National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Anish Yadav Pinjarla
Student ID: x23216093

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
Msc in Cloud Computing

Student Name:	Anish Yadav Pinjarla
Student ID:	x23216093
Programme:	MSc in Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	02/01/2026
Project Title:	Configuration Manual
Word Count:	2500
Page Count:	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Anish Yadav Pinjarla
Date:	28th January 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anish Yadav Pinjarla
x23216093

1 Introduction

This Configuration Manual provides all the necessary steps to configure and deploy the Containerized E-Commerce Microservices Platform. This implementation is a production-read e-commerce microservices solution that has a React based frontend, Node.js backend microservices, complete Admin Dashboard, all running in Docker containers and orchestrated using AWS ECS Fargate.

1.1 Purpose

This document is intended to guide system administrators and developers through the procedure of:

- Setting up the local development environment
- Install requisite dependencies
- Environment variable configuration for various services
- Deploying infrastructure and services on Amazon Web Services - AWS
- Deployment verification and access to the application

1.2 System Requirements

Before proceeding, ensure the following hardware and software requirements are met.

1.2.1 Hardware Requirements

- **CPU**: Dual-core processor or better (Apple Silicon or Intel/AMD).
- **RAM**: Minimum 8GB (16GB recommended for running all microservices locally).
- **Storage**: At least 10GB of free disk space for Docker images and containers.

1.2.2 Software Requirements

- Operating System: macOS (tested), Linux (Ubuntu/Debian), or
- Windows 10/11 (with WSL2).
- Docker Desktop must be version 4.0 or greater.
- Node.js: Version 20.x or above.
- AWS CLI: version 2.x, configured with appropriate credentials.
- Git: Used for version control.

2 Environment Setup

This section details the installation of prerequisite tools required to run and deploy the application.

2.1 Installing Node.js

The project relies on Node.js (v20+) for both frontend and backend services.

1. Download the official installer from https://nodejs.org/.
2. Run the installer and follow the on-screen instructions.
3. Verify the installation by running:

```
node --version
npm --version
```

2.2 Installing Docker

Docker is essential for containerizing the microservices.

1. Download Docker Desktop for your OS from https://www.docker.com/products/docker-desktop.
2. Install and start Docker Desktop.
3. Verify that the Docker engine is running:

```
docker --version
docker-compose --version
```

2.3 Installing AWS CLI

To interact with AWS services and deploy the infrastructure:

1. Follow the official AWS guide to install the CLI: https://aws.amazon.com/cli/.
2. Configure your credentials:

```
aws configure
```

3. Enter your Access Key ID, Secret Access Key, Default region (e.g., us-east-1), and Output format (json).

3 Installation

3.1 Cloning the Repository

Start by cloning the project repository to your local machine:

```
git clone <repository-url>
cd e-commerce
```

3.2 Project Structure

The project assumes the following structure (verified from README.md):

- services/: Contains source code for all microservices (product-catalog, user-service, etc.).
- frontend/: React-based frontend application.
- infrastructure/: CloudFormation templates.
- scripts/: Utility scripts for build and database initialization.
- docker-compose.yml: For local orchestration.

3.3 Environment Configuration

Each service requires specific environment variables. Navigate to each service directory (e.g., services/product-catalog) and create a .env file based on the provided .env.example.

Example configuration for product-catalog/.env:

```
PORT=3001
NODE_ENV=development
DB_HOST=localhost
DB_PORT=5432
DB_NAME=products_db
DB_USER=postgres
DB_PASSWORD=postgres
```

Repeat this process for:

- user-service
- order-service
- payment-service
- notification-service

4 Running the Application

4.1 Local Development with Docker Compose

To start the entire platform locally by utilizing the docker-compose.yml file:

1. From the root directory in the terminal, run:

   ```
   docker-compose up -d
   ```

   This command builds the images and starts the containers in detached mode.

2. Verify the status of the containers:

   ```
   docker-compose ps
   ```

   Ensure all services (frontend, product, user, order, payment, notification, postgres) are in the Up state.

4.2 Accessing the Application

Once the containers are running,we can access the various components via your browser:

- **Frontend Storefront**: http://localhost:3000 - The main shopping interface for users.
- **Admin Dashboard**: http://localhost:3010 - For managing products and orders.
- **API Gateway / Services**:
  - Product Service: http://localhost:3001
  - User Service: http://localhost:3002
  - Order Service: http://localhost:3003
- **Mailhog (Email Testing)**: http://localhost:8025 - To view emails sent by the notification service.

