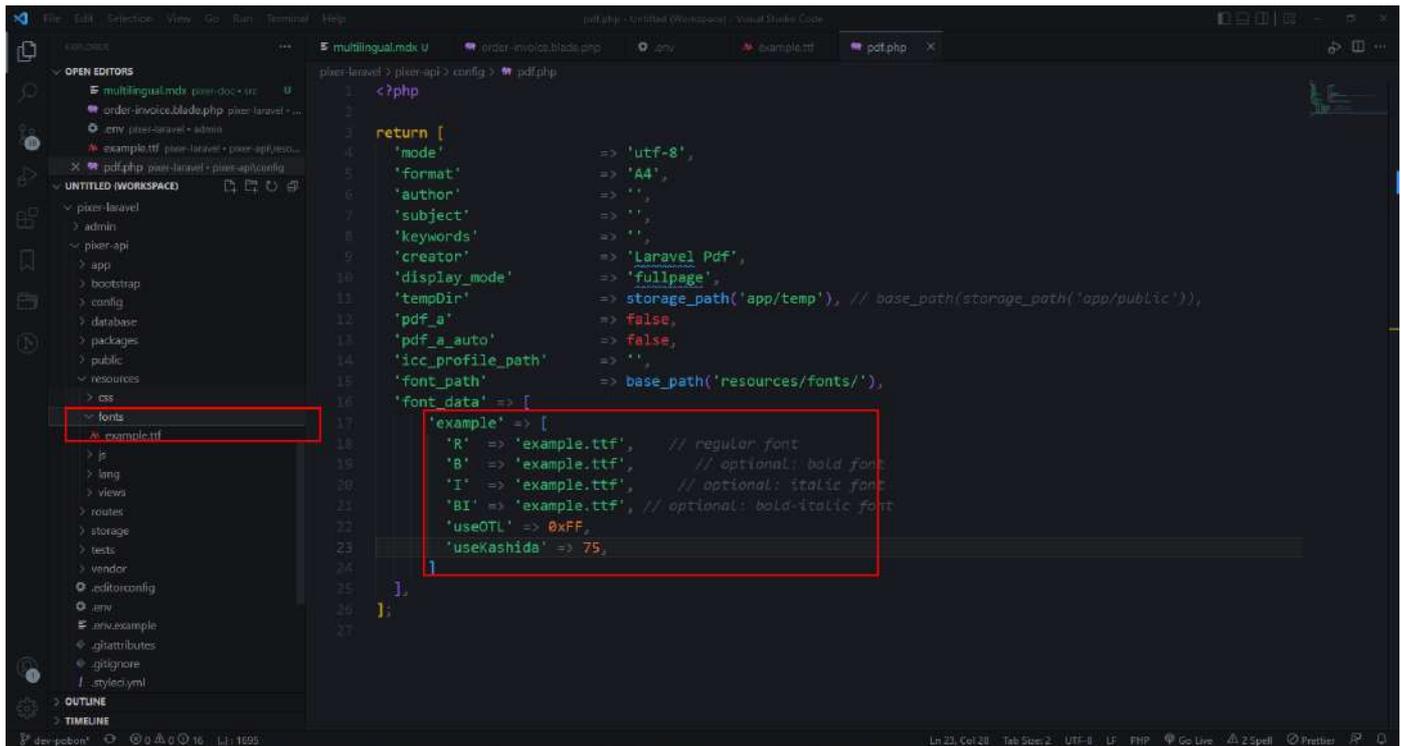
pixier-api\config\pdf.php

```
'font_path' => base_path('resources/fonts/'),
'font_data' => [
  'example' => [
    'R' => 'example.ttf', // regular font
    'B' => 'example.ttf', // optional: bold font
    'I' => 'example.ttf', // optional: italic font
    'BI' => 'example.ttf', // optional: bold-italic font
    'useOTL' => 0xFF,
    'useKashida' => 75,
  ]
],
```



File Location

pixer-api\resources\fonts\



#### File Location

pixer-api\packages\marvel\stubs\resources\views\pdf\order-invoice.blade.php  
 pixer-api\resources\views\pdf\order-invoice.blade.php

```

@if($language === "ex")
    <style type="text/css">
        body {
            font-family: Arial, sans-serif, example;
        }
    </style>
@endif
  
```

```

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Order Invoice</title>
<style type="text/css">
  body {
    font-family: Arial, sans-serif;
  }
</style>
@if($language == "ex")
<style type="text/css">
  body {
    font-family: Arial, sans-serif, example;
  }
</style>
@endif
</head>
<body>
@php
$contactDetails = $settings->options['contactDetails'];
$customer = $order->customer;
$shippingAddress = $order->shipping_address;
$products = $order->products;
$settings = $settings->options;
$authorDetails = $settings['contactDetails'];
$authorLocation = $authorDetails['location'];
$currency = isset($settings['currency']) ? $settings['currency'] : 'USD';
@endphp

```

MpdfException\FontException

Unable to set font "E:\Server\www\wordpress\redq\laravel\pixer-laravel-for-doc\pixer-laravel\pixer-api/resources/fonts/example.ttf" to use OTL as it does not include OTL tables (or at least not a GDEF table).

<http://127.0.0.1:8000/download-invoice/fakers/whXQ5hpa7ck270>

```

Stack trace:
Request: App: User: Context: Debug: Share
Mpdf\Font\MetricGenerator
42: Mpdf\Font\MetricGenerator
41: Mpdf\MpdiF
40: Mpdf\MpdiF
39: Mpdf\MpdiF
38: Mpdf\MpdiF
Mpdf\FontFile::getGDEFtables
E:\Server\www\wordpress\redq\laravel\pixer-laravel-for-doc\pixer-laravel\pixer-api\vendor\mpdf\mpdf\src\FontFile.php:1355
1348: $marksSetTableFormat = $this->read_ushort();
1349: $marksSetCount = $this->read_ushort();
1350: $marksSetOffset = [];
1351: for ($i = 0; $i < $marksSetCount; $i++) {
1352:     $marksSetOffset[] = $this->read_ulong();
1353: }
1354: for ($i = 0; $i < $marksSetCount; $i++) {
1355:     $this->seek($marksSetOffset[$i]);
1356:     $glyphs = $this->getCoverage();
1357:     $this->markGlyphSets[$i] = '' . implode(' ', $glyphs);
1358: }
1359: } else {
1360:     $this->markGlyphSets = [];
1361: }
1362: } also [
1363:     throw new MpdfException\FontException(sprintf('Unable to set font "%s" to use OTL as

```

If you are facing above the issue just replace these code.

#### File Location

pixer-api\config\pdf.php

```

'font_path' => base_path('resources/fonts/'),
'font_data' => [
    'example' => [
        'R' => 'example.ttf', // regular font
        'B' => 'example.ttf', // optional: bold font
        'I' => 'example.ttf', // optional: italic font
        'BI' => 'example.ttf', // optional: bold-italic font
        // 'useOTL' => 0xFF,
        'useKashida' => 75,
    ],

```

```
]
],
```

If you face any issues with Multilingual, or you've any suggestions to improve the Multilingual feature, then please open a ticket at (<https://redqsupport.ticksy.com/>)[<https://redqsupport.ticksy.com/>]

## 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>

## FAQ:

### How to configure Stripe payment gateway?

To configure **Stripe** payment gateway,

1. At first, create a developer account from stripe developer dashboard (<https://dashboard.stripe.com/>)
2. log in to that account
3. Then go to the **Developer** -> **API Keys** section and on that section create an API key. After creating, you'll get one **Secret key** and one **Publishable key**.

The screenshot shows the Stripe Developer Dashboard. On the left, the 'API keys' option is highlighted in the 'Developers' section of the sidebar. The main content area displays a table of API keys. The table has columns for NAME, TOKEN, LAST USED, and CREATED. The 'Publishable key' row is highlighted with a red arrow, and the 'Secret key' row is also highlighted with a red arrow. The 'Secret key' row has a 'Reveal test key' button next to it. Below the main table, there is a 'Restricted keys' section with a '+ Create restricted key' button.

4. Then copy **Secret key** to your `pixer api->.env`

```

41 AWS_ACCESS_KEY_ID=
42 AWS_SECRET_ACCESS_KEY=
43 AWS_DEFAULT_REGION=
44 AWS_BUCKET=
45
46 STRIPE_API_KEY=
47
48 LIGHTHOUSE_CACHE_ENABLE=false
49 MEDIA_DISK=public
50 SHOP_URL=
51 ADMIN_EMAIL=support@example.com
52
53 GOOGLE_CLIENT_ID=
54 GOOGLE_CLIENT_SECRET=
55 GOOGLE_REDIRECT_URI=http://localhost/login/google/callback
56
57 FACEBOOK_CLIENT_ID=
58 FACEBOOK_CLIENT_SECRET=
59 FACEBOOK_REDIRECT_URI=https://localhost/login/facebook/callback
60
61
62 ACTIVE_OTP_PROVIDER=messagebird
63
64 TWILIO_AUTH_TOKEN=
65 TWILIO_ACCOUNT_SID=
66 TWILIO_VERIFICATION_SID=

```

5. And similarly, add Publishable key to your shop -> .env

```

1 # Secrets like the one below should go into ".env.local" instead to avoid pushing
2 # them to a repository.
3 NEXT_PUBLIC_REST_API_ENDPOINT="http://localhost/"
4
5 NEXTAUTH_URL=http://localhost:3003
6 SECRET= # Linux: `openssl rand -hex 32` or go to https://generate-secret.now.sh/32
7
8 GOOGLE_ID=
9 GOOGLE_SECRET=
10
11 NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=
12

```

After configuration, make sure you rebuild your project using this command,

```
yarn build
```

Why am I facing "You may need an appropriate loader to handle this file type" during running shop?

Ans: Please remove `node_modules` from `shop` then `yarn` then run `yarn dev`

I am changing schema files but changes is not working

Your changes might not work because schema is cached. SO you might need to clear schema cache using the below command `php artisan lighthouse:clear-cache`.

## Changing .env files but not getting the changes

Run Below command `php artisan optimize:clear`

## Changing route but not getting the changes.

Run `php artisan optimize:clear` or `php artisan route:clear`

## I have set `STRIPE_API_KEY` in .env but still getting error.

In some cases `STRIPE_API_KEY` value can't read from .env in those cases you have to put the key in the config file directly in `api/packages/marvel/src/Config/laravel-omnipay.php`

- Getting error on forget password email sending Make sure you have run the `php artisan marvel:install` commands successfully and copied the necessary email templates to your resources folder. You can also do it by `php artisan marvel:copy-files` command.

NB: This same issue can occur during order creation.

## Can I use it with my existing laravel?

Yes, you can. Follow the below steps.

