Pixer React Documentation

> For updated doc, please follow our online doc,

https://pixer-react-doc.vercel.app/

Introduction

Fastest E-commerce template built with React, NextJS, TypeScript, React-Query, NestJs & Tailwind. Its very easy to use, you can build your schema very easily. we use swagger for

rest documentation, and your frontend team will love using it.

Navigation

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

Supported Platforms For Local Development

- Compatible Browsers (Firefox, Safari, Chrome, Edge)
- Server(Node.js 12.13.0 or later)

- MacOS, Windows, and Linux are supported

Requirements

- node(12.13.0 or later)
- yarn
- 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
- React-Query
- Axios
- Tailwind
- Typescript
- React Hook Form

Shop

- NextJs
- React-Query

- Axios
- Typescript
- Tailwind
- Stripe Integration
- React Hook Form

Mock API

- NestJS

Getting Started & Installation

For getting started with the template you have to follow the below procedure.

API

- Go to `pixer-api` directory and run below command.

```
# on pixer-api directory
yarn
yarn dev
```

NOTE : This will start the server at `http://localhost:5000/api/` for rest api and you can access it from browser at `http://localhost:5000/docs`(for rest swagger doc).

Admin(REST)

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

1. Go to `admin` folder.
2. Copy the contents of `.env.template` into a new file called `.env`
3. Run below command to start the server.

```
# on admin directory
yarn dev
```

Shop

For starting the shop part with corresponding api run below commands.

1. Go to `/shop` folder.
2. Copy the contents of `.env.template` into a new file called `.env`
3. Fill the `.env` file with your values like,

`NEXT_PUBLIC_REST_API_ENDPOINT=`,
`NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=`,
`NEXT_PUBLIC_WEBSITE_URL=`.

Run below command to start the server.

```
# on shop directory
yarn dev
```

NOTE : API must be running for the above commands to work.

NOTE : `.env` file must be filled with your values.

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

API

- Go to `/pixer-api` folder.
- Run `yarn build`
- Run `yarn start:prod`

Admin

- Go to `/admin` folder.
- Run `yarn build`

- Run `yarn start`

NOTE : API must be running for the above commands to work.

NOTE : `.env` file must be filled with your values.

Shop

- Go to `/shop` folder.
- Run `yarn build`
- Run `yarn start`

NOTE : API must be running for the above commands to work.

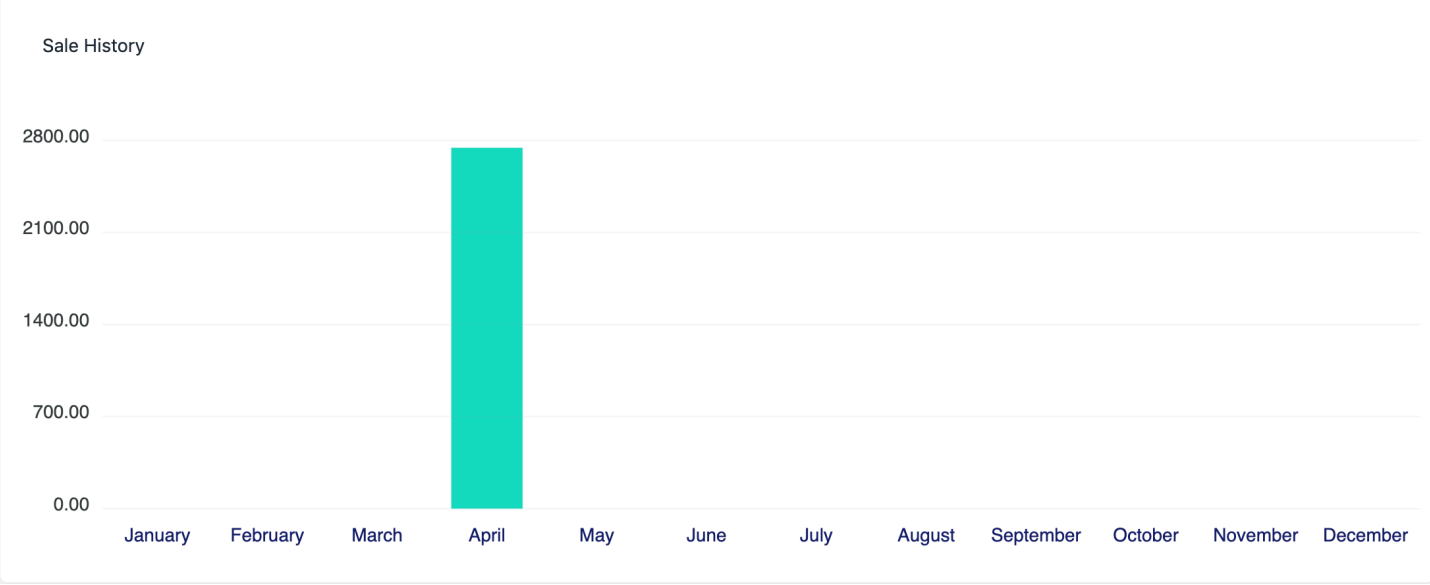
NOTE : `.env` file must be filled with your values.

Admin dashboard

- ## Analytics Dashboard

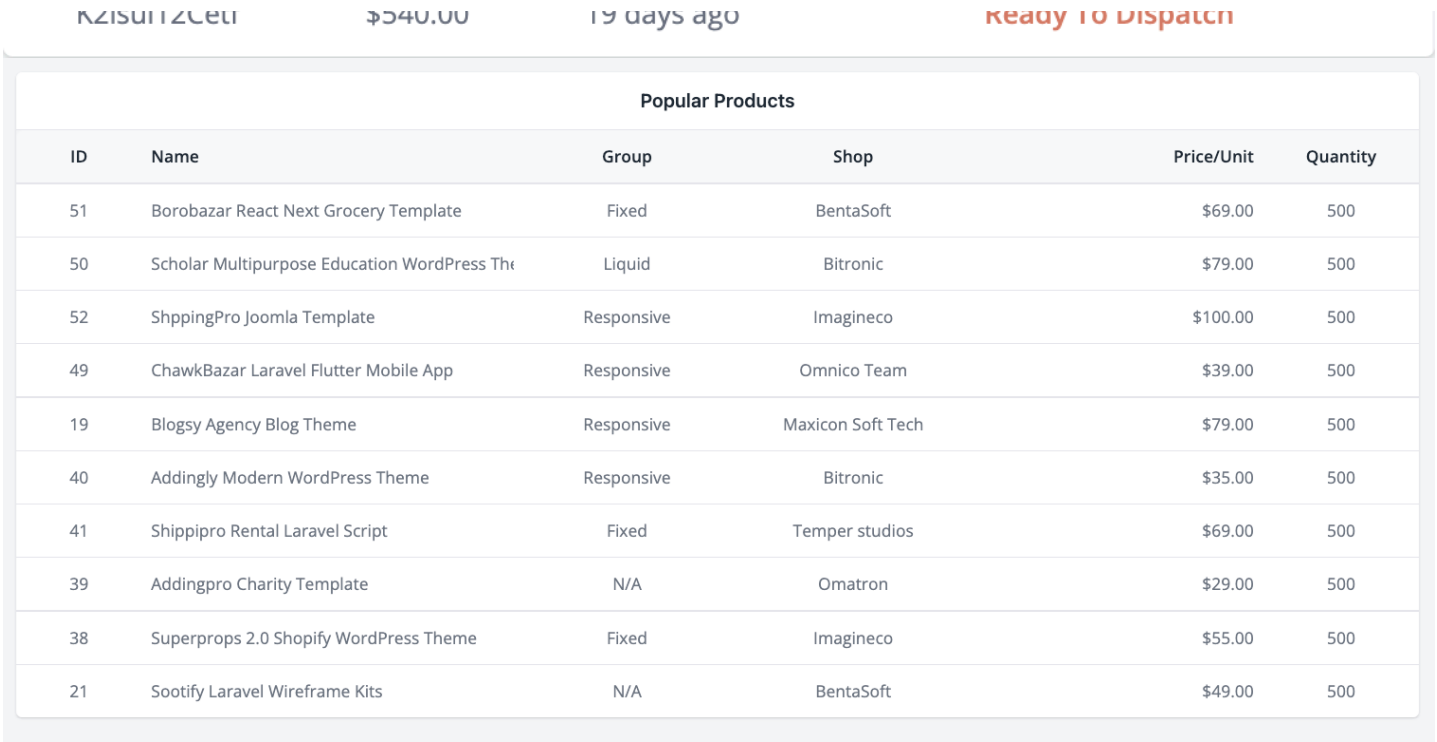
You will get a complete analytics dashboard to know the overview of your shop.