4.3 Stopping the Application

To stop and remove the containers in terminal run:

```
docker-compose down
```

5 AWS Deployment

This section outlines the steps to deploy the platform to Amazon Web Services using CloudFormation.

5.1 Infrastructure Deployment

The infrastructure/cloudformation/main.yaml file initializes the VPC, RDS(PostgreSQL), ECR repos, SNS topics, and SQS queues.

1. Deploy the network and data stack:

```
aws cloudformation deploy \
  --template-file infrastructure/cloudformation/main.yaml \
  --stack-name dev-ecommerce-infrastructure \
  --parameter-overrides Environment=dev DBPassword=YourSecurePassword \
  --capabilities CAPABILITY_NAMED_IAM
```

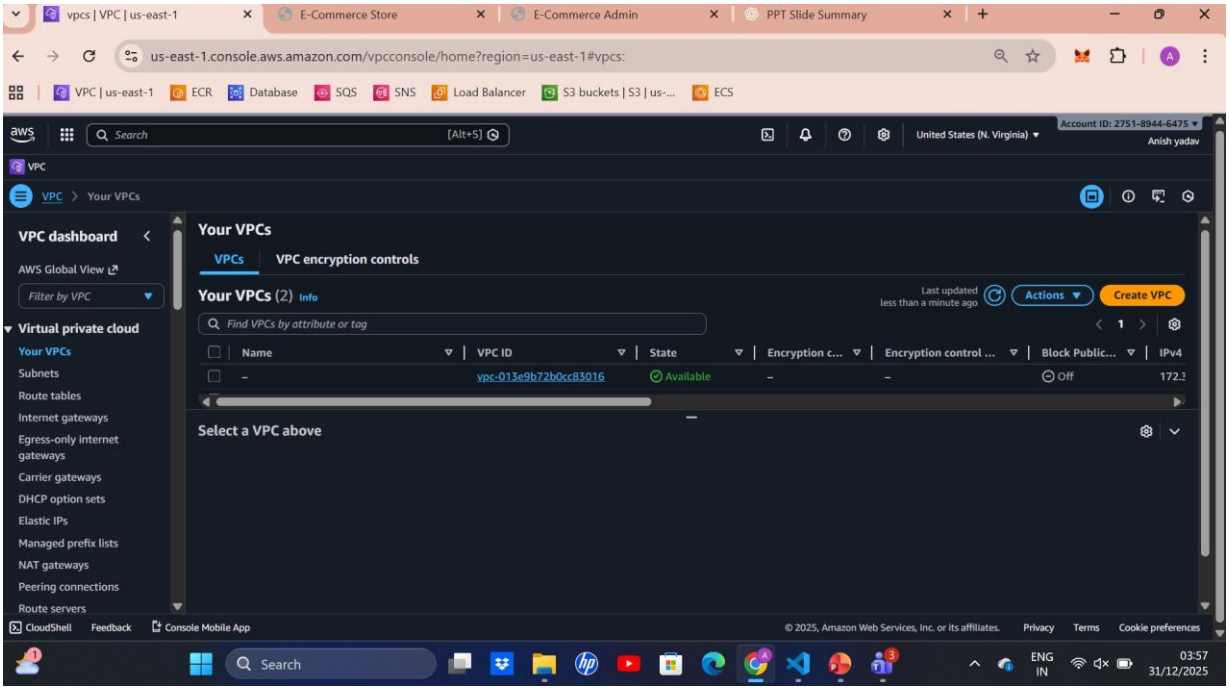

Figure 1: AWS CloudFormation Console showing successful stack deployment.

5.2 Building and Pushing Images

Use the interpreter script to generate and upload the Docker images into the ECR repository set up in step 1.