- Make sure you are using laravel 8.
- Copy `api/packages` folder from the downloaded files into your laravel root folder.
- Put below code in your laravel `composer.json` file into `require` section.

```
"ignited/laravel-omnipay": "dev-master",
"omnipay/common": "dev-master",
"omnipay/stripe": "dev-master",
"pixier/shop": "dev-master"
```

- Put below code in bottom of your `composer.json`. If you already have an `repositories` section then put code inside `repositories` to your existing `repositories`

```
"repositories": {
  "pixier/shop": {
    "type": "path",
    "url": "packages/marvel"
  },
  "ignited/laravel-omnipay": {
    "type": "path",
    "url": "packages/laravel-omnipay"
  },
  "omnipay/common": {
    "type": "path",
    "url": "packages/omnipay-common"
  },
  "omnipay/stripe": {
    "type": "path",
    "url": "packages/omnipay-stripe"
  }
}
```

- Now run `composer install`
- Copy necessary env variables from `.env.example` to you env file.
- Run `php artisan marvel:install` and follow necessary steps.
- To run server `php artisan serve`
- For image upload to work properly you need to run `php artisan storage:link`.

## Why am I getting `Access denied for user?`

navigate to `api` then run `./vendor/bin/php down -v`. It will delete any of your existing mysql volumes. Now run `./vendor/bin/php up -d` on same directory or run `bash install.sh` on root directory

## Why am I getting permission issue during deployment?

Run below commands for fixing permission issue in your laravel app during deployment

```
sudo chown -R $USER:www-data storage
```

```
sudo chown -R $USER:www-data bootstrap/cache

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

Why am I getting "The GET method is not supported for this route. Supported methods: HEAD"?

Run `php artisan optimize:clear`

How to resolve the **Load More Infinity** loading issue?

If you click on the **Load More** button and the button is spinning continuously, then check API request from the network tab. If it is HTTP and your API is hosted on HTTPS, then open,

```
pixer-api/app/Providers/AppServiceProvider.php
```

And add this code to `boot` method,

```
if (!\App::environment('local')) {
    $this->app['request']->server->set('HTTPS', true);
}
```

I'm trying to upload images, but the images are not displayed on the frontend?

We support two types of file uploads; one is `local`, and another is `AWS S3`.

Both uploading systems follow this procedure,

#### API

At first, add your API URL to your `pixer-api` -> `.env`.

```
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
```

If you're using windows and your API is running with ports like `localhost:8000` or `127.0.0.1:8000`, then make sure you add the domain with port to `APP_URL`. Like this `API_URL=http://localhost:8000`

After that, clear your API cache by using this command from the `pixer-api` folder,

```
php artisan optimize:clear
```

Then link the storage system by using this command,

```
php artisan storage:link
```

If you're using AWS S3, then update `MEDIA_DISK=s3` and add all `AWS_` credentials For AWS s3, make sure your properly setup permission of the bucket with ACL enable by follow this link -> <https://stackoverflow.com/a/70603995/2158023>

Amazon S3 > Buckets > pickbazarlaravel > Edit Object Ownership

## Edit Object Ownership [Info](#)

**Object Ownership**

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

**ACLs disabled (recommended)**  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

**ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

**Object Ownership**

**Bucket owner preferred**  
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

**Object writer**  
The object writer remains the object owner.

[Cancel](#) [Save changes](#)

### Admin

Then add your domain to your `pixer-laravel` -> `admin` -> `next.config.js`

```
images: {
  domains: [
    "YOUR_DOMAIN.COM",
    "via.placeholder.com",
    "res.cloudinary.com",
    "s3.amazonaws.com",
    "18.141.64.26",
    "127.0.0.1",
    "localhost",
    "picsum.photos",
    "pickbazar-sail.test",
    "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "lh3.googleusercontent.com",
  ],
},
```

If you're using [AWS S3](#), then also add your root S3 domain like this,

```
images: {
  domains: [
    "YOUR_DOMAIN.COM",
    "via.placeholder.com",
    "res.cloudinary.com",
    "s3.amazonaws.com",
    "18.141.64.26",
    "127.0.0.1",
    "localhost",
    "picsum.photos",
    "pickbazar-sail.test",
    "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "lh3.googleusercontent.com",
  ],
},
```

You, 7 minutes ago • Uncommitted changes

Like this

### Shop

Similarly, add your domain to your `pixier-laravel -> shop -> next.config.js`

```
images: {
  domains: [
    "YOUR_DOMAIN.COM",
    "via.placeholder.com",
    "res.cloudinary.com",
    "s3.amazonaws.com",
    "18.141.64.26",
    "127.0.0.1",
    "localhost",
    "picsum.photos",
    "pickbazar-sail.test",
    "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "lh3.googleusercontent.com",
  ],
},
```

If you're using [AWS S3](#), then also add your root S3 domain like this,

```
images: {
  domains: [
    "YOUR_DOMAIN.COM",
    "via.placeholder.com",
    "res.cloudinary.com",
    "s3.amazonaws.com",
    "18.141.64.26",
    "127.0.0.1",
    "localhost",
    "picsum.photos",
    "pickbazar-sail.test",
    "pickbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "chawkbazarlaravel.s3.ap-southeast-1.amazonaws.com",
    "lh3.googleusercontent.com",
  ],
},
```

You, 7 minutes ago • Uncommitted changes

Like this

### Build Frontend,

If you run in dev mode, then re-run will affect your update. But if you're running as a production mode, then you've to rebuild the project and re-run again. Otherwise, image upload will not work.

<https://pixier-doc.vercel.app/vps-server#build-project>

After building the project, replace your server shop and admin folder using this rebuild folder and re-run the app using this command,

```
pm2 restart all
```

After that, make sure you rebuild the frontend part and upload it to your server again.

## How to resolve the 502 Bad Gateway error for frontend/admin?

There can be several reasons that throw 502 errors for the frontend in the production server.

Reason 1:

On your **Nginx file**, a specific port is set for shop or the admin, so when you run the script as a PM2 instance, the script has to be run that specific port; otherwise, your domain/subdomain will throw a 502 error.

**PORT:**

```
3000 -> Shop Rest  
3001 -> Admin Rest
```

To check that at the first stop the PM2 instance by this command,

```
pm2 stop 'all'
```

then go to the **pixer** folder from your server and try to run the script using yarn and check which port it is running,

From **admin directory**

```
yarn start
```

From **shop directory**

```
yarn start
```

And, check if the port matched with the Nginx port or not? If not matched, then change the port to **Nginx config**.

Reason 2:

To check that at the first stop the PM2 instance by this command,

```
pm2 stop 'all'
```

then go to the **pixer** folder from your server and try to run the script using yarn and check which port it is running,

From **admin directory**

```
yarn start
```

From **shop directory**

```
yarn shop
```

With this command, it'll give you an error that no build file is found. To resolve that, rebuild your project and then rerun the scripts.

<https://pixer-doc.vercel.app/vps-server#frontend-project-build>

## How to resolve javascript heap out of memory issue?

If you're using cPanel based server, then you maybe get an error like "call\_and\_retry\_last allocation failed - javascript heap out of memory".

To resolve that issue, you've to add a limit of the memory bound,

Please check this,

<https://stackoverflow.com/questions/38558989/node-js-heap-out-of-memory>

<https://support.snyk.io/hc/en-us/articles/360002046418-JavaScript-heap-out-of-memory>

<https://blog.openreplay.com/javascript-heap-out-of-memory-error>

## Image upload throw Internal Server Error; how to resolve that?

For uploading, the php8.1-gd library is required. So make sure you install the php8.1-gd library on your server.

## How to resolve docker `invalid reference format: repository name must be lowercase?`

To resolve that issue, you've to move the `laravel-ecommerce` folder from `Pixer Laravel - React Next REST Ecommerce` to a generic folder where you store your code. After that, use this command from your `laravel-ecommerce` folder,

```
chmod +x install.sh
bash install.sh
```

## Sometimes my server shutdown or halts automatically. After restart, it works fine again; how to resolve that?

In general, this happens mainly when your upstream server fails to handle its processing power when there is lots of traffic or requires lots of operation. Please upgrade your server configuration to double to check the issue. Please check this video,

<https://www.youtube.com/watch?v=4plqAz5bzX0>

If that doesn't solve the issue, then you've to redeploy again with a new ec2 instance.

## S3 uploading doesn't work after adding credentials to `.env`; how to resolve that?

For AWS s3, make sure your properly setup permission of the bucket with ACL enable by follow this link

<https://stackoverflow.com/a/70603995/2158023>

## How to rebuild the project?

When you upload the code to the server as a production, NextJS fetch its code from the build file. So after deployed, when you update or change code on scripts, the updated scripts don't work at the front end until you rebuild the project again. NextJs works this way.

Now there are two ways you can follow to rebuild the project,

If your server has more than 2+ CPU core and 6GB+ of memory, then you can directly edit code at your server then rebuild the project from the server directly using this command,

For `shop`,

Go to `/var/www/pixer-laravel/shop`

```
yarn build
```

And for `admin`,

Go to `/var/www/pixer-laravel/admin`

```
yarn build
```

After completing the build, restart the PM2 process using this command,

```
pm2 restart 'all'
```

But suppose your server doesn't have that processing power, and you try to build your scripts directly from the server. In that case, your server will be shut down during the build, and you've to restart the server.

In that case, you can rebuild the scripts at your local computer and upload them to your server. If you follow our [VPS Server Deployment](#) guide, then you already know the process. To build the scripts on your pc, follow this procedure,

At first, download the `pixer-laravel` folder from the server to your computer, and customize the code at your computer.

Then to build shop, go to `shop` folder and use this command,

To install all the packages,

```
yarn
```

Then build the shop,

```
yarn build
```

Then to build the admin, go to `admin` folder and use this command,

To install all the packages,

```
yarn
```

```
yarn build
```

Then delete the shop and admin folder from the server and upload your build shop and admin folder to the server at the exact same location and use this command to restart the PM2,

```
pm2 restart 'all'
```

## How to increase upload size?

The API supports `10MB` of upload limit for a single file by default. If your upload file is more than `10MB`, then please follow this,

At first, login to the server using ssh, and then,

Edit,

```
/etc/php/8.1/fpm/php.ini
```

Then update,

```
upload_max_filesize = 256M
post_max_size = 256M
max_execution_time = 300
```

Then restart the PHP,