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



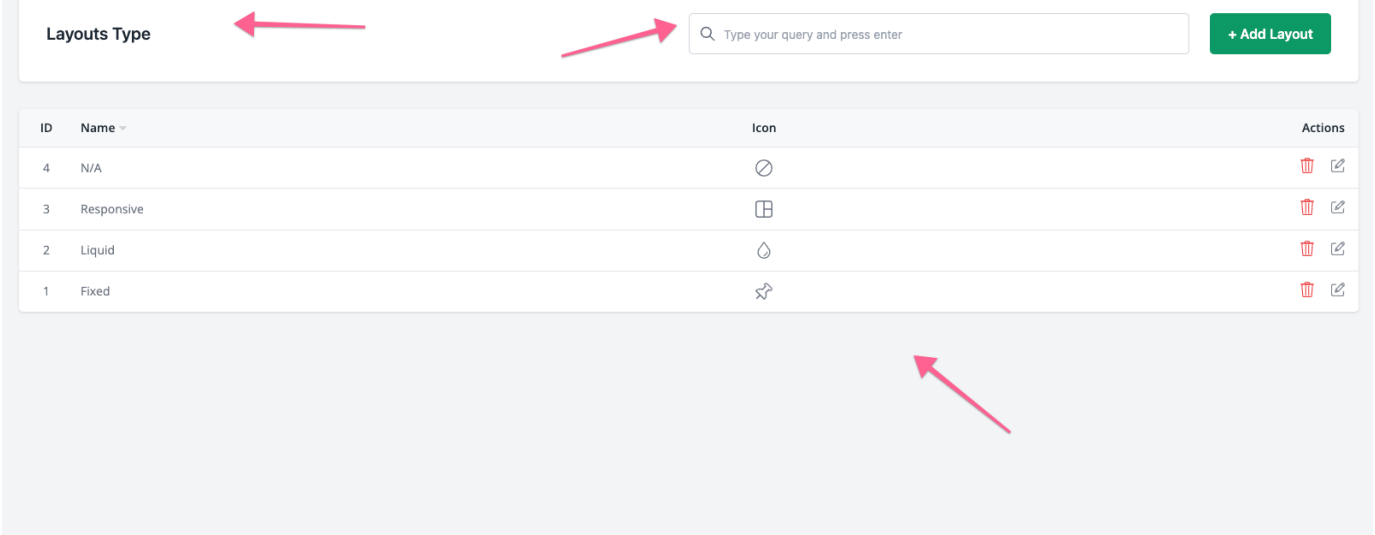
Recent Orders

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
ruIb1kz4UlIN	$2.45	19 days ago	Shipment Refused by Consignee
NqlykeIW6awe	$97.80	18 days ago	Delivered
TfzYmld1AR32	$18.00	18 days ago	Ready To Dispatch
PXqs8RXQBGm5	$7.20	19 days ago	Delivered