./scripts/build-and-push.sh

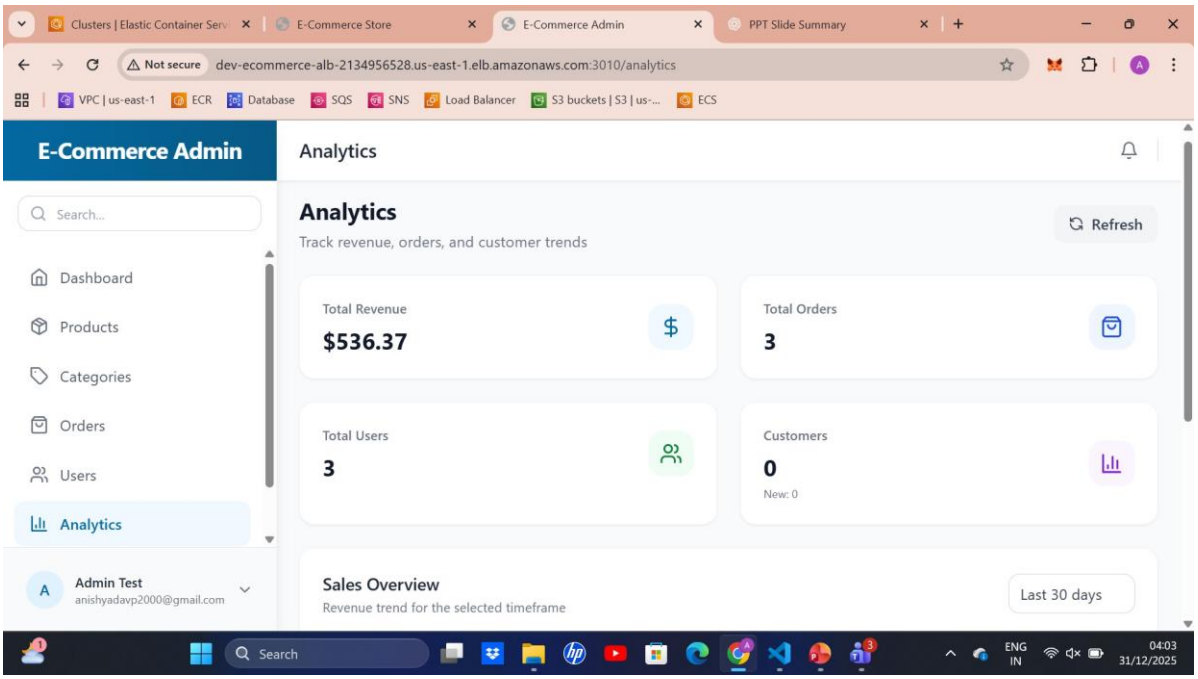

Figure 2: Amazon ECR Repositories listing the uploaded service images.

5.3 Service Deployment (ECS Fargate)

Deploy the ECS Cluster and Services using the ecs-cluster.yaml template.

```
aws cloudformation deploy \
  --template-file infrastructure/cloudformation/ecs-cluster.yaml \
  --stack-name dev-ecommerce-ecs \
  --parameter-overrides Environment=dev \
    DBEndpoint=<RDS_ENDPOINT_FROM_STEP_1> \
    DBPassword=YourSecurePassword \
  --capabilities CAPABILITY_NAMED_IAM
```