```
sudo service php8.1-fpm restart
```

After that update,

```
pixer-api/packages/marvel/config/media-library.php
```

```
'max_file_size' => 1024 * 1024 * 256,
```

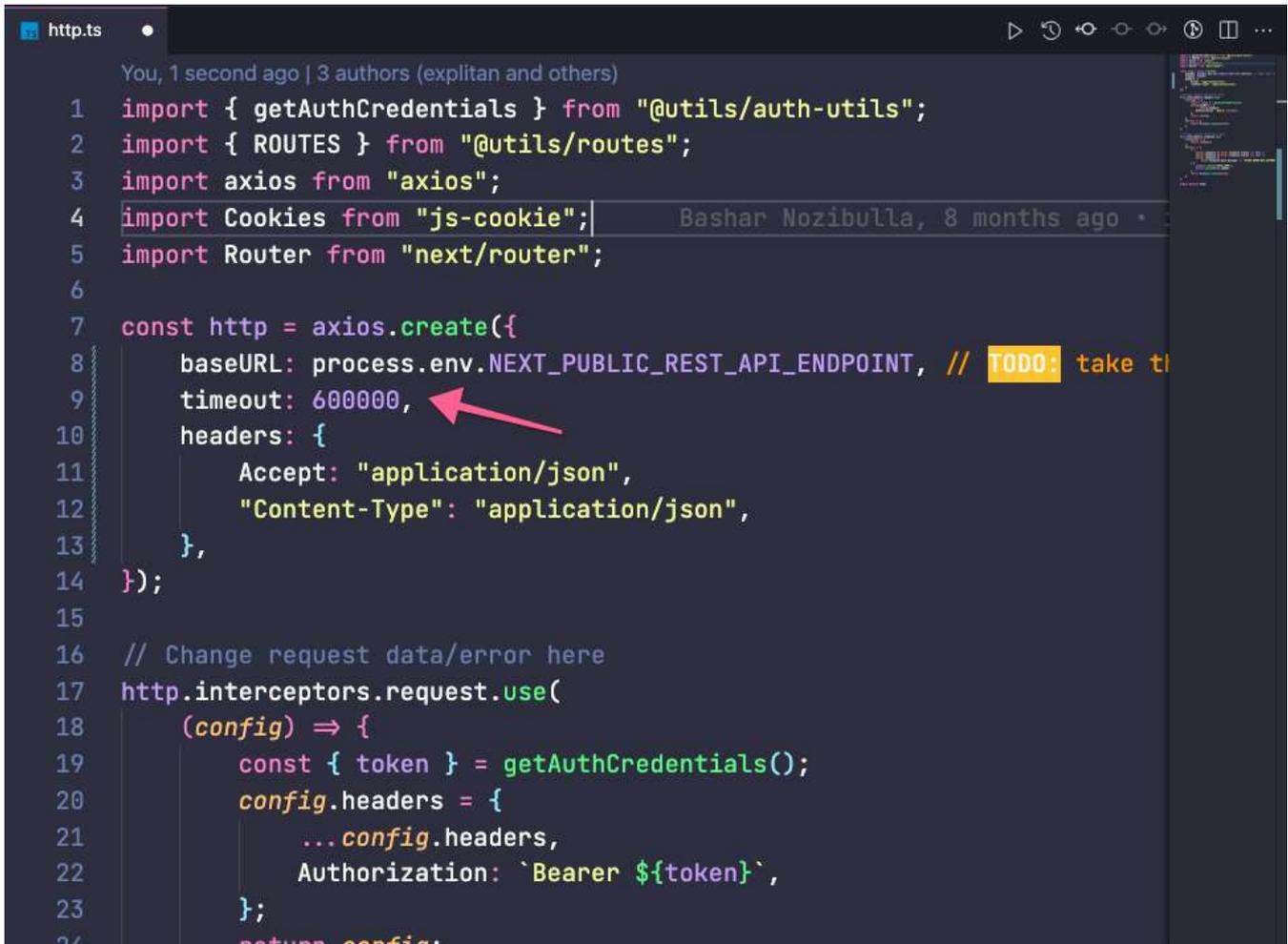
```
...
1  <?php
2
3  return [
4
5      /*
6       * The disk on which to store added files and derived images by default.
7       * one or more of the disks you've configured in config/filesystems
8       */
9      'disk_name' => env('MEDIA_DISK', config('shop.media_disk')),
10
11     /*
12      * The maximum file size of an item in bytes.
13      * Adding a larger file will result in an exception.
14      */
15     'max_file_size' => 1024 * 1024 * 256,
16
17     /*
18      * This queue will be used to generate derived and responsive images.
19      * Leave empty to use the default queue.
20      */
21     'queue_name' => '',
22
23     /*
```

After that, run this command from the `pixier-api` folder,

```
php artisan optimize:clear
```

And for `admin`, increase `timeout` from,

```
admin/src/utils/api/http.ts
```



```
http.ts
You, 1 second ago | 3 authors (explitan and others)
1 import { getAuthCredentials } from "@utils/auth-utils";
2 import { ROUTES } from "@utils/routes";
3 import axios from "axios";
4 import Cookies from "js-cookie";
5 import Router from "next/router";
6
7 const http = axios.create({
8   baseURL: process.env.NEXT_PUBLIC_REST_API_ENDPOINT, // TODO: take ti
9   timeout: 600000,
10  headers: {
11    Accept: "application/json",
12    "Content-Type": "application/json",
13  },
14 });
15
16 // Change request data/error here
17 http.interceptors.request.use(
18   (config) => {
19     const { token } = getAuthCredentials();
20     config.headers = {
21       ...config.headers,
22       Authorization: `Bearer ${token}`,
23     };
24     return config;
```

```
timeout: 600000
```

After that [Rebuild Your Project](#) and restart again.

How to resolve `docker: invalid reference format: repository name must be lowercase.?`

Please open your terminal and use this command,

```
brew services stop mysql
```

It'll stop any MySQL instance that is installed using brew.

Then use this command to check is there MySQL running or not,

```
mysql -v
```

If no MySQL is running then, go to,

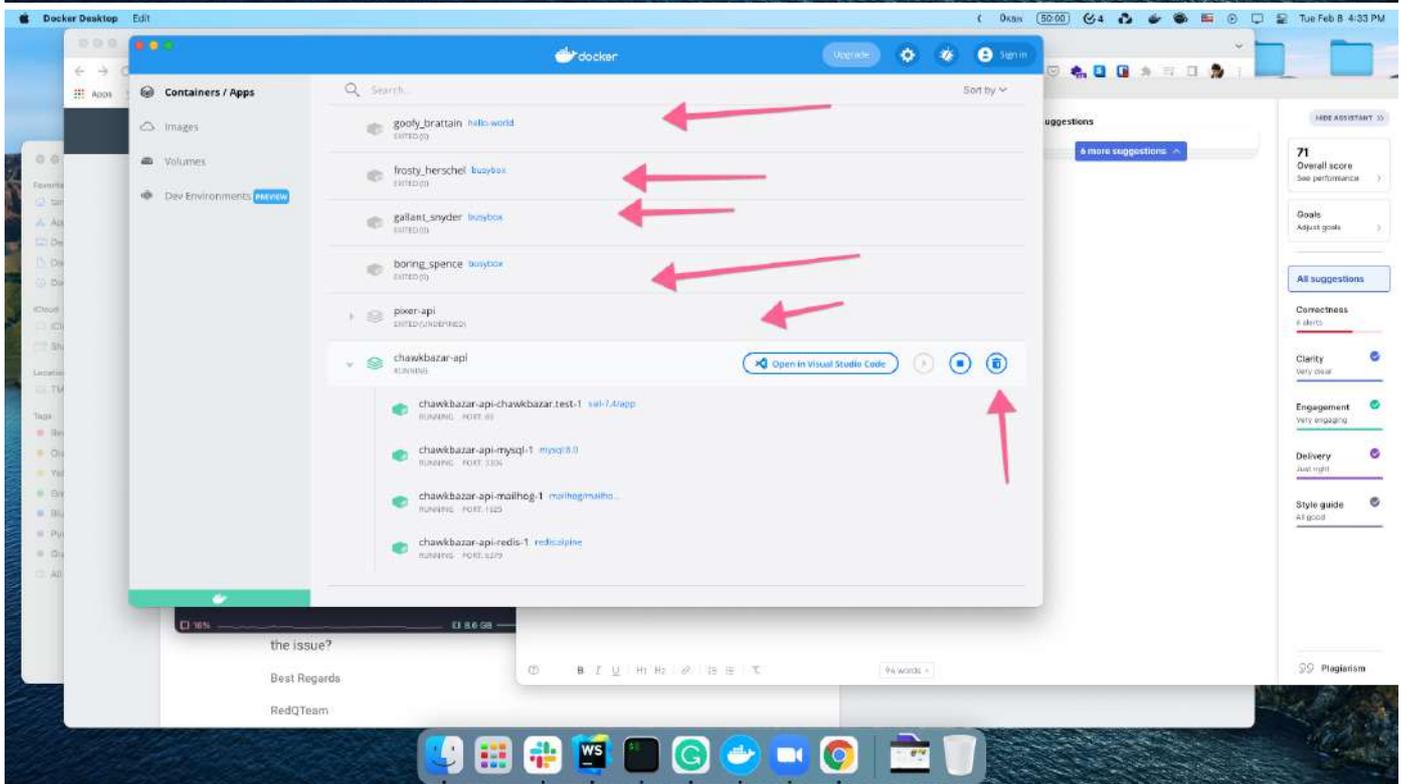
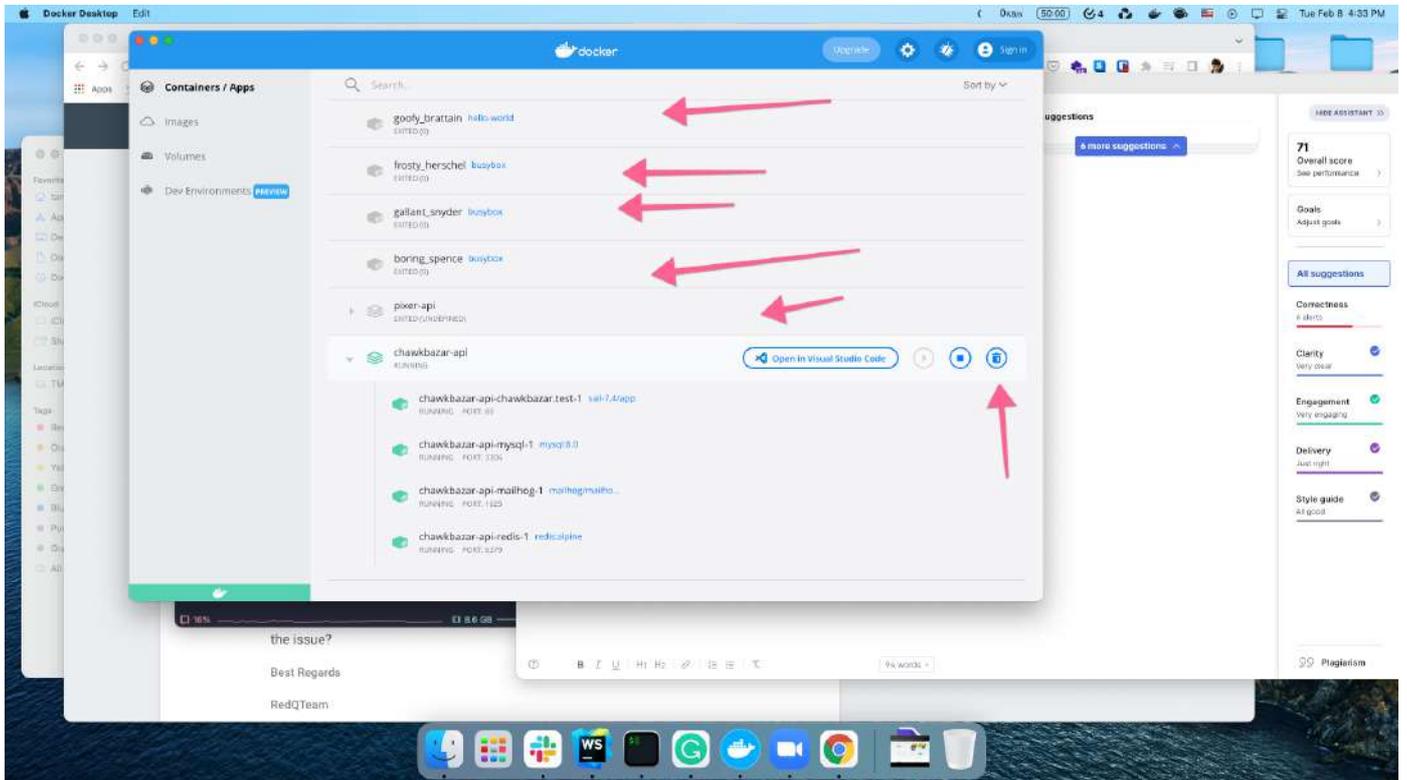
```
pixer-laravel -> pixer-api
```

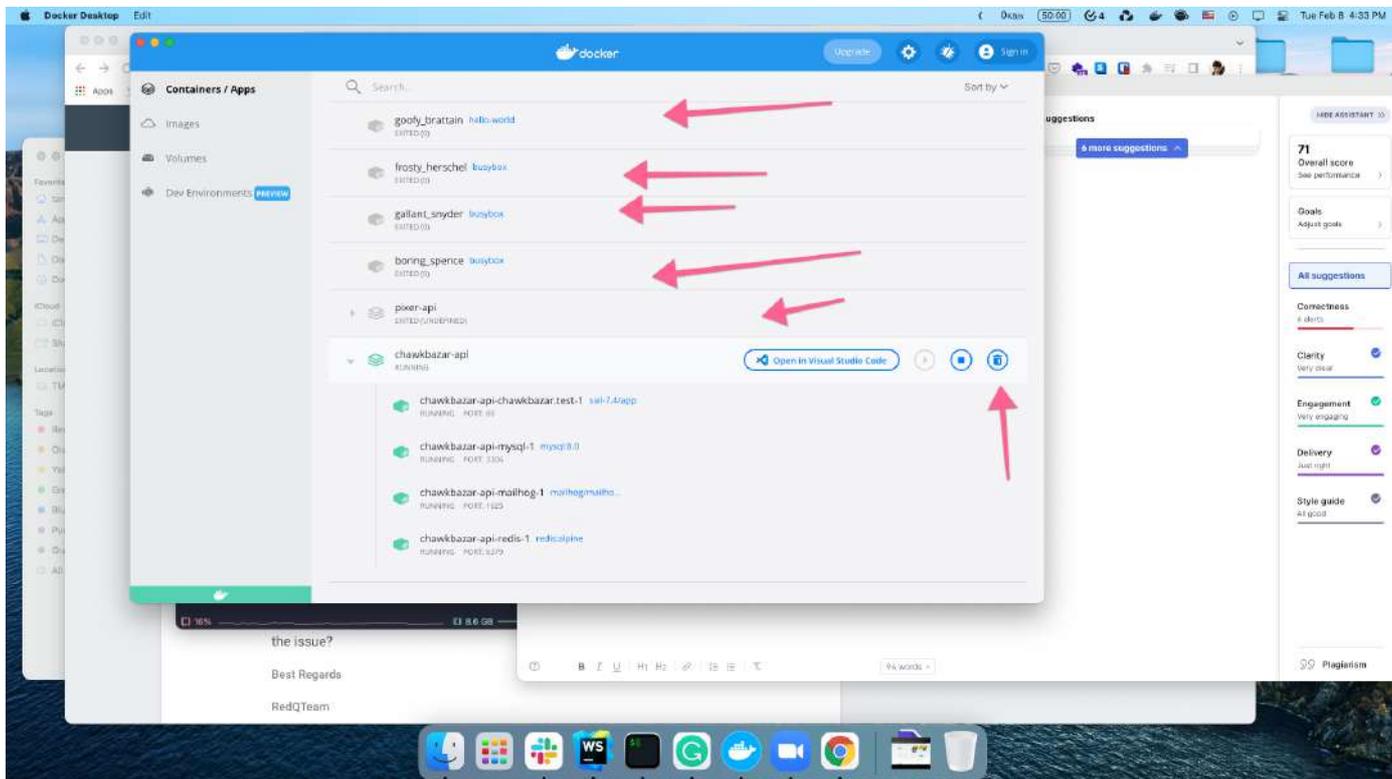
and use this command,

