Instructions on how to deploy on IBM Cloud.

1. Create an IBM Cloud account.
2. Choose IBM Cloud object storage
3. Install and configure your chosen e-commerce platform.
4. Add products and content, optimize performance, and test thoroughly.
5. Configure your domain and go live.
6. Maintain, update, and ensure compliance with regulations.

1. Create .env File

- duplicate .env.example in backend folder and rename it to .env

2. Setup MongoDB

- Local MongoDB
  - Install it from here
  - In .env file update MONGODB_URI=mongodb://localhost/auratraffic
- OR Atlas Cloud MongoDB
  - Create database at https://cloud.mongodb.com
  - In .env file update MONGODB_URI=mongodb+srv://auratraffic:auratraffic@aura.komktxw.mongodb.net/auratraffic?retryWrites=true&w=majority&appName=AtlasApp

3. Run Backend

```
$ cd backend
$ npm install
$ npm start
```

4. Run Frontend

```
# open new terminal
$ cd frontend
$ npm install
$ npm start
```

5. Seed Users and Products

- Run this on browser: http://localhost:5000/api/seed
- It returns admin email and password and 6 sample products

6. Admin Login

- Run http://localhost:3000/signin
- Enter admin email and password and click signin

Step By Step Process

1. **Create React App**
2. **List Products**
   i. create products array

ii. add product images
iii. render products
iv. style products

3. **Add page routing**
   i. npm i react-router-dom
   ii. create route for home screen
   iii. create router for product screen
4. **Create Node.JS Server**
   i. run npm init in root folder
   ii. Update package.json set type: module
   iii. Add .js to imports
   iv. npm install express
   v. create server.js
   vi. add start command as node backend/server.js
   vii. require express
   viii. create route for / return backend is ready.
   ix. move products.js from frontend to backend
   x. create route for /api/products
   xi. return products
   xii. run npm start
5. **Fetch Products From Backend**
   i. set proxy in package.json
   ii. npm install axios
   iii. use state hook
   iv. use effect hook
   v. use reducer hook
6. **Manage State By Reducer Hook**
   i. define reducer
   ii. update fetch data
   iii. get state from usReducer
7. **Add bootstrap UI Framework**
   i. npm install react-bootstrap bootstrap
   ii. update App.js
8. **Create Product and Rating Component**
   i. create Rating component
   ii. Create Product component
   iii. Use Rating component in Product component
9. **Create Product Details Screen**
   i. fetch product from backend
   ii. create 3 columns for image, info and action
10. **Create Loading and Message Component**
   i. create loading component
   ii. use spinner component

iii. craete message component
iv. create utils.js to define getError fuction

11. Create React Context For Add Item To Cart

i. Create React Context
ii. define reducer
iii. create store provider
iv. implement add to cart button click handler

12. Complete Add To Cart

i. check exist item in the cart
ii. check count in stock in backend

13. Create Cart Screen

i. create 2 columns
ii. display items list
iii. create action column

14. Complete Cart Screen

i. click handler for inc/dec item
ii. click handler for remove item
iii. click handler for checkout

15. Create Signin Screen

i. create sign in form
ii. add email and password
iii. add signin button

16. Connect To MongoDB Database

i. create atlas monogodb database
ii. install local mongodb database
iii. npm install mongoose
iv. connect to mongodb database

17. Seed Sample Products

i. create Product model
ii. create seed route
iii. use route in server.js
iv. seed sample product

18. Seed Sample Users

i. create user model
ii. seed sample users

19. Create Signin Backend API

i. create signin api
ii. npm install jsonwebtoken
iii. define generateToken

20. Complete Signin Screen

i. handle submit action
ii. save token in store and local storage
iii. show user name in header

21. Create Shipping Screen

i. create form inputs
ii. handle save shipping address
iii. add checkout wizard bar

22. Create Sign Up Screen

i. create input forms
ii. handle submit
iii. create backend api

23. Implement Select Payment Method Screen

i. create input forms
ii. handle submit

24. Create Place Order Screen

i. show cart items, payment and address
ii. calculate order summary

25. Implement Place Order Action

i. handle place order action
ii. create order create api

26. Create Order Screen

i. create backend api for order/:id
ii. fetch order api in frontend
iii. show order information in 2 cloumns

27. Pay Order By PayPal

i. generate paypal client id
ii. create api to return client id
iii. install react-paypal-js
iv. use PayPalScriptProvider in index.js
v. use usePayPalScriptReducer in Order Screen
vi. implement loadPaypalScript function
vii. render paypal button
viii. implement onApprove payment function
ix. create pay order api in backend

28. Display Order History

i. create order screen
ii. create order history api
iii. use api in the frontend