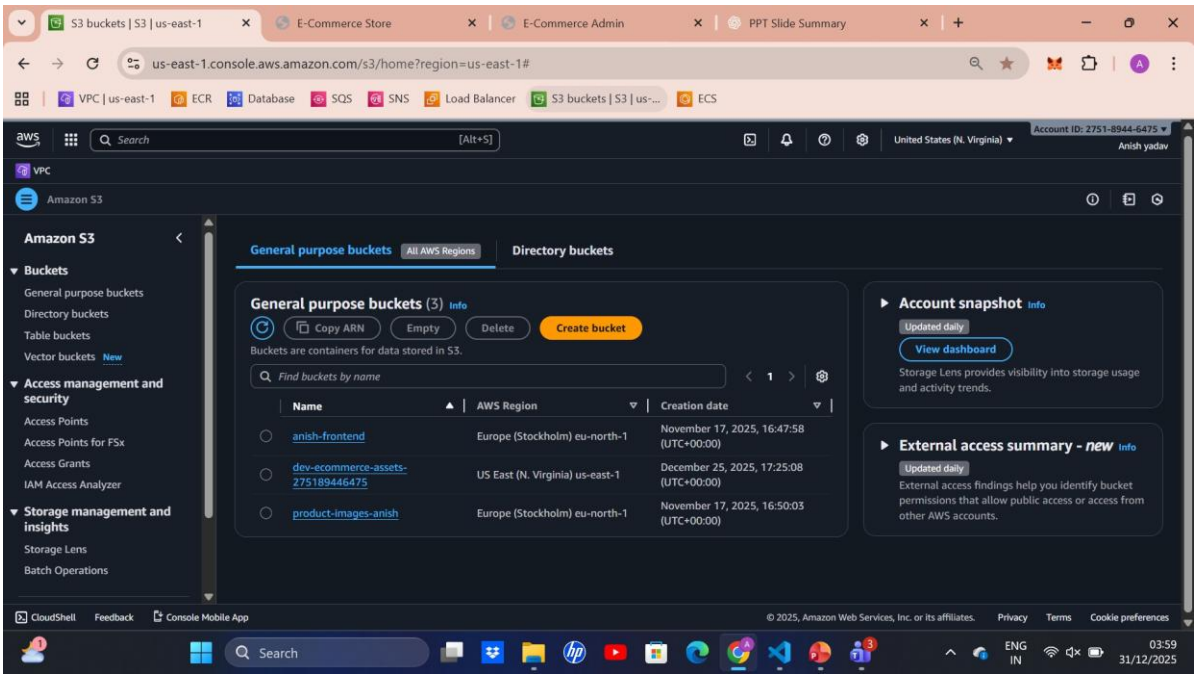

Figure 3: AWS ECS Cluster overview showing running services.

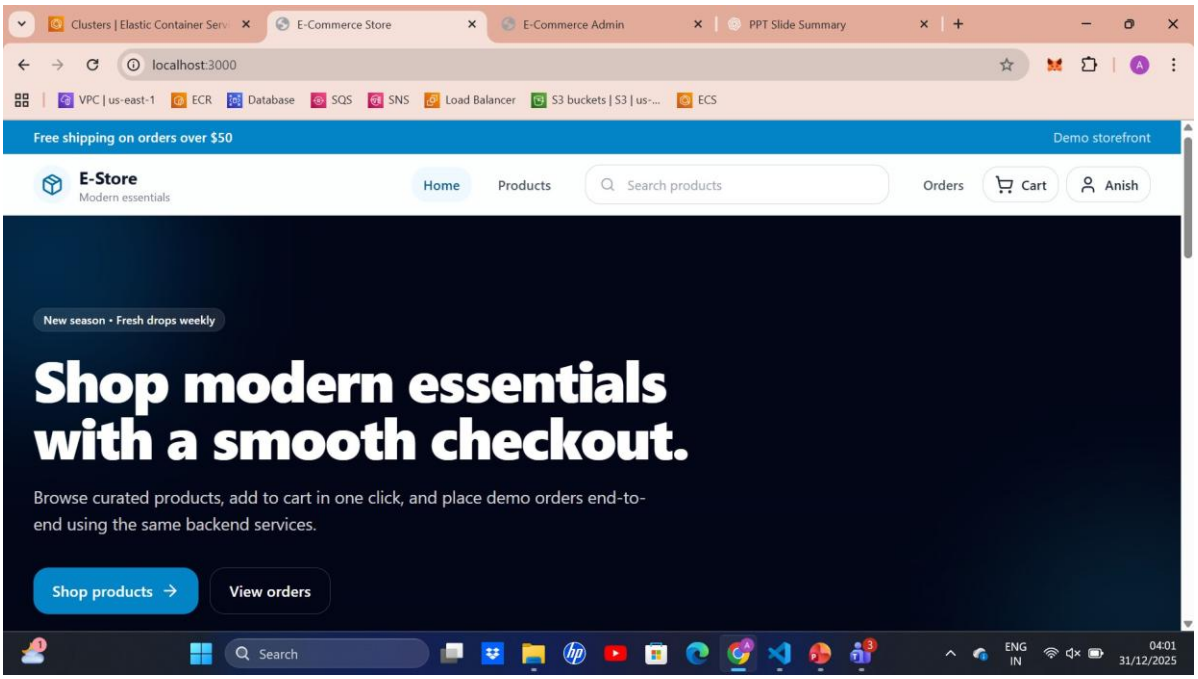

Figure 4: ECS Service details displaying the Task Definition configuration.