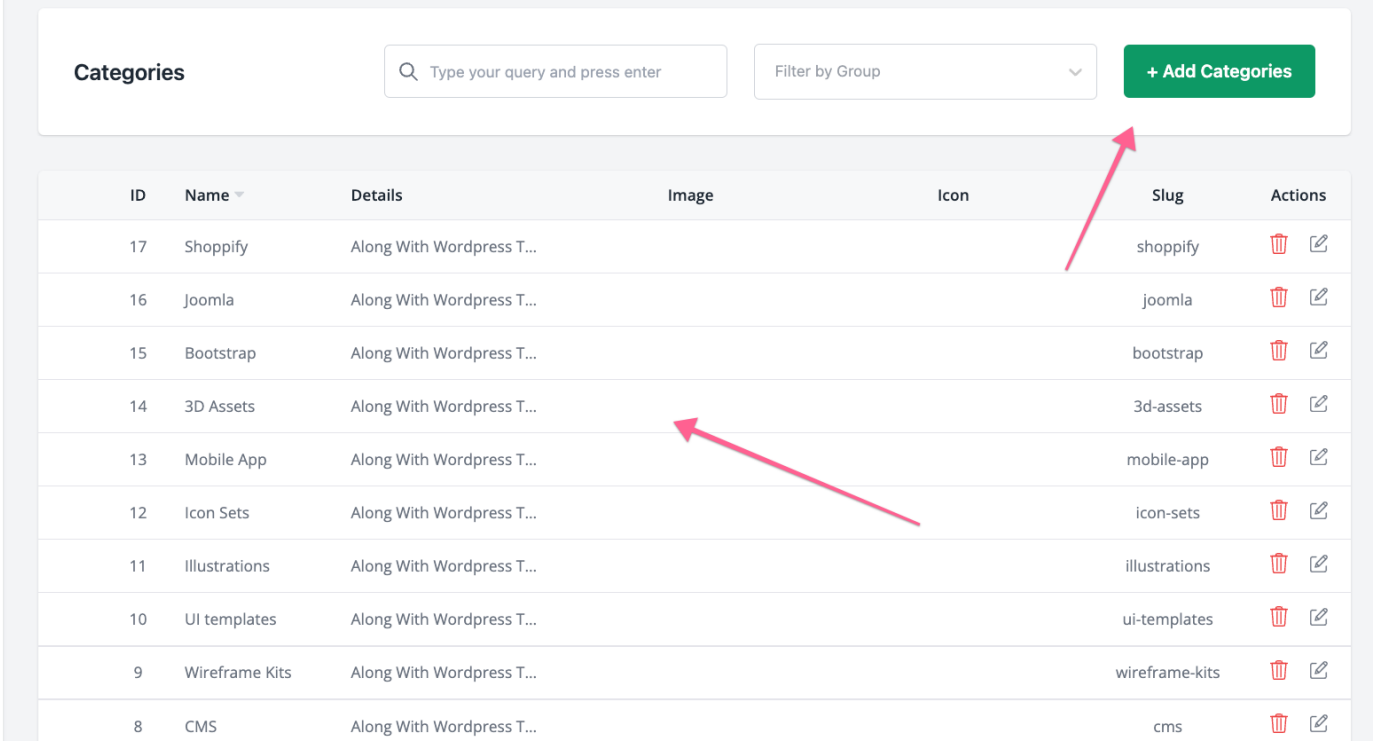
- # Manage Layout Type

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



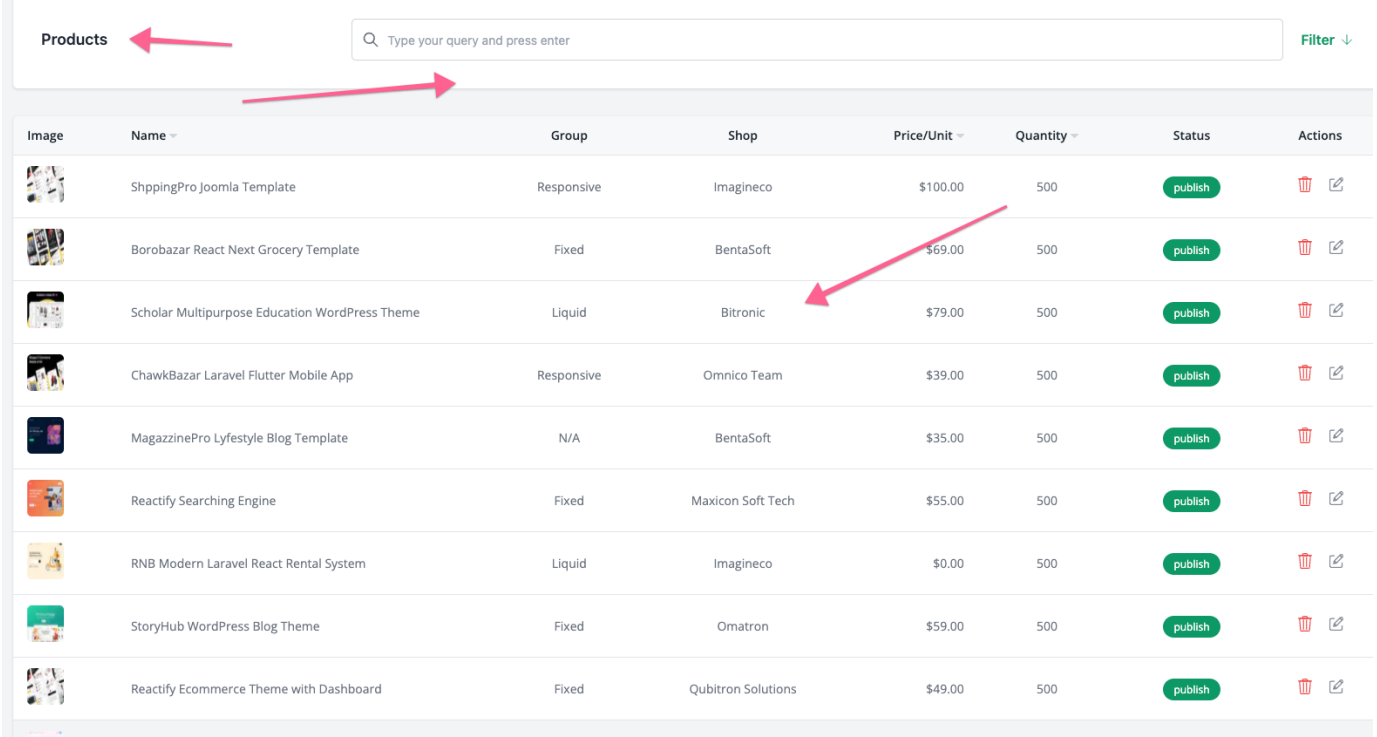
- # Manage Product Category

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

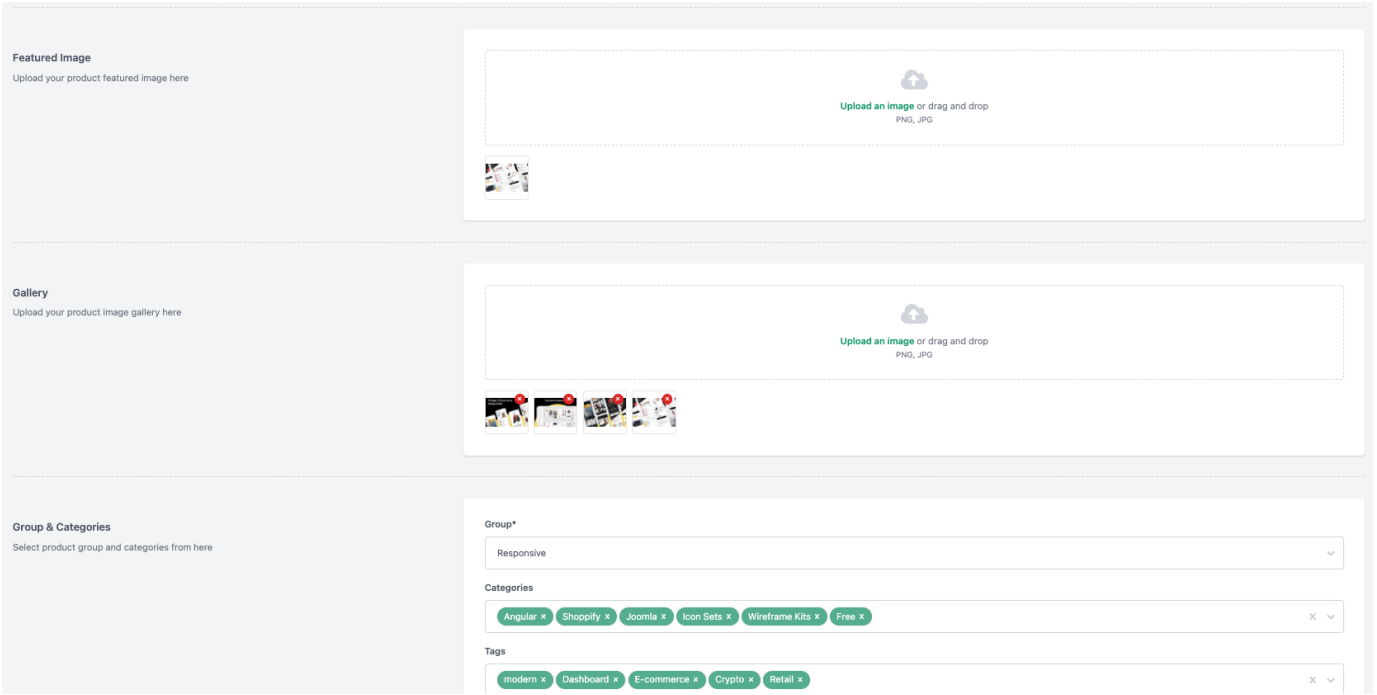


- # Product Management

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

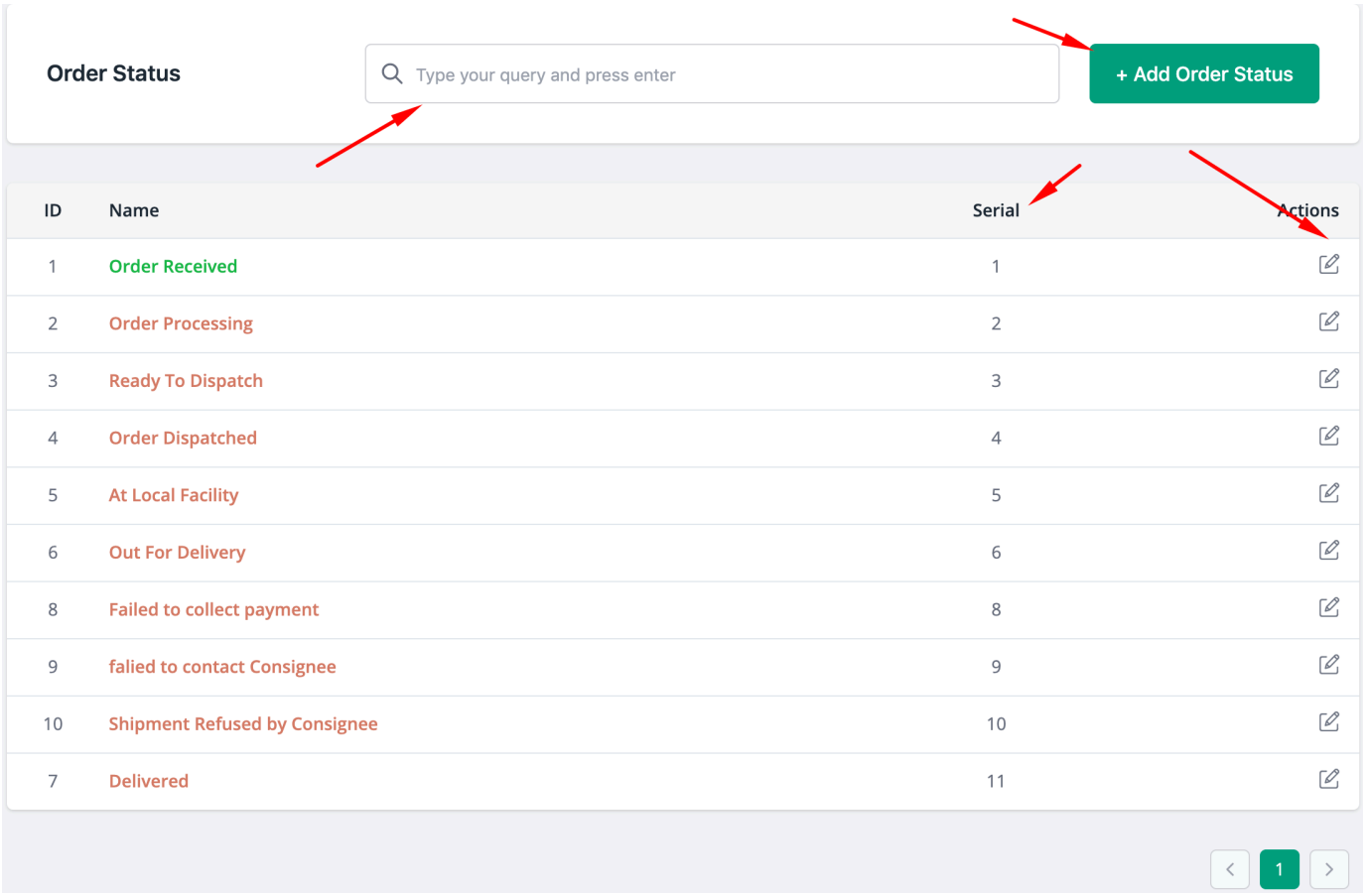


A portion of product form



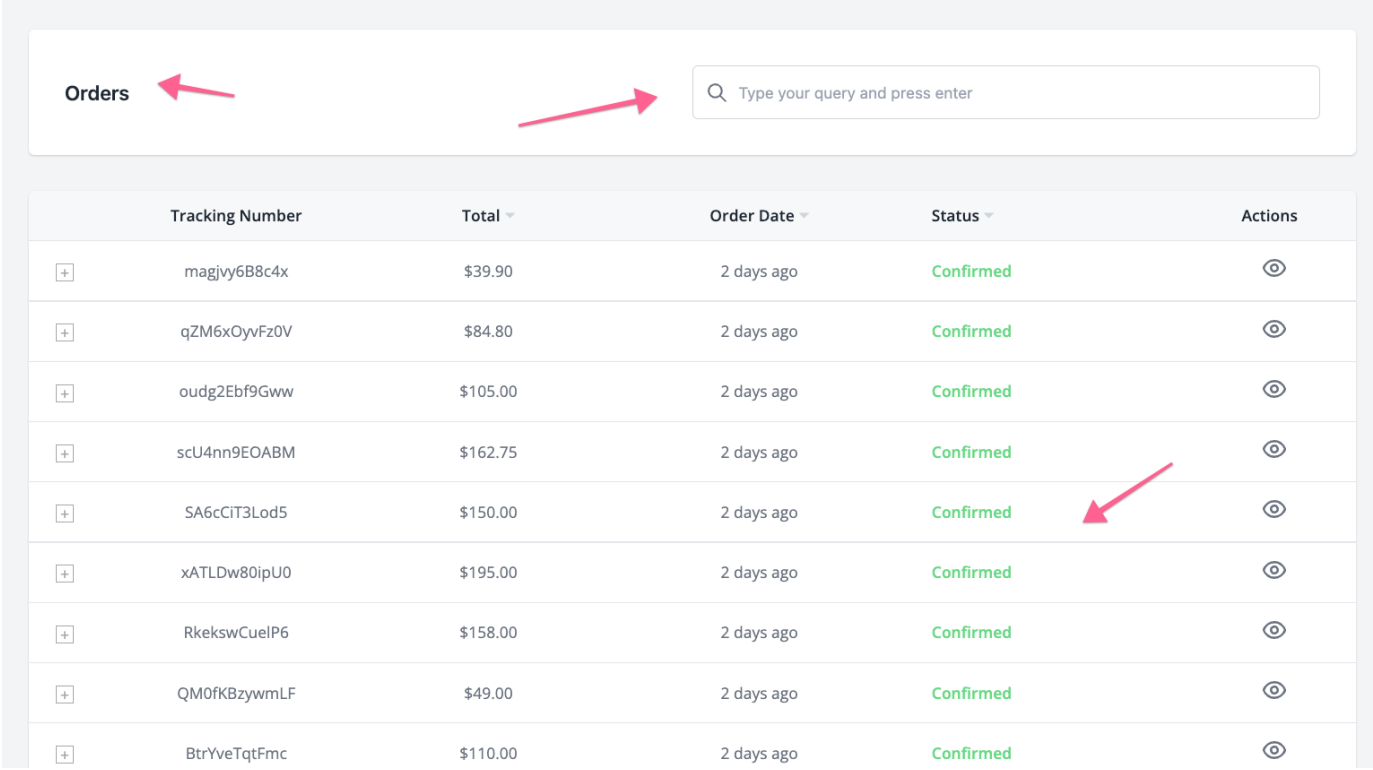
- ## Order Status