```
./vendor/bin/sail down -v
```

After that,

Open docker dashboard and delete all the existing container and images,





And after that remove,

```
pixier-laravel -> pixier-api -> .env
```

And then go to your root `pixier-laravel` folder and use this command again,

```
chmod +x install.sh
bash install.sh
```

## What type of stripe payment support in pixier?

Pixier only supports a card-based payment system, so if your country doesn't support a card-based payment system, then you've to customize the script to implement another payment system in stripe.

## How to remove the existing payment gateway?

You can remove the Stripe Payment from the available payment array,

```
shop/src/components/checkout/payment/payment-grid.tsx
```

## How do I log in to the admin panel as the main administrator?

When you install the API using `php artisan marvel:install` or using `bash install.sh(docker)`, you'll get a prompt to create an admin account, and you've to create admin credentials that time. And then, you can use that credentials for the main admin account.

## How to upgrade the existing deployed laravel 8 server to laravel 9?

At first, remove all the existing `php 7.4` and its extensions by using this command,

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

Then remove the composer,

```
sudo rm /usr/bin/composer
```

Then delete the `vendor` folder from the `pixier-laravel` -> `api` folder.

```
cd /var/www/pixier-laravel/api

sudo chown -R $USER:www-data storage
sudo chown -R $USER:www-data bootstrap/cache

rm vendor -rf
```

Then install PHP 8, and its extensions,

```
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

```
sudo apt install php8.0-fpm php8.0-mysql
```

```
sudo apt install php8.0-mbstring php8.0-xml php8.0-bcmath php8.0-simplexml php8.0-intl php8.0-mbstring php8.0-gd php8.0-curl
php8.0-zip
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a450292b6cfd1d64d92e6563bbde02') { echo 'Installer
verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

```
sudo mv composer.phar /usr/bin/composer
```

Then update 7.4 fpm to 8.0 fpm from,

```
/etc/nginx/sites-enabled/pixier
```

```
# For API
location /backend {
    alias /var/www/pixer-laravel/pixer-api/public;
    try_files $uri $uri/ @backend;
    location ~ /\.php$ {
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $request_filename;
        fastcgi_pass unix:/run/php/php8.0-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;
}

error_page 404 /index.php;

location ~ /\.php$ {
    fastcgi_pass unix:/var/run/php/php8.0-fpm.sock;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
```

Then install composer packages,

