## Day 03: API Integration & Data Migration

## Introduction

Shopify is a popular eCommerce platform that allows users to build online stores and sell products. Shopify provides an API that lets you fetch store data such as products, collections, and orders. By integrating Shopify's API into your **Next.js** project, you can display dynamic content such as products and collections on your website. **Next.js** provides server-side rendering (SSR) and static site generation (SSG), making it a great choice for building fast, SEO-friendly web applications.

Step 1: Shopify Store Setup  
  
1.1 Create a Shopify Store  
  
 If you haven't already, you'll need to create a Shopify store first.

 Visit [Shopify](https://www.shopify.com" \t "_new) and sign up.

 Set up your store with a name, theme, and products.

 Log in to your Shopify Admin panel.

## 1.2 Create a Private App

To access Shopify’s API, you need to create a private app that provides an API key and password.

 In the Shopify Admin panel, go to Apps.

 Click on Manage private apps and then Create a new private app.

 Provide an app name and set the required API permissions (e.g., Products, Orders, Inventory).

 Once the app is created, you’ll be provided with an API Key and API Password. These credentials will be used to authenticate API requests.

## Step 2: Next.js Project Setup 2.1 Initialize a New Next.js Project

To create a new Next.js project, use the following command:



This will set up a new Next.js application.

2.2 Install Dependencies

Next, install Axios for making HTTP requests and dotenv for managing environment variables.  
  
  
  
 Axios: Used to make HTTP requests to the Shopify API.

 dotenv: Used to securely manage environment variables such as the Shopify API key.

## Step 3: Setup Environment Variables 3.1 Create .env.local File

Create a .env.local file to store sensitive information such as your Shopify API credentials.

File: .env.local



 SHOPIFY\_STORE\_URL: The URL of your Shopify store (e.g., your-shopify-store.myshopify.com).

 SHOPIFY\_API\_KEY: The API key from your Shopify private app.

 SHOPIFY\_API\_PASSWORD: The API password from your Shopify private app.

## Step 4: Install Dependencies

Make sure you have installed the necessary dependencies like Axios and dotenv.



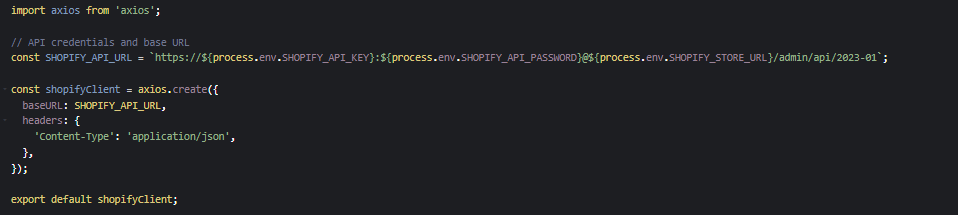
These dependencies will allow you to interact with the Shopify API and securely manage sensitive credentials.

## Step 5: Create Shopify API Client

## 5.1 Create an API Client

Create a file to handle interactions with the Shopify API. This file will authenticate the requests with your API credentials.

File: lib/shopify.js



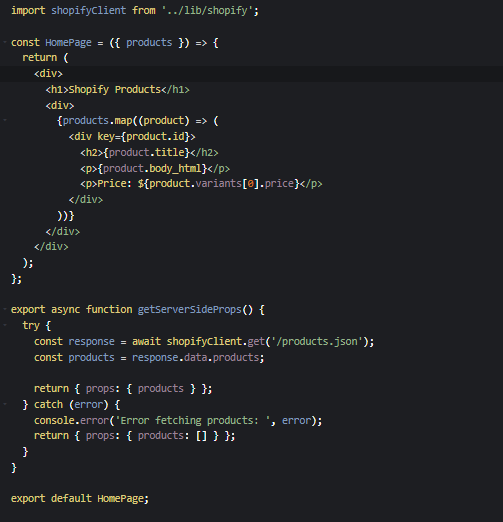
This code sets up the API client using your Shopify API credentials and defines the base URL for making requests.

## Step 6: Fetch Shopify Products

## 6.1 Server-Side Data Fetching with getServerSideProps

To fetch Shopify products, use Next.js’s getServerSideProps to retrieve data from the Shopify API on the server side.

File: pages/index.js

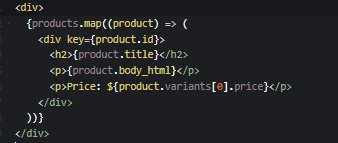


 getServerSideProps: Fetches the products from the Shopify API when the page is requested, passing the products data to the page component.

## Step 7: Display Products in Next.js

## 7.1 Render Products on the Homepage

Inside your component, map over the products and display them. The following code renders product details like the title, description, and price:



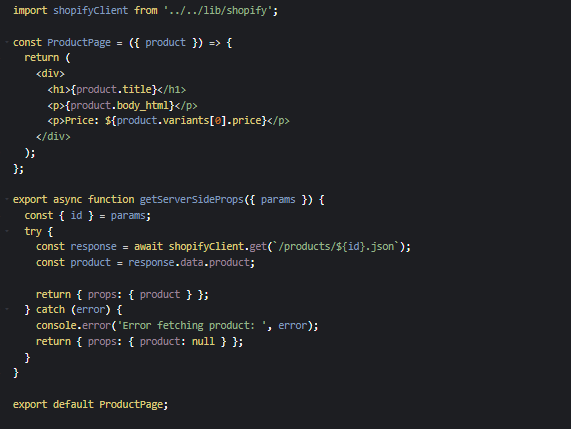
This code will display the Shopify products on the homepage of your Next.js app.

## Step 8: Handling Product Details Page

## 8.1 Create Dynamic Product Page

To display detailed product information, create a dynamic route using Next.js's dynamic routes.

File: pages/products/[id].js



· The getServerSideProps function fetches detailed product data based on the product ID from the URL.

· Dynamic routes allow you to fetch and display information for individual products.