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



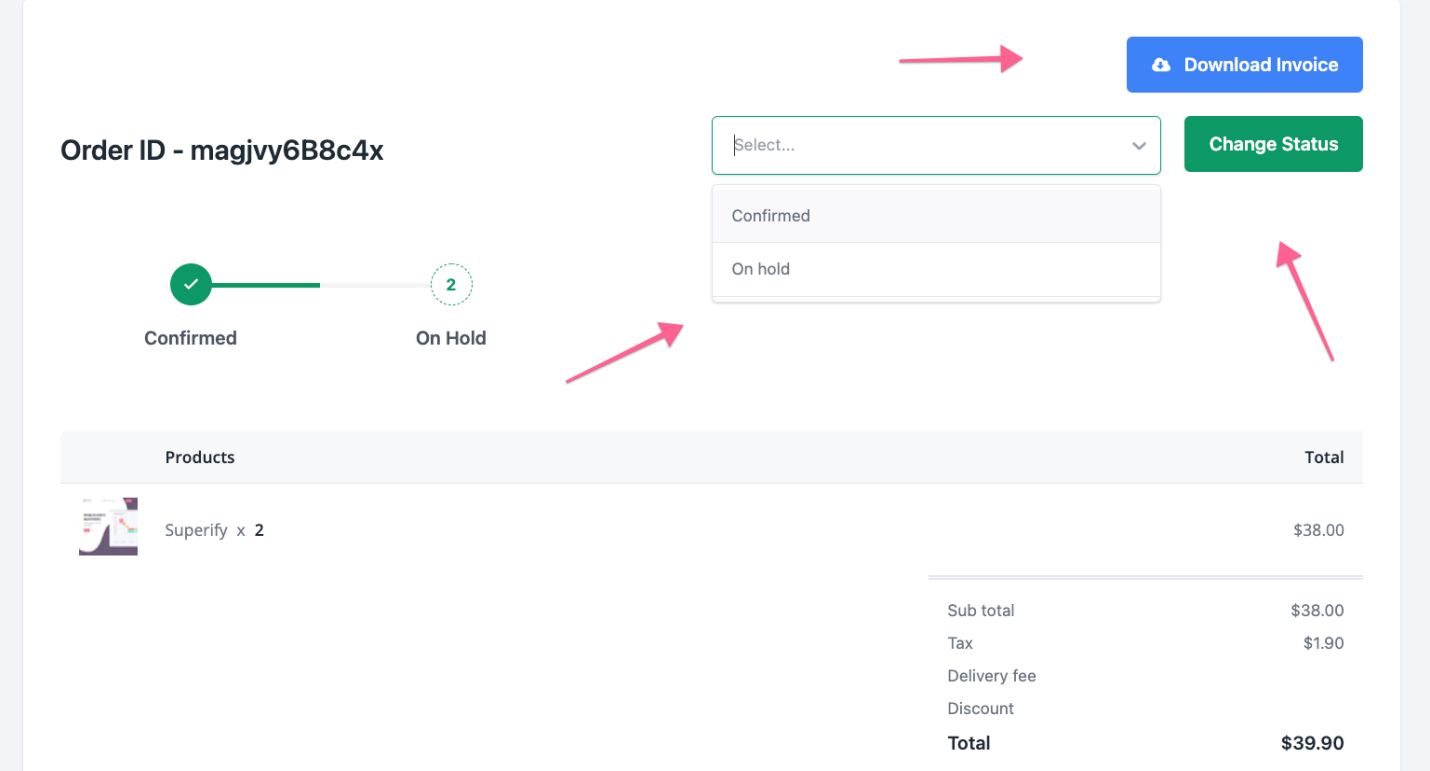
Order Management

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



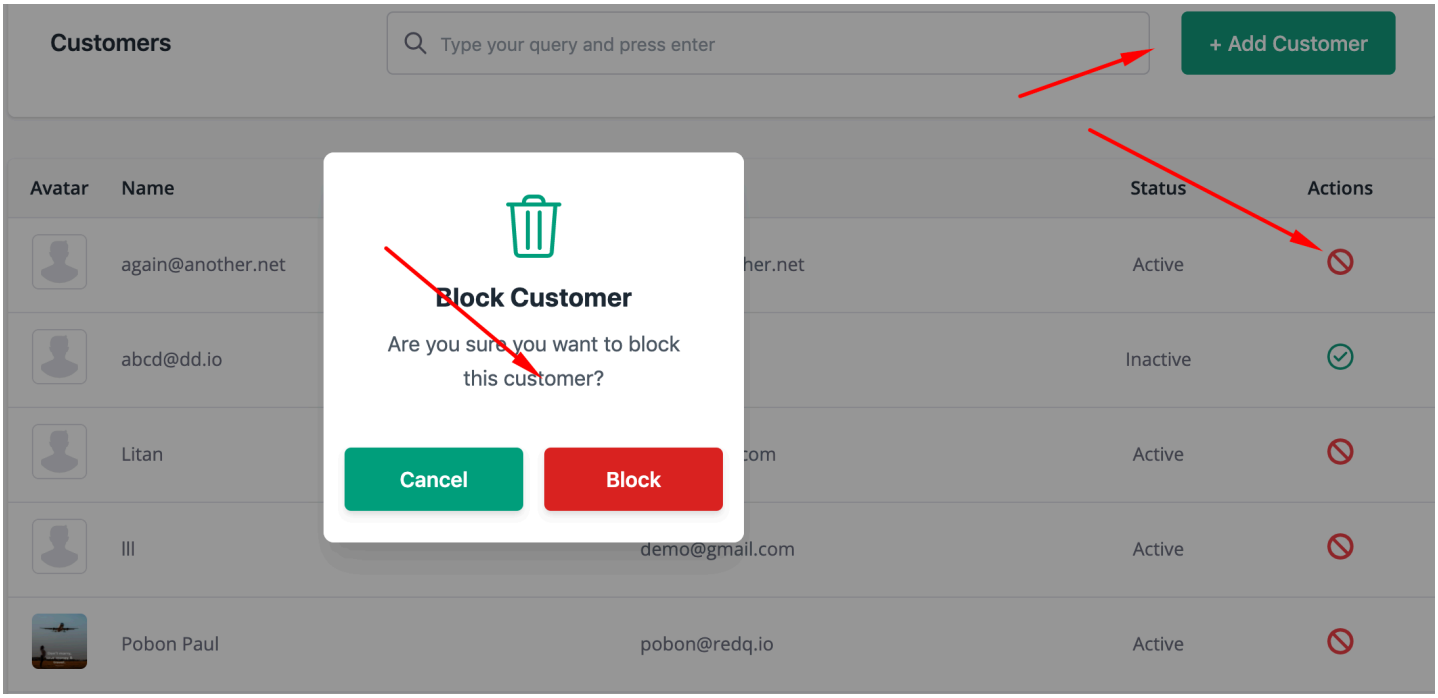
- A portion of order management.

Order status change.



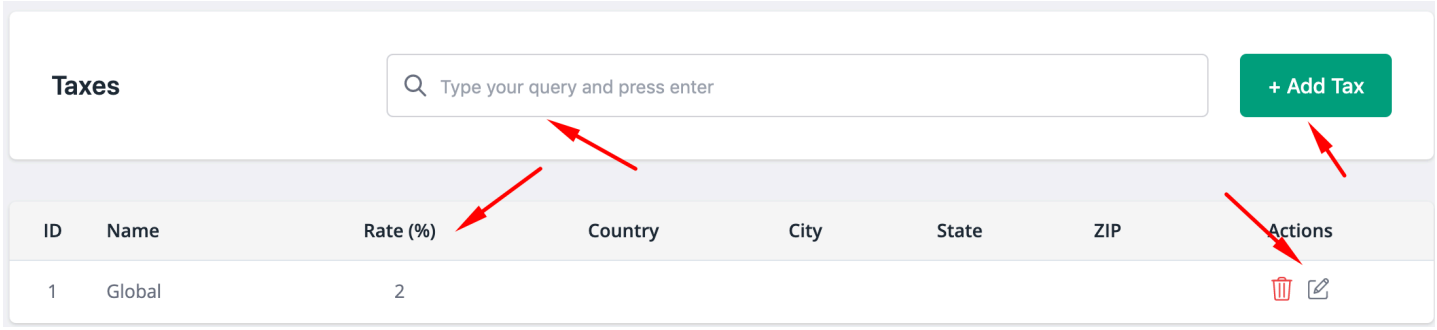
Customer Management

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



Tax Management

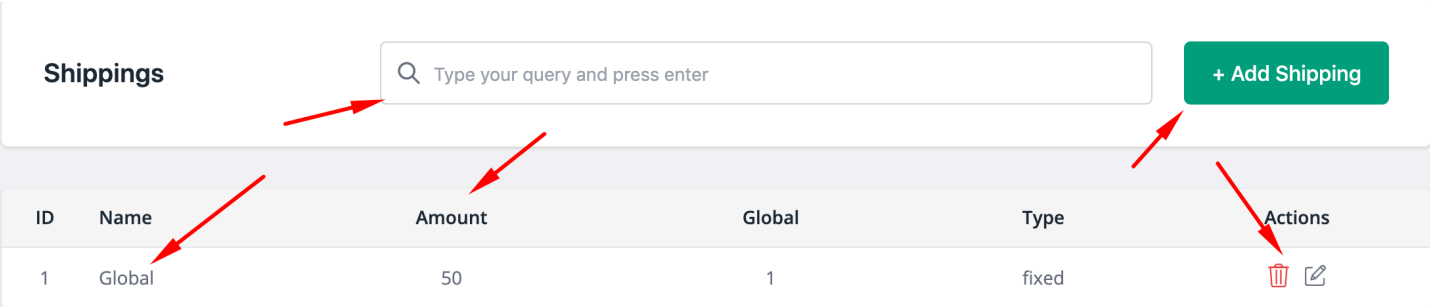
In `Tax` menu you will get the Tax list and you can add, remove or modify Tax from there.