```
cd /var/www/pixer-laravel/api
```

```
composer install
```

```
php artisan optimize:clear
```

```
php artisan marvel:install
```

php artisan marvel:install will remove all of your existing data. Ensure you export or backup your data before using that command.

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

## Support

For support, Please open a ticket in the support forum

<https://redqsupport.ticksy.com/>

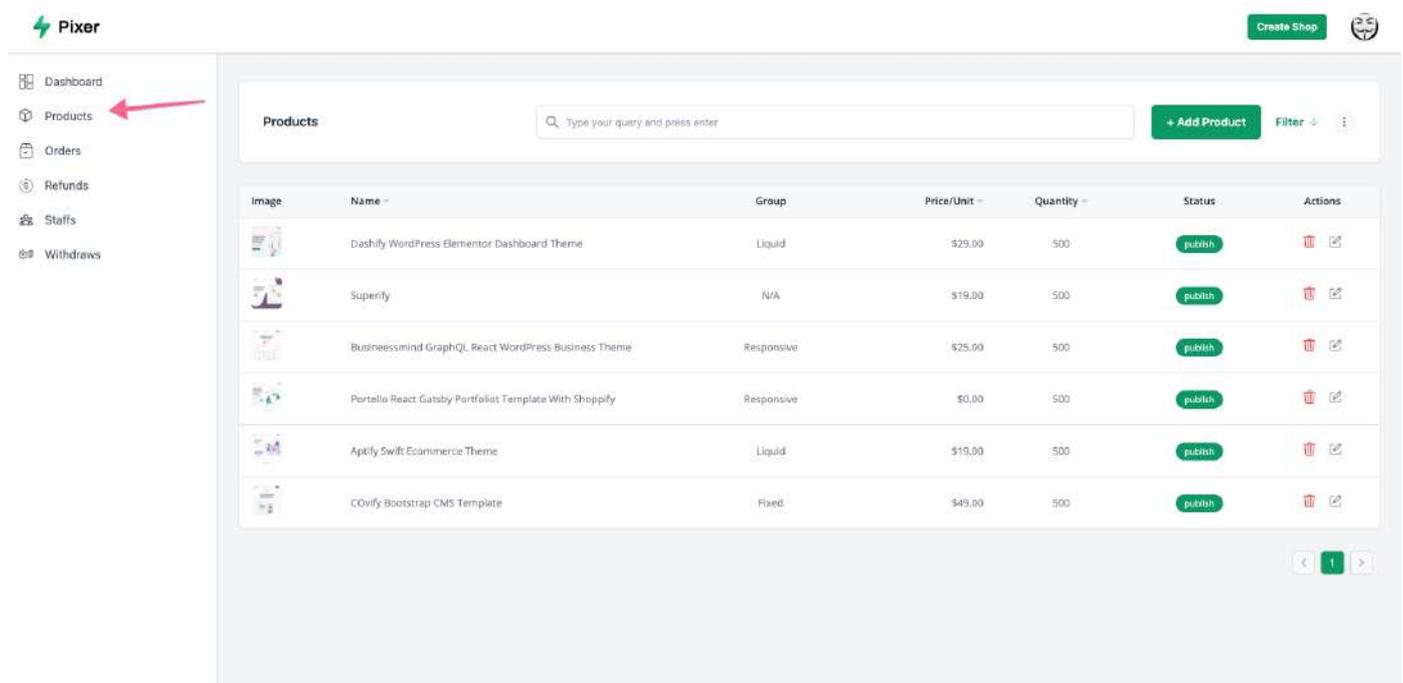
There are some basic requirements for our application so that support team can help you quick such as,

1. Asking queries regarding feature that is already implemented in the application
2. Following recommended configuration, environments & server which you need to met first before you proceed with installation, deployment in your server to receive support.
3. Support query need to be within Envato item support policy. ([https://themeforest.net/page/item\\_support\\_policy](https://themeforest.net/page/item_support_policy))
4. You should maintain only one support ticket at a time. Creating multiple ticket can cause unexpected delays.
5. Ticket should provide as much details as possible related to the issue such as screenshot, video explanation, access, how to reproduce, environments, if any changes or customisations are made etc

