**BITS WILP – FSE OCT 23**

**C4 – Cloud Native Development**

Assignment 2

Building microservices using Docker

Table of Contents

[Problem Statement: 3](#_Toc171390586)

[Solution: 4](#_Toc171390587)

[Source Code-Project structures 5](#_Toc171390588)

[Docker Build and Run status 7](#_Toc171390589)

[Postman Test – Screenshots 9](#_Toc171390590)

# Problem Statement:

Arbees Grocer is a cloud-based online grocer that provides customized delivery of groceries. Their business model is targeted towards those days where a person needs groceries but is unable to go the physical store to shop.

Arbees Grocer (AG) supports three different ordering options. A customer can:

1. Place an order with one of the approved grocery stores, send the item list, and order confirmation to AG via a web interface or a mobile application.
2. Send a grocery list to AG via a web interface or a mobile application.
3. Select from AG grocery items list and place an order.

Create the following microservices using any programming language you are comfortable with (python/java)

Once you have created the services, these needs to be containerized using dockers.

***Note****:*

The services should be able to accept a dummy payload. You can provide static message like Order Received, Order Confirmed, Delivery Complete etc.

* webOrder
* orderTracking
* orderConfirmation
* deliveryConfirmation

# Solution:

As mentioned in the problem statement, there are three different ways to place an order. Hence every order in AG platform must be categorized into one of below **Order Type**.

* PARN *– Partner Grocery Store Order*
* ONLW *– Web order with products from the list available on website*
* ONLL *– Web order using custom product list uploaded via Web/Mobile app.*

The services that need to be created are assumed to have the below functionality:

* **webOrder** – *Service for Web Orders with custom product list or product list generated from the website catalogue (OrderType: ONLW, ONLL)*

This service will accept the following parameters 🡪 customerId, orderType, items[itemName, itemQuantity, itemUOM(unit of measure), itemDescription]

And the service will provide the following output parameters 🡪 orderId, orderStatus, message

* **orderTracking** – *Service to track the progress of an order irrespective of the order type.*

This service will accept the following parameters 🡪 orderId, orderType

And the service will provide the following output parameters 🡪 orderId, orderStatus, message

* **orderConfirmation** – *Service to cater to the order which is placed with a partner grocery store (OrderType: PARN).*

This service will accept the following parameters 🡪 customerId, storeId, partnerOrderId, orderStaus

And the service will provide the following output parameters 🡪 orderId, orderStatus, message