Shipping Management

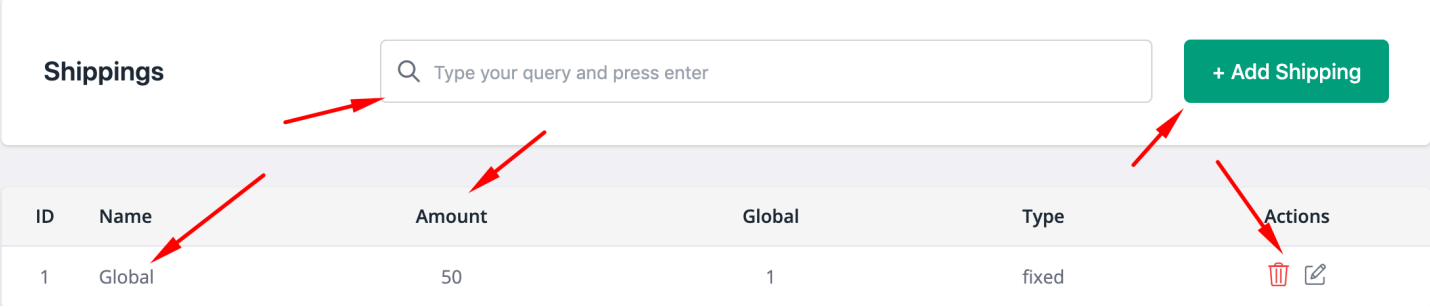
Shipping Management

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



Shipping Management

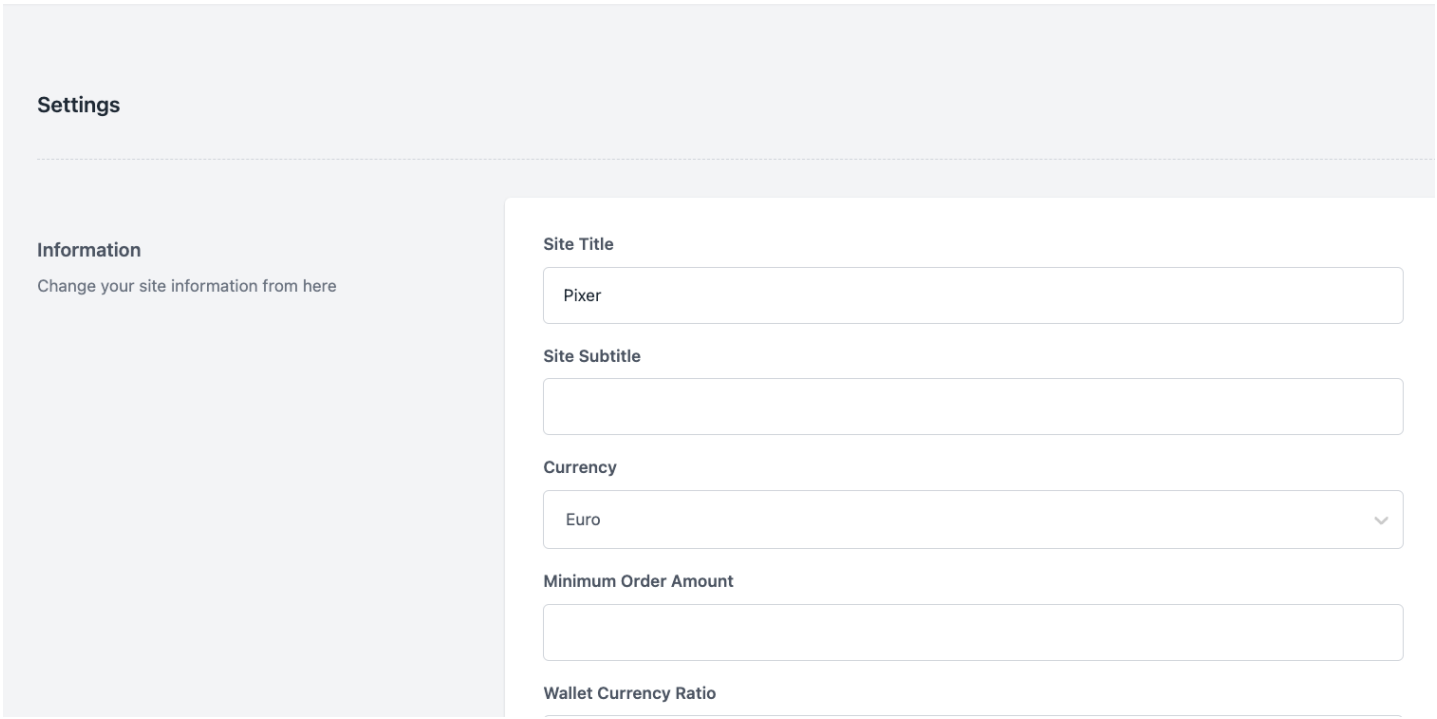
In `shipping` menu you will get the shipping list and you can add, remove or modify shipping from there.



Setting Management

Settings Management

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



Available Scripts:

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

Shop

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"
```

** Note: ** Also, individual Scripts are

** Note: ** Also, individual scripts are available under every individual package.
You can check out them from thier individual pacakge.json file.

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`

- `/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

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

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:

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 reffernce you can see one of our icon.)
5. If your custom SVG code has any single custom color then change them into fillColor.

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 mock rest api integration. We have used `react-query` ~ hook pattern ~ and fetched data from nest mock API. Please go to `data/` folder for those hooks.

- Creating the hook.
  - We have imported the `Product` type from `@/types` (We have used typescript path

aliasing for this. For more info please see our `tsconfig.json` file).

- We have built an `axios instance` and a wrapper class which called `http-client`.
- We have put all ours endpoint at `@data/client/endpoints` file using constant value.
- We have created a `client` class which will define all the methods required for API actions.

```
class Client{
  users = {
    me: () => HttpClient.get<User>
(API_ENDPOINTS.USERS_ME),
    update: (user: UpdateProfileInput) =>
      HttpClient.put<User>
(`${API_ENDPOINTS.USERS}/${user.id}`, user),
    login: (input: LoginUserInput) =>

      HttpClient.post<AuthResponse>
```

```
(API_ENDPOINTS.USERS_LOGIN, input),
    register: (input: RegisterUserInput) =>
      HttpClient.post<AuthResponse>
(API_ENDPOINTS.USERS_REGISTER, input),
    forgotPassword: (input: ForgetPasswordInput) =>
      HttpClient.post<PasswordChangeResponse>(
        API_ENDPOINTS.USERS_FORGOT_PASSWORD,
        input
      ),
  }
  ....
}
export default new Client();
```

using the client class

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

```
```

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

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

```
import type { Product } from '@/types';
import { 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());
```

API Documentation

We add `Swagger UI` for API documentation. You'll get that in NestJS API.

https://docs.nestjs.com/openapi/introduction

Or for live link,

Swagger UI

https://mock.redq.io/pixer-docs

Also, we implement `Laravel` on our Pixer Laravel project. You'll also get a full postman document from this URL,

https://documenter.getpostman.com/view/11693148/UVC5Fo9R

AWS (Amazon Web Service)

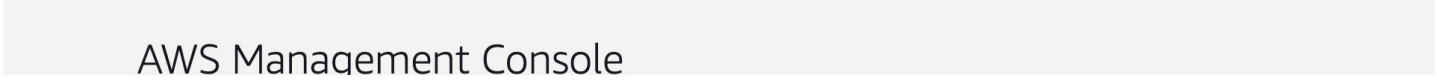
AWS (Amazon web service)

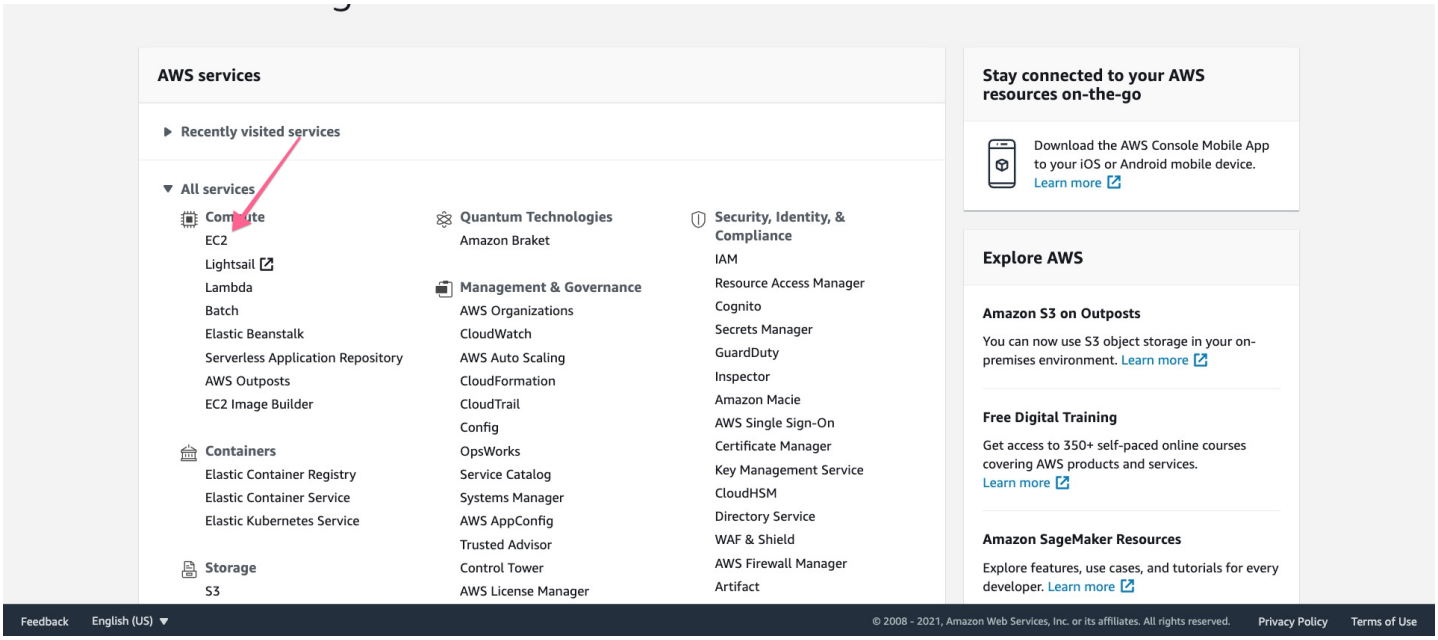
> If you want to use all the scripts (`shop`, `admin`, `api`) on the same server as this tutorial, then we recommend creating a blank ubuntu-based server with at least 2+ CPU cores and 2GB+ memory.

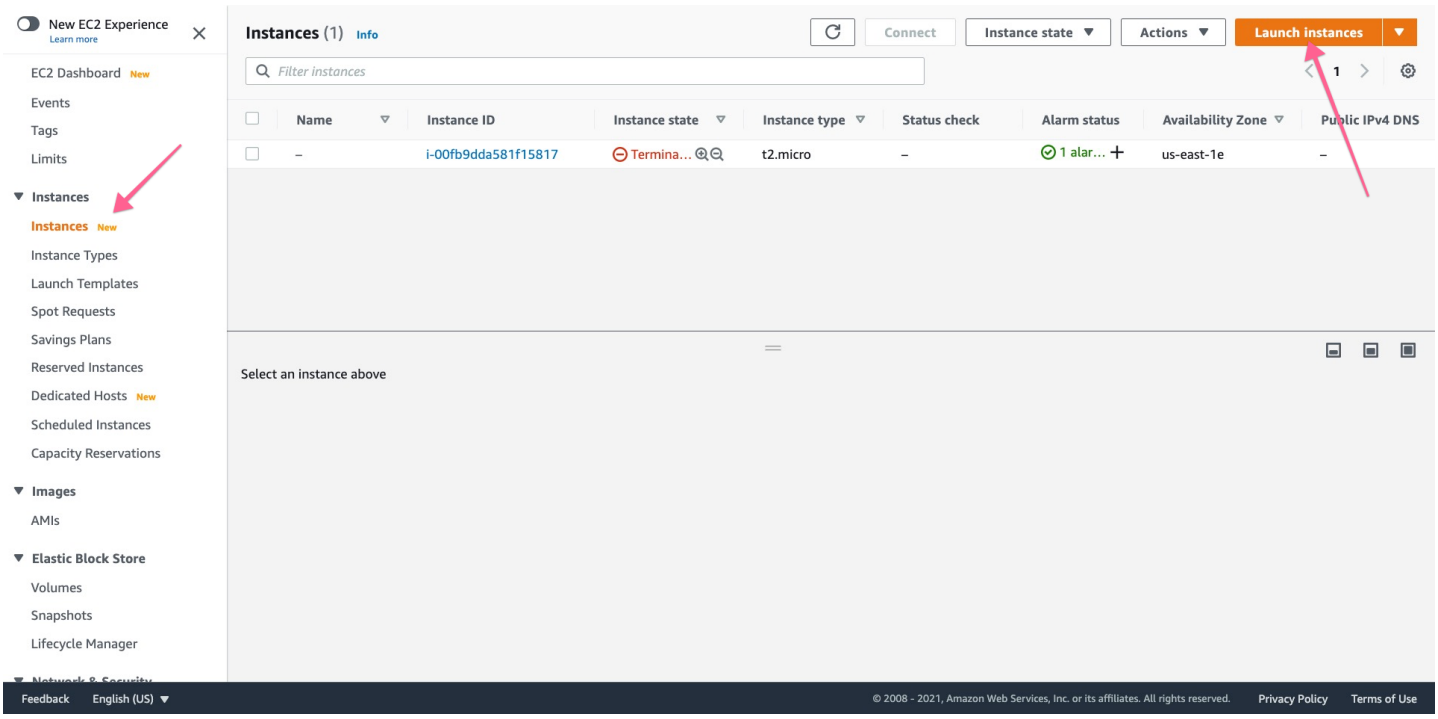
How to create ec2 server?

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
```