During working days our ticket response can take 24 hours to 48 hours depending on the volume of tickets pending prior to your ticket . We follow Envato Item support policy [https://themeforest.net/page/item\\_support\\_policy](https://themeforest.net/page/item_support_policy) to provide standard support for our items

## Export Import

Export Import works under a shop. So, You have to go to the [shop page](#) -> [products menu](#) for product or [shop page](#) -> [attributes menu](#) to import or export attributes.



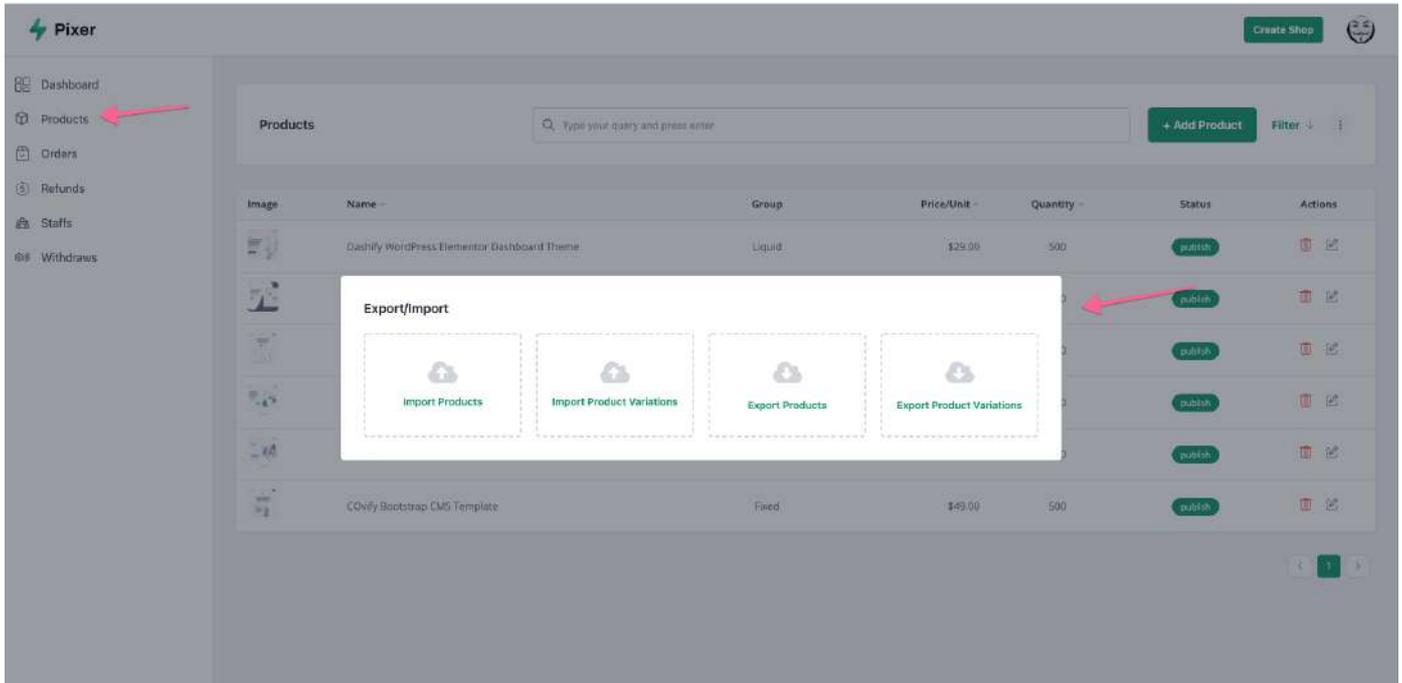
The screenshot shows the Pixier dashboard interface. On the left sidebar, the 'Products' menu item is highlighted with a red arrow. The main content area displays a 'Products' table with the following data:

Image	Name	Group	Price/Unit	Quantity	Status	Actions
	Dashify WordPress Elementor Dashboard Theme	Liquid	\$29.00	500	publish	
	Superfly	N/A	\$19.00	500	publish	
	Businessmind GraphQL React WordPress Business Theme	Responsive	\$25.00	500	publish	
	Portello React Gatsby Portfolio Template With Shopify	Responsive	\$0.00	500	publish	
	Aptify Swift Ecommerce Theme	Liquid	\$19.00	500	publish	
	COvify Bootstrap CMS Template	Fixed	\$49.00	500	publish	

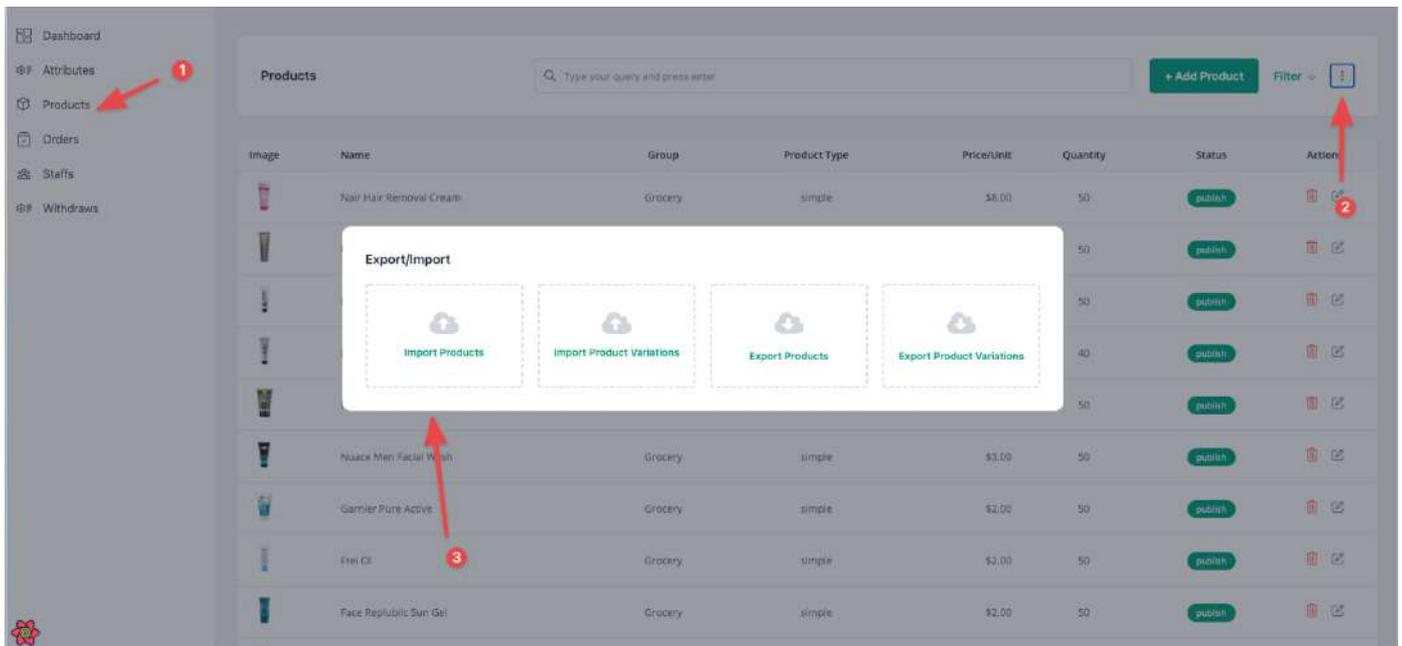
Export Import work different way for simple and variable product

### Simple Product

For simple products, you've to export only products data. To do that, go to [Your Shop](#) -> [products](#). Then click on [three dots](#) and export the simple products.



Similarly, for import *csv*, go to the same option and import exported *csv*

