

**Cloud Computing 2024 Project [SC]**

# **Online Shopping Platform**

**Table of Contents:**

1. Project Overview
2. Architecture
3. Setup and Installation
4. Development Environment
5. Deployment
6. Scaling and High Availability
7. API Endpoints
8. **Project Overview:**

This project is an e-commerce platform designed to provide a seamless shopping experience for customers. The platform consists of multiple services handling various functionalities such as user authentication, product catalog management, order processing, and payment processing.

1. **Architecture:**

The Online Shopping Platform is designed to offer a seamless shopping experience through a collection of services. The key services and functionalities include:

1. **User Management:** Handles user authentication.
2. **Product Catalog Management:** Manages the catalog of products available for purchase.
3. **Cart Management:** Allows users to add, remove, and manage items in their shopping carts.
4. **Order Processing:** Handles the creation, modification, and fulfillment of customer orders.
5. **Payment Processing:** Facilitates secure payment transactions for completed orders.

These services are designed to be independent, loosely coupled, and communicate with each other via lightweight APIs, primarily using HTTP protocols.

1. **Setup and Installation:**

**Technologies Used**

The Online Shopping Platform utilizes the following technologies:

* **Docker:** Containerization of services for consistency and ease of deployment.
* **Docker-Compose:** Management of multi-container development, staging, and production environments.
* **Kubernetes:** Orchestration and efficient management of containerized services for scalability and reliability.

**Dependencies**

Ensure the following dependencies are installed:

* Docker Engine(dockerfile , dockerimage)
* Docker-Compose
* Kubernetes Cluster (for production deployment)

1. **Development Environment:**

Running Locally with Docker Compose

docker-compose up

Accessing the Application

* Frontend: <http://localhost:3000>
* Backend: <http://localhost:7249>

1. **Deployment:**

**Kubernetes Deployment**

1. Created Docker Images: Build our Docker images.
2. Kubernetes Manifests: Use the manifests provided earlier to deploy your services.

kubectl apply -f deployment.yml

1. Verify Deployment:

kubectl get deployments

kubectl get pods

**Load Balancing and Fault Tolerance**

Kubernetes services with type ‘LoadBalancer’ will distribute traffic across multiple replicas, ensuring load balancing and fault tolerance.

1. **Scaling and High Availability:**

We used the Horizontal Pod Autoscaler (HPA) to automatically scale your services based on CPU utilization.

1. **API Endpoints:**

Items Service

* GET /api/ Items/view: Retrieve all products
* Post /api/ Items/add : Add new items
* Puts /api/Items/update{id} : update a single items using id
* Delete /api/Items/delete{id} : delete a single items using id

Order Service

* Get /api/order/create order: Create a new order by initialize new id.
* GET /api/orders/get my order{id} : Retrieve an order by ID.
* GET /api/orders/get all order : view all order
* Post /api/order/make order{id} : make a new order by using id and select items you want
* Delete api/order/delete order{id} : cancel order and delete all items in this order
* Delete api/order/delete items from order{id} : delete a single item

From this order by using there ids (id.order,id.item)

Payment Service

* post /api/pay/get {id}: pay your items by using id.order and info of visa

User

* GET /api/user/login : take mail & password for user to

verification accepted

* Post /api/user/register : sign up for new user

A computer server and a person

Description automatically generated

Responce



[This Photo](https://andrejgajdos.com/how-to-hire-a-freelance-front-end-developer/) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

Collect Data

Request

A metal cylinder with a globe inside

Description automatically generated

Backend

Frontend

Display Results