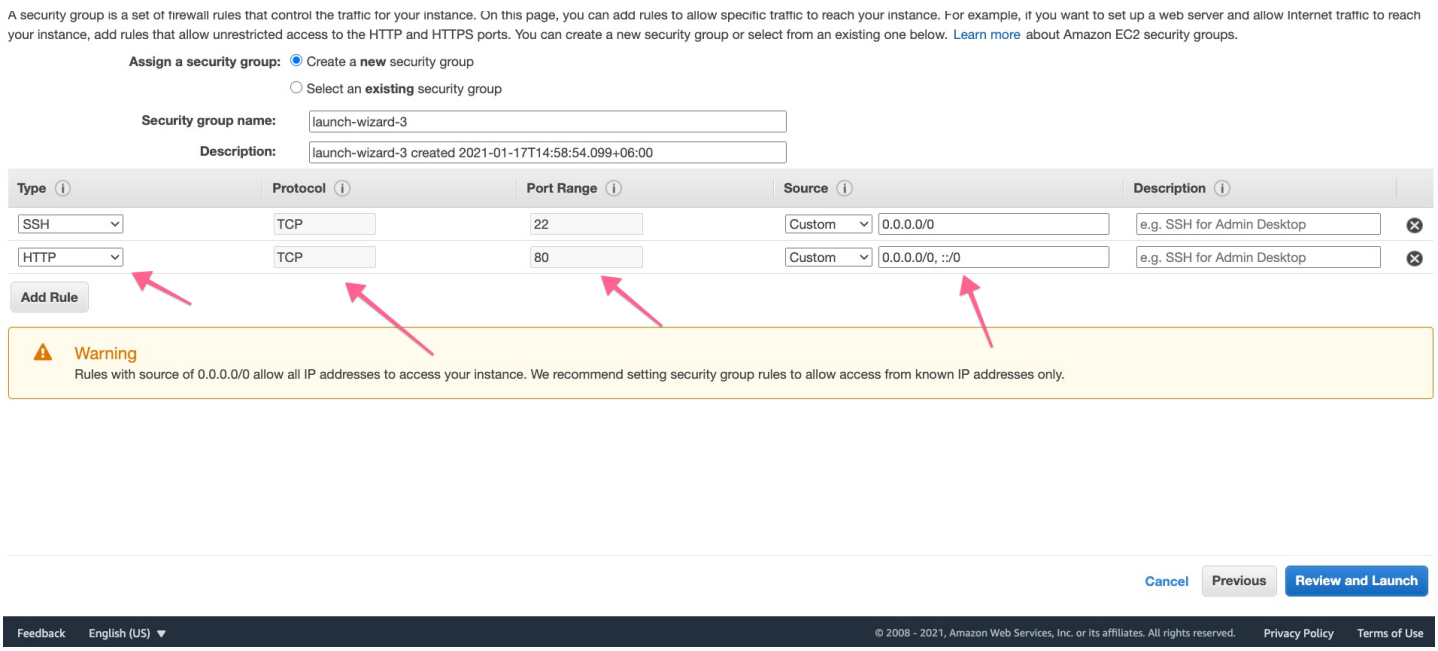




Then select a ubuntu 20.04 server

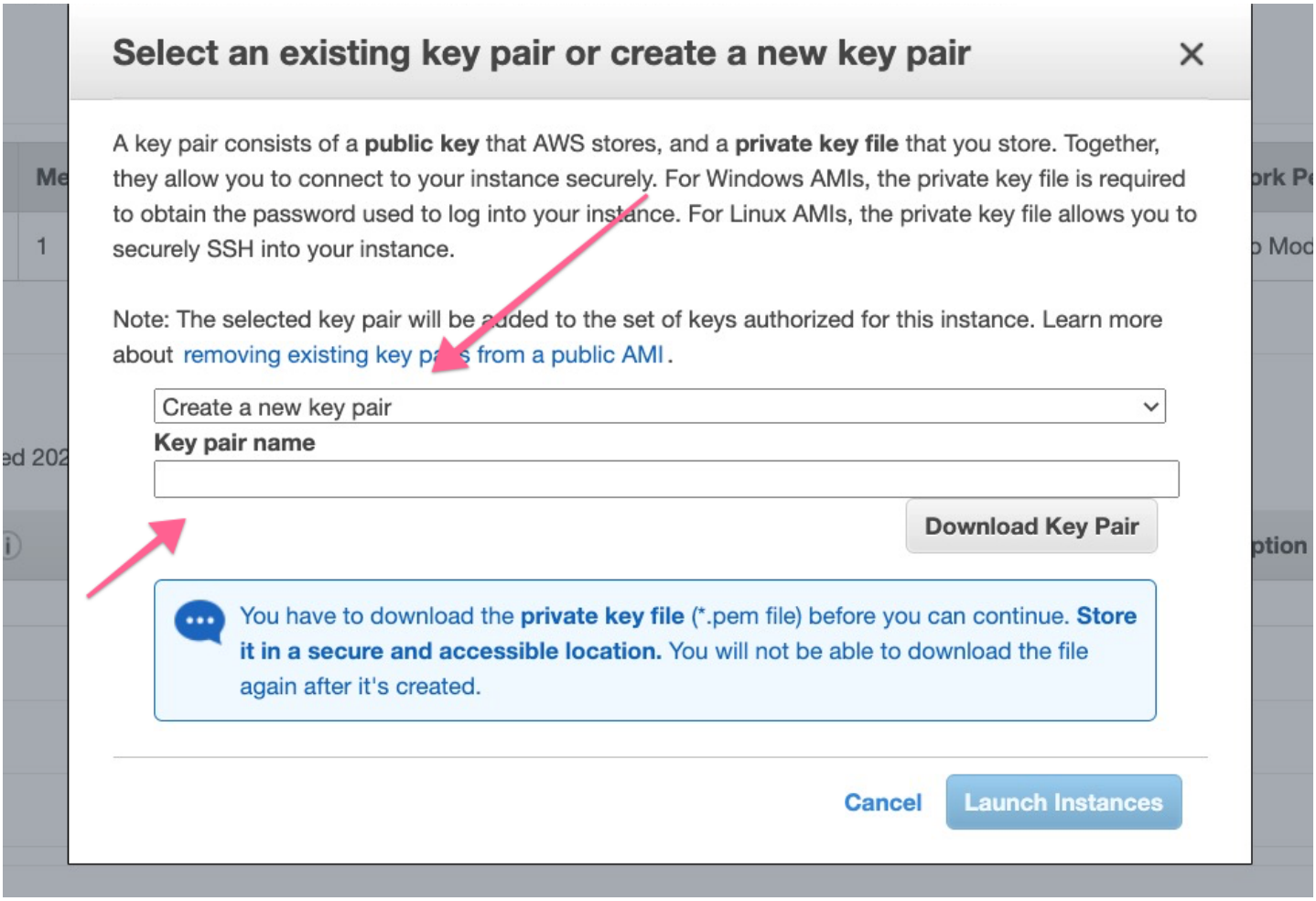
After that, click `Next` → `Next` → `Next` → `Next`