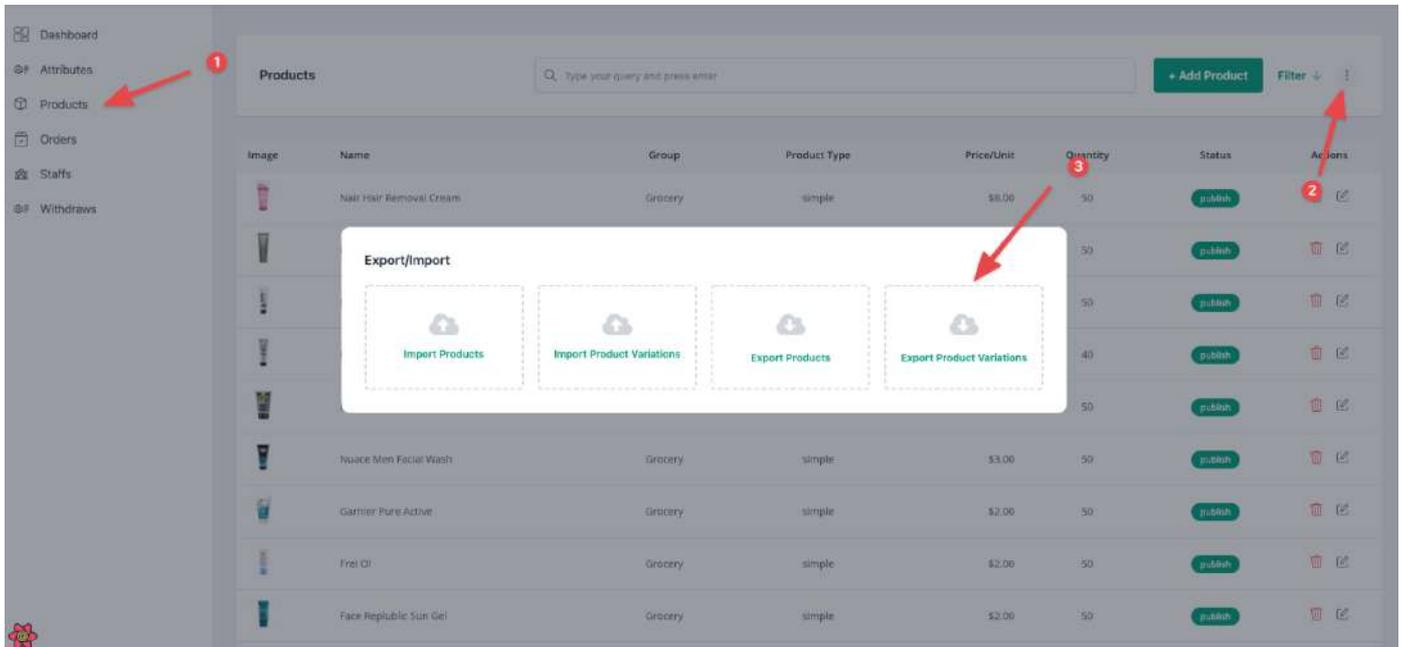
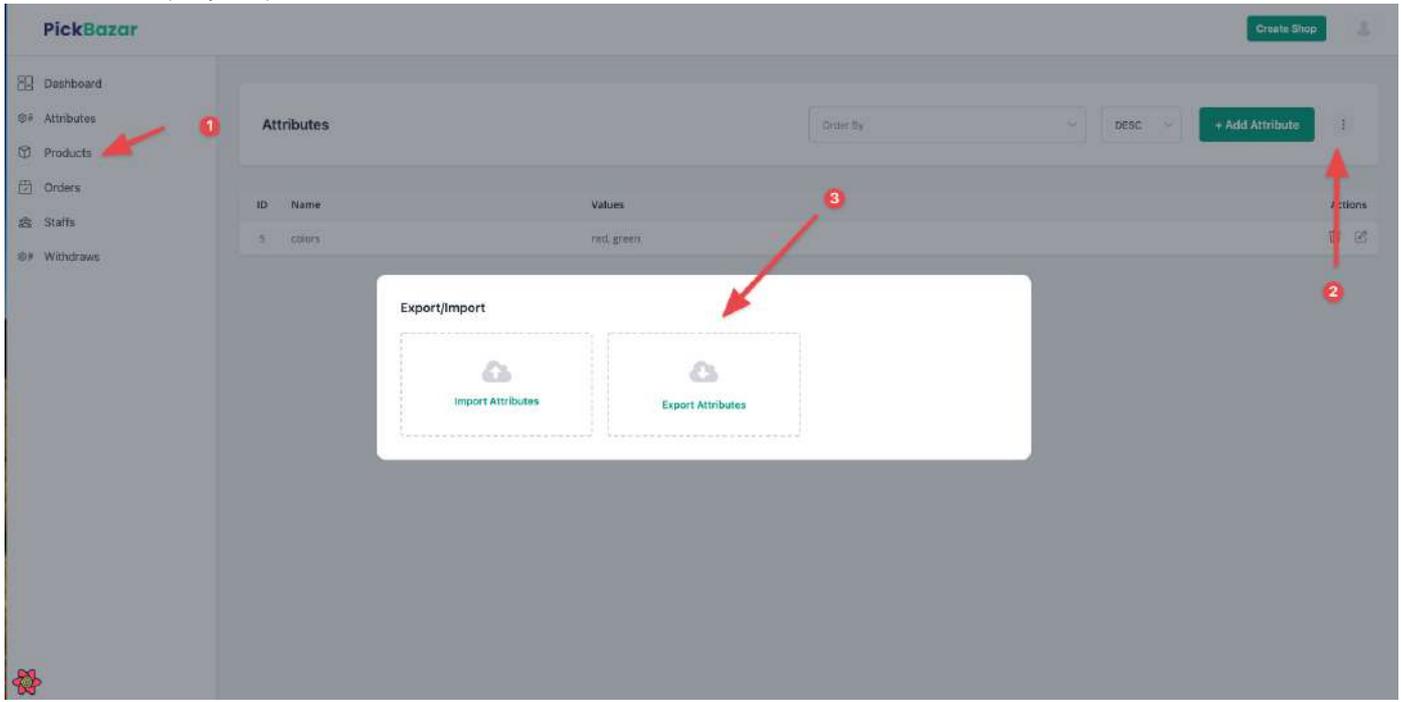


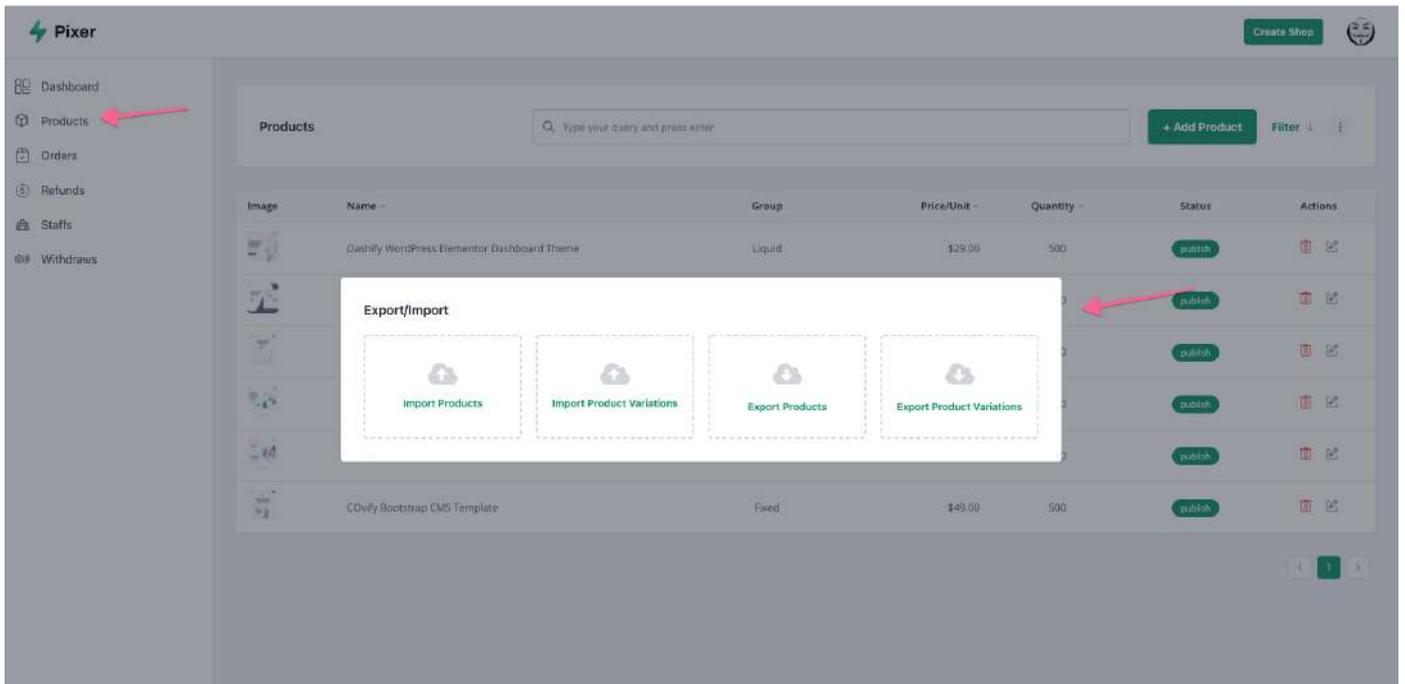
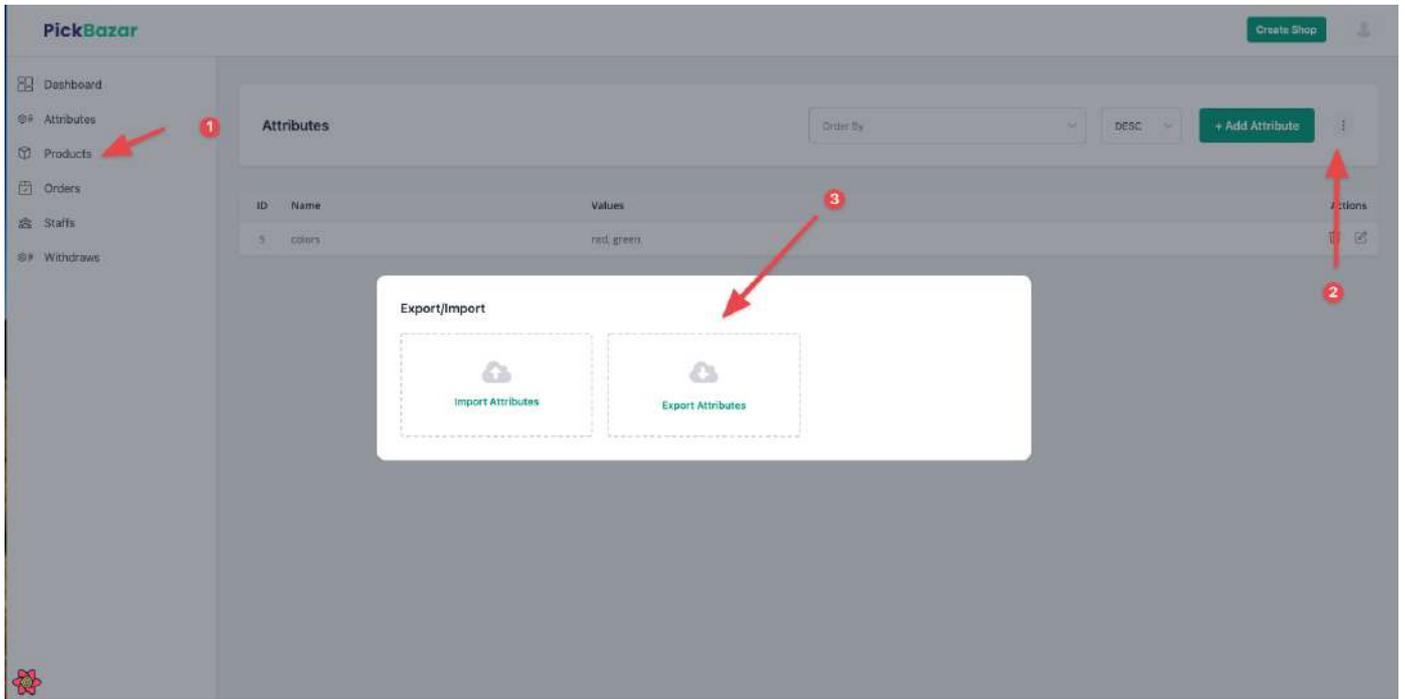
### variable products

For variable products, you will need three different *csv*.

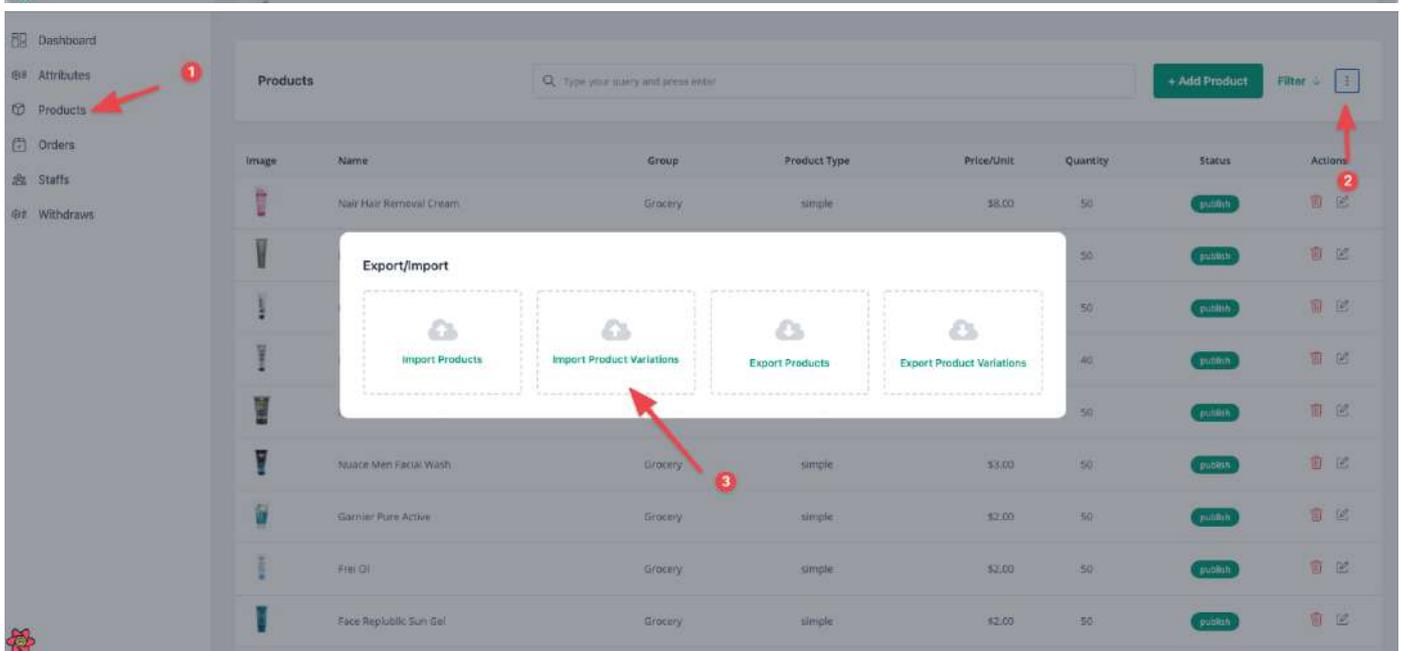
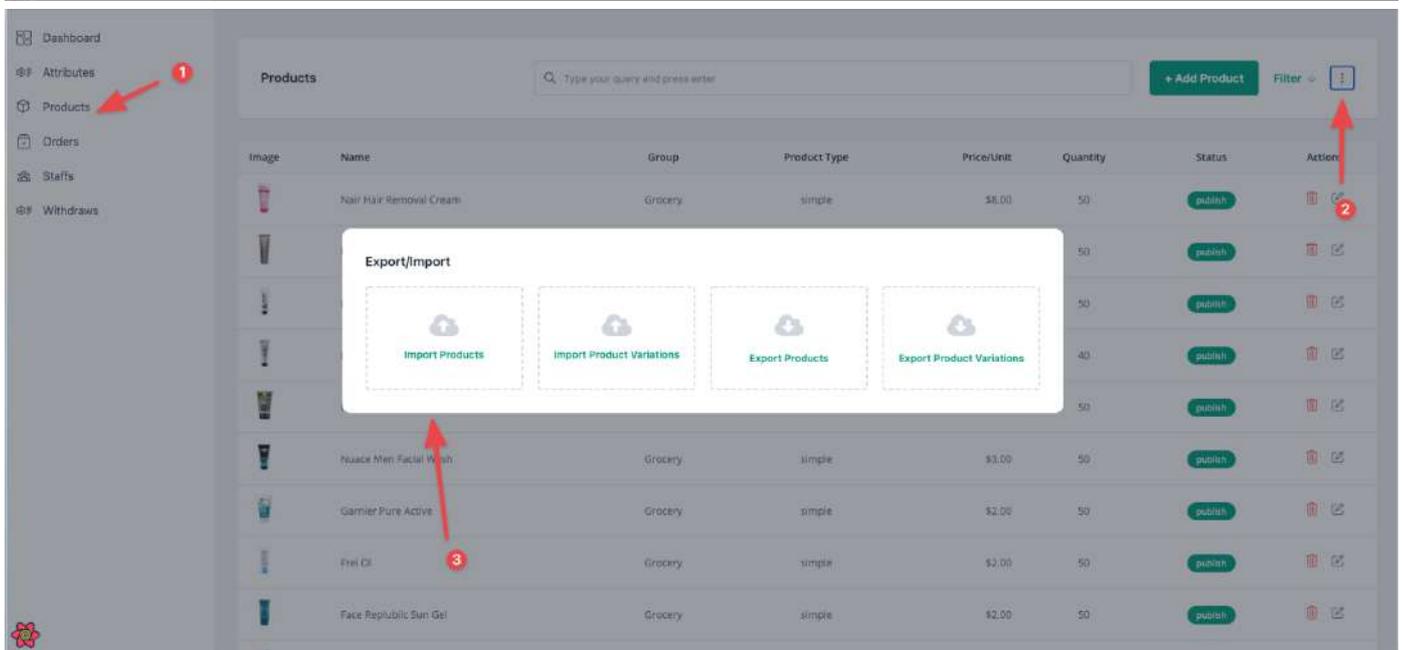
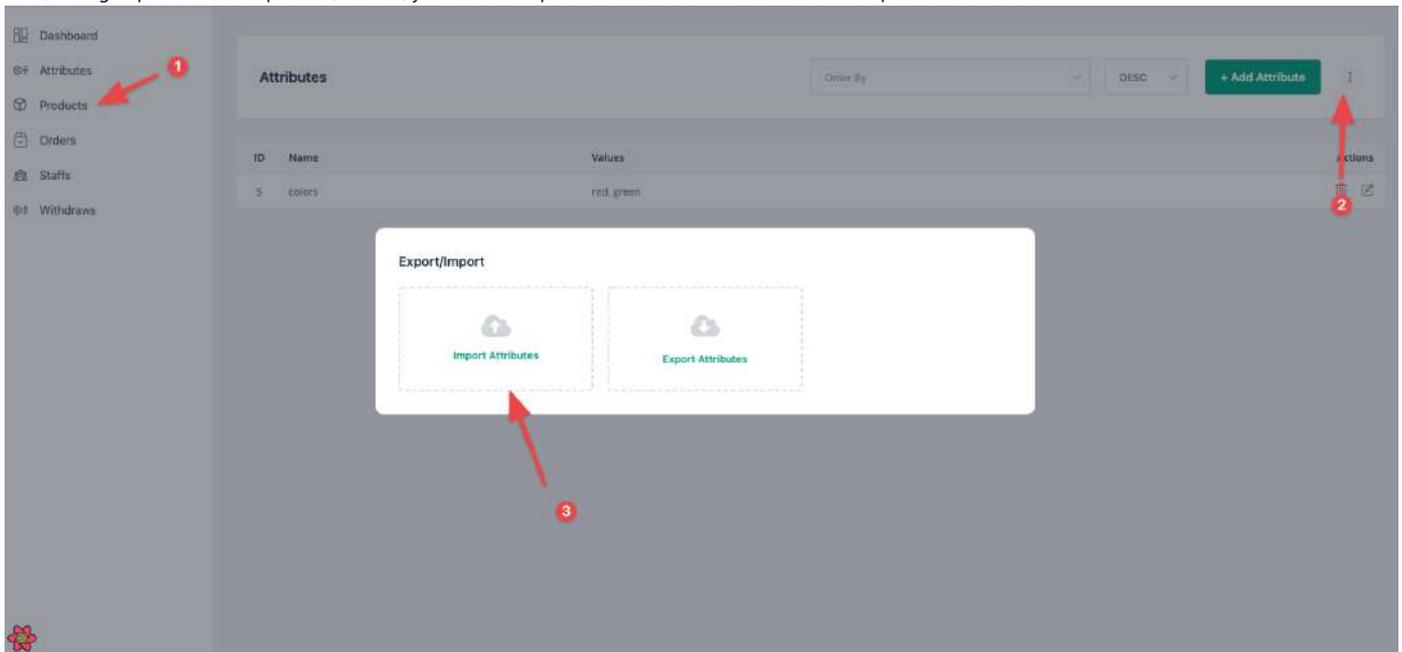
- Attributes *csv*
- products *csv*
- variation options *csv*

Make sure to re-import you export these three csv.





Now, during import variations product, At First, you have to import Attributes then Products and Then products variation.



NB: Here, type\_id is the group id like Grocery or clothing id. You can get type info from [api\\_url/types url](#);

# Withdrawals

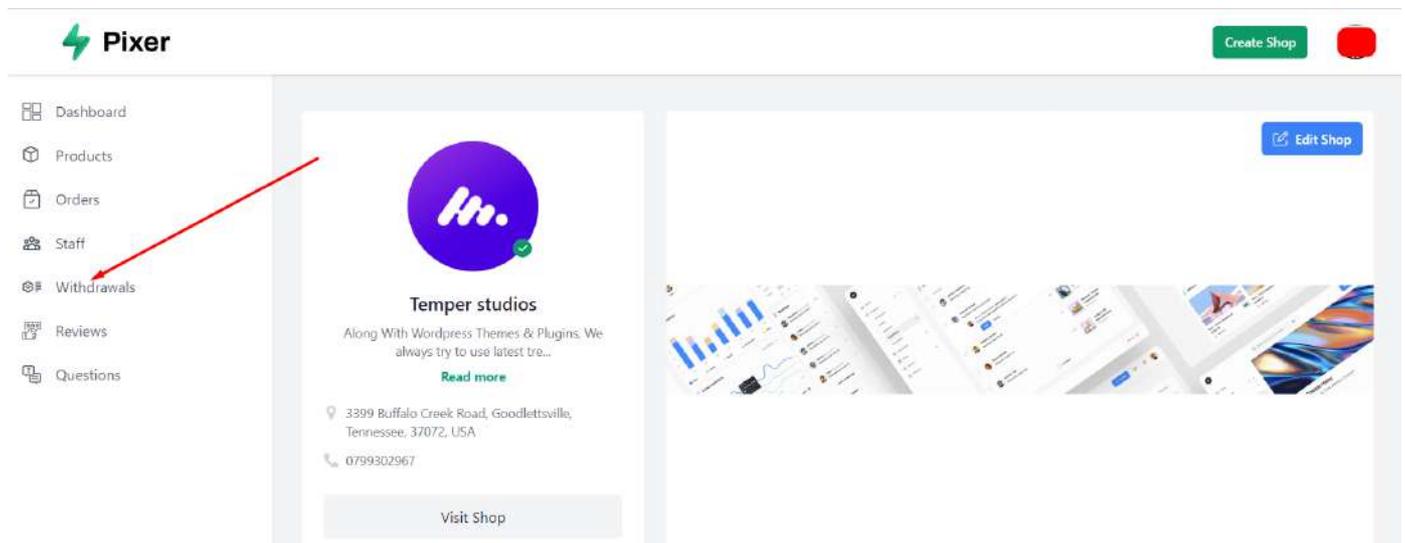
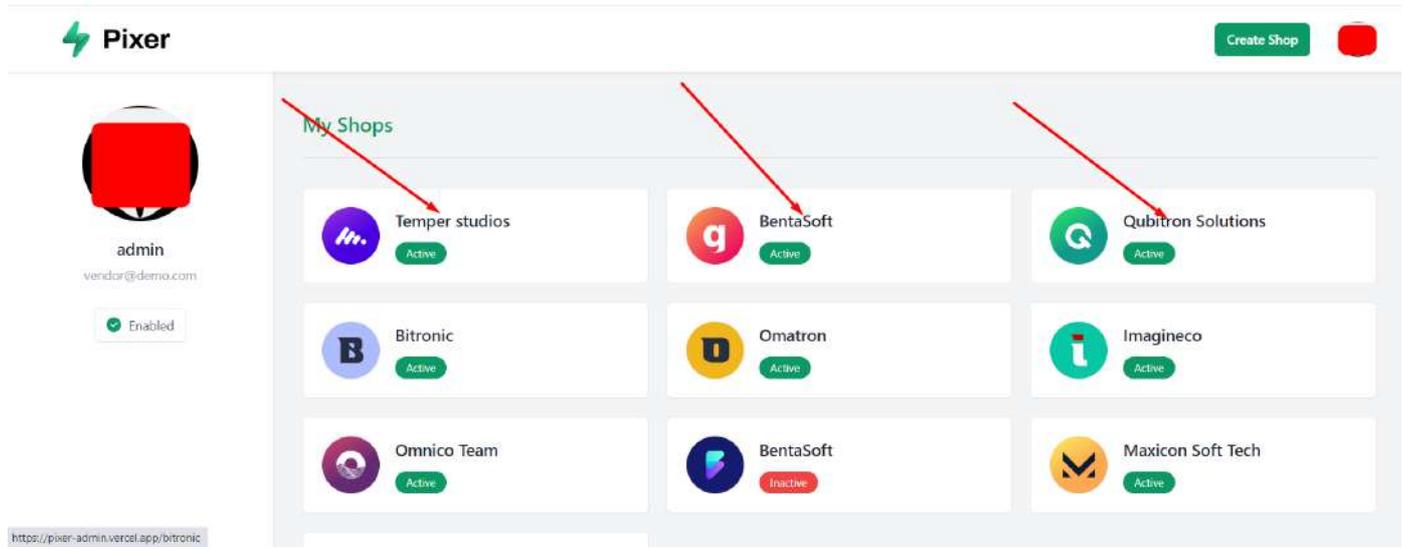
The procedure by which a seller can move their profits or balance from the Pixier platform to their own bank account or another payment method is known as a withdrawal mechanism in Pixier. In most withdrawal systems, the seller must ask to remove money from their account balance. With the seller's identification being confirmed, the platform may impose conditions regarding the minimum withdrawal amounts or withdrawal costs.

The platform will transfer the money to the seller's chosen payment method as soon as the withdrawal is accepted. The payment method and the platform's rules will determine how long this procedure will take, which might be several business days.