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

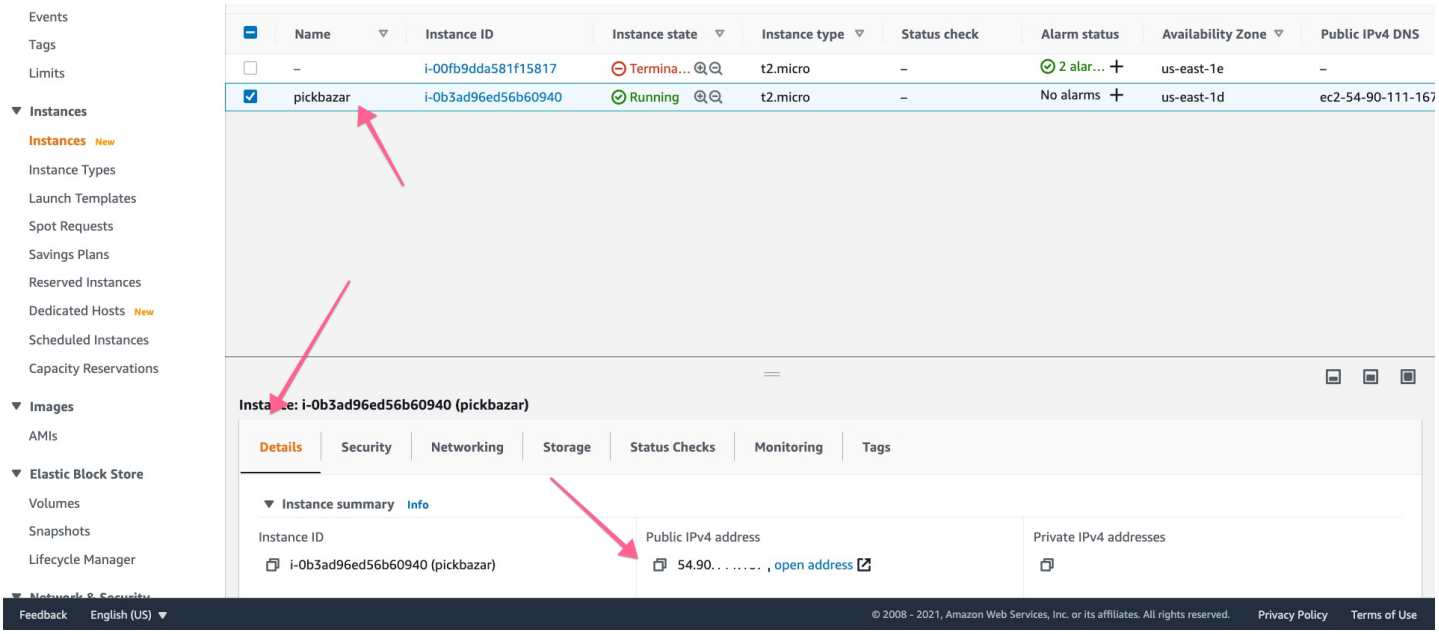


After review, click `Launch,` and you'll get and popup for KeyPair, which will be required to login to the server using ssh.

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.

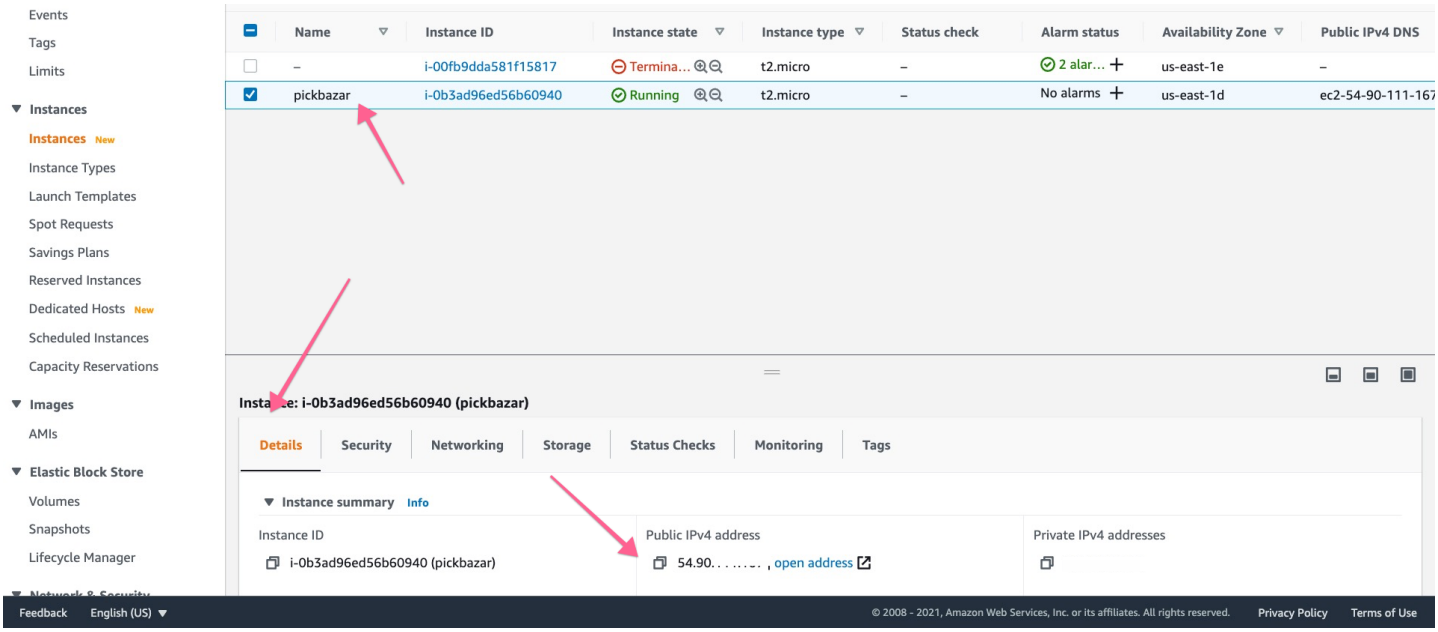


After launching the instance, you'll get the server IP, which will be required to login into ssh.



Domain Setup

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



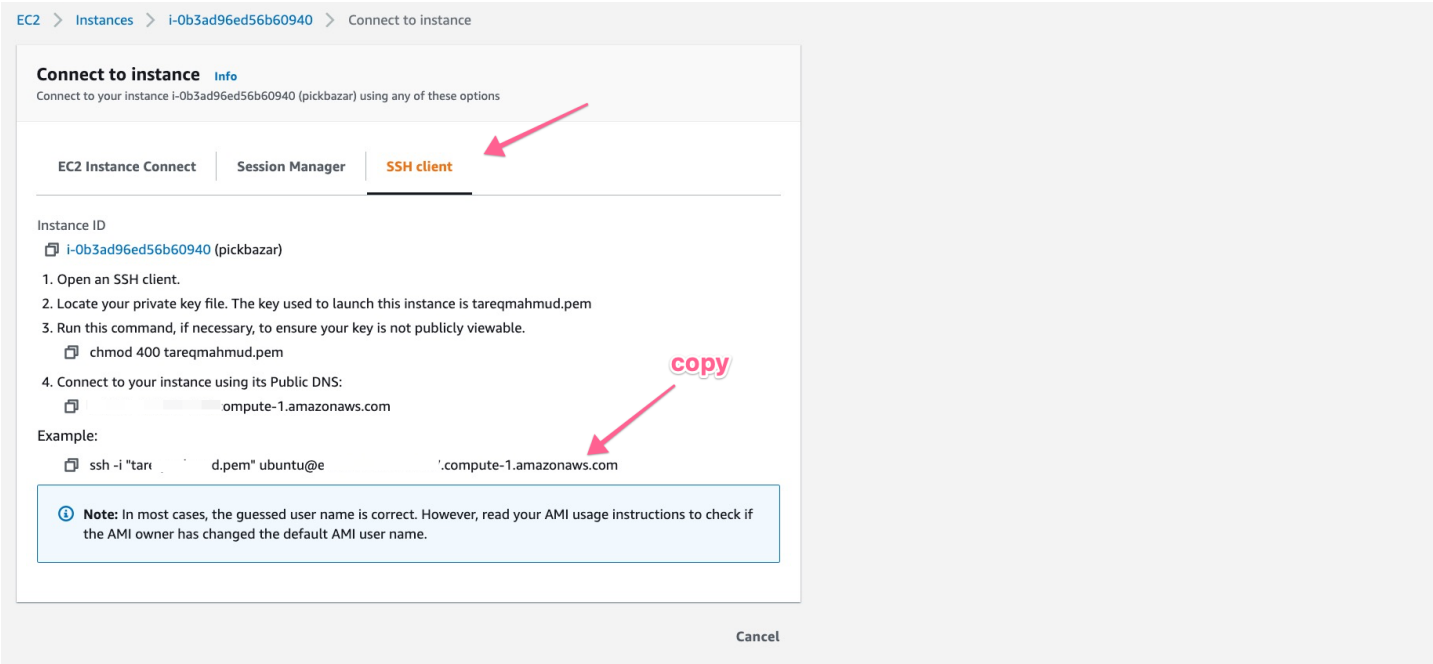
> “
> Please contact your domain provider for detailed explanation of how to do that.