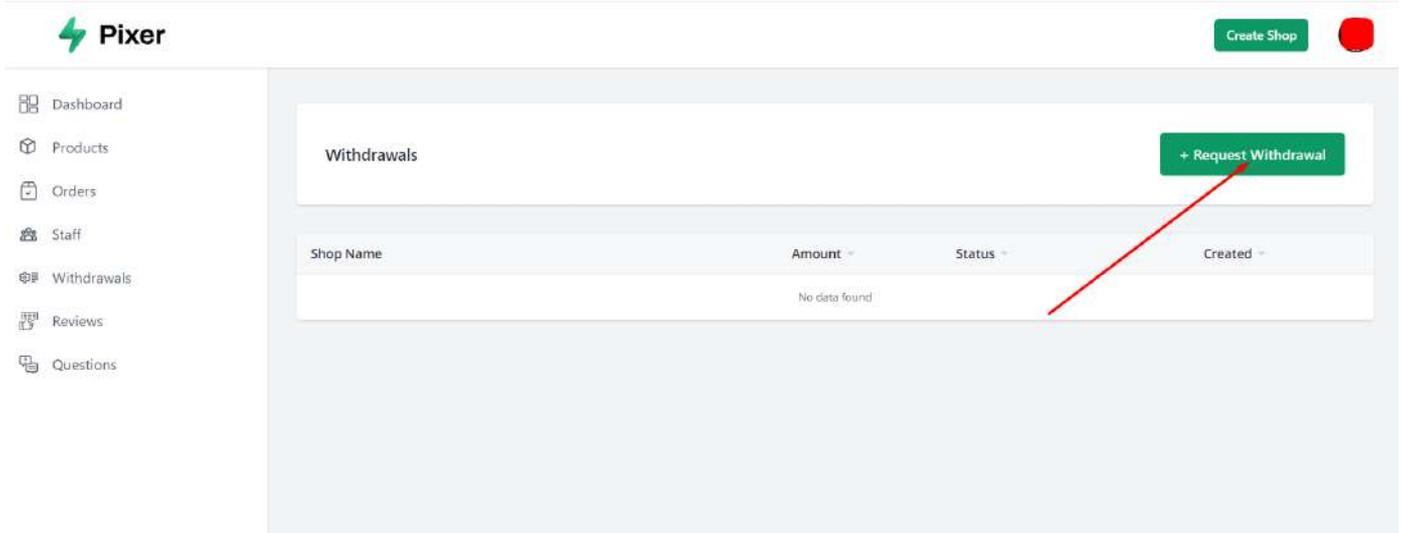
The ability for sellers to obtain their revenues and use them for either personal or professional objectives makes withdrawal mechanisms an essential component.

## Vendor

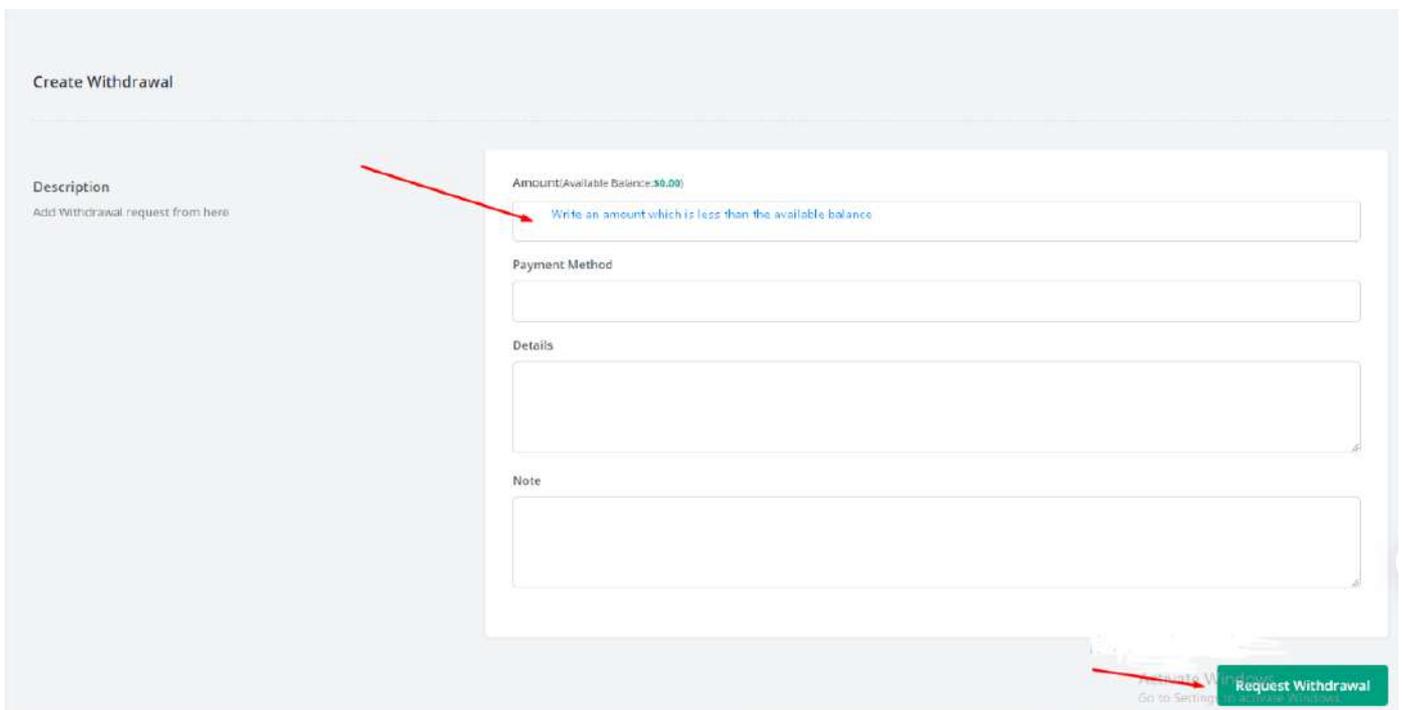
The store owner must first log in, then from the dashboard, we must locate "my shops". The seller's shop is where he may locate withdrawal options. The options below will be displayed if he visits his own store.



The seller may submit withdrawal requests under the withdrawals area.

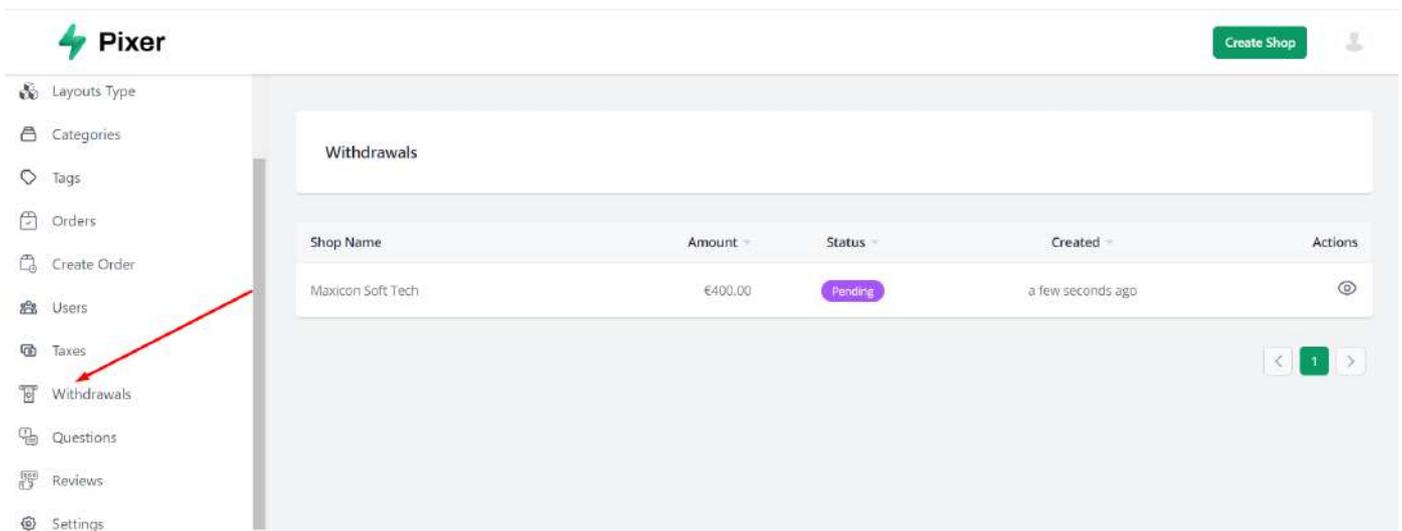


The seller can select **request withdrawals** and enter a request for a lesser quantity than what is now available.



### Super Admin

Requested withdrawals will show up in the "recent withdrawals" section of the admin dashboard. The administrator can use the action function and change the withdrawal request's status from pending to approved by clicking on **change status** button..



**Pixer** Create Shop

Layouts Type  
Categories  
Tags  
Orders  
Create Order  
Users  
Taxes  
Withdrawals  
Questions  
Reviews  
Settings

### Withdrawals

Shop Name	Amount	Status	Created	Actions
Maxicon Soft Tech	€400.00	Pending	a minute ago	

1 >

### Withdrawal Information

Amount : 400

Payment Method : paypal

Status : Pending

Pending Change Status

### Withdrawal Information

Amount : 400

Payment Method : paypal

Status : Pending

Approved Change Status