Login to Server

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

then click `Connect`

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



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

- **Change permission .pem**

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

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

```
chmod 400 pixer.pem
```

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

> “
> Now go to the `VPS Server` section for deploy the `Pixer React`

VPS Server

Virtual Private Server

With this tutorial, you can install Pixer to

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

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

Access Server

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

Install NodeJS & Required

Application

Install NodeJS

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

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

Install Yarn

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

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

```
sudo npm install -g yarn
```

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

Npm Permission Issue

Install Zip & Unzip

```
sudo apt install zip unzip
```

Install PM2

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

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

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

```
pm2 startup systemd
```

Setup Server

Introduction

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

Step 1 - Installing Nginx

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

```
sudo apt update
```

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

```
sudo apt install nginx
```

Step 2: Adjusting the Firewall

Before testing Nginx, the firewall software

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

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

```
sudo ufw app list
```

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

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

At first, add ssh to the firewall,

```
sudo ufw allow ssh
```

```
sudo ufw allow OpenSSH
```

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

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

Now enable the firewall,

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

You can verify the change by typing:

```
sudo ufw status
```

The output will be indicated which HTTP traffic

is allowed:

```
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 - Change permission for the

`www` folder

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

use this command to create a directory on `/var/www/` on your server

```
mkdir /var/www/pixer-react
```

Step 9: Setting Up Server & Project

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

At first, we'll disable the default configuration.

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

Step 10 - Create New Nginx for the domain

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

Add this Nginx config file to that edited file,

```
server {
    listen 80;

    server_name YOUR_DOMAIN.com;

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

    index index.html index.htm index.php;

    charset utf-8;
```

```
    charset utf-8;

    # For API
    location /backend {
        proxy_pass http://localhost:5000;
        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;
    }

   # 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_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 ~ /\.(?!well-known).* {
        deny all;
    }
}
```

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

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 8 - Build API**

> “
> Because of the small server, our procedure is to build scripts on a local pc and run it on the server. If your server is at least 4+ CPU core and 8GB+ memory, you don't have to build it on your pc. Just upload the scripts to your server and build them directly on the server.

To build the scripts, Download the scripts from themeforest and then,

```
1. Extract the `pixer` package that you download from `ThemeForest`
2. Move `pixer` folder to generic folder like desktop, document, etc.
```

Then go to,

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

and use this command

```
yarn
yarn build
```

Step 8 - Upload API to Server

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

After that run API as a pm2 process

go to `/var/www/pixer-react/pixer-api` and run this command,

```
pm2 start --name=mock-rest yarn -- start:prod
```

Now, when you go to the `YOUR_DOMAIN/backend/api`

you'll get a page like this

FrontEnd Project Build

> Typescript requires a huge chunk of memory to build the project, so if your server has at least 8gb+ of memory, then you can build the project on your server directly. If not, then build the project on your server, then move
>
> the folder to the server then serve the

> project. We'll do the second method in this tutorial.

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

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

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

Edit `admin/next.config.js`,

add `basePath` for '/admin'

```
module.exports = {
basePath: '/admin',
images: {
    domains: [
```

Step 2 - Install & Build

go to your `pixer` 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`

and use

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

```
backend/api"
```

After that, go to the `admin` folder,

```
cd ../admin
```

then use this command to copy,

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

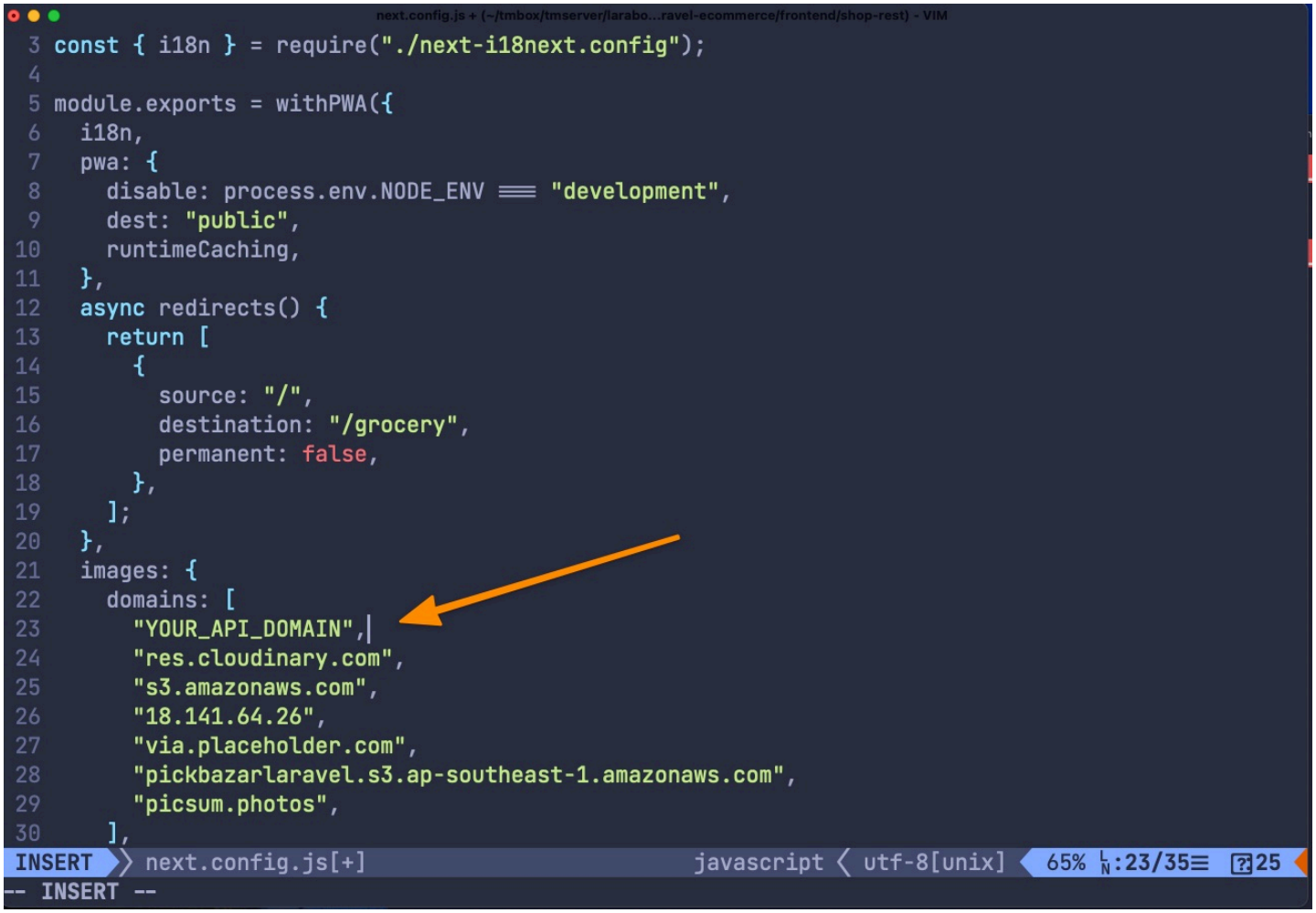
and use

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

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

`next.config.js`

and add your domain to `images` object



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

- Build Project

- **Admin**

  - Go to `/admin` folder.
  - Run `yarn build`

- **Shop**

  - Go to `/shop` folder.
  - Run `yarn build`

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

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

Then `unzip` the `frontend` zip file.

Install FrontEnd And Run

- **Run frontend app**

For `shop` app, go to `/shop` folder and run this command,

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

For `admin` app, go to `/admin` folder and run this command,

```
yarn
```

Then to run the `admin` app, use this command,

```
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

the shop page and `YOUR_DOMAIN/admin` for the access admin section.

Vercel

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

> Before procedure vercel, you've to host API on a server, and after that, follow this procedure.

- vercel.com

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

Frontend

Step 1:

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

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

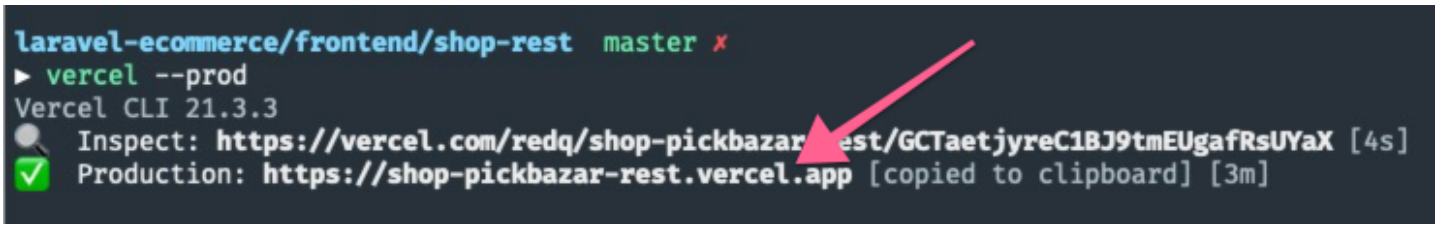
Step 2:

- Go to `admin` directory and install packages and build packages using this command

```
vercel
```

- Similarly, go to `shop` directory and install packages and build packages using this command

```
vercel
```



** NOTE: ** for deploying to `vercel` you need to install `vercel-cli` on your machine for more information please visit here

- **For others hosting providers:**

Please follow nextjs deployment docs:

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/shop/src/pages
```

- **Admin,**

```
pixer/admin/src/pages
```

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

pages and routing,

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

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

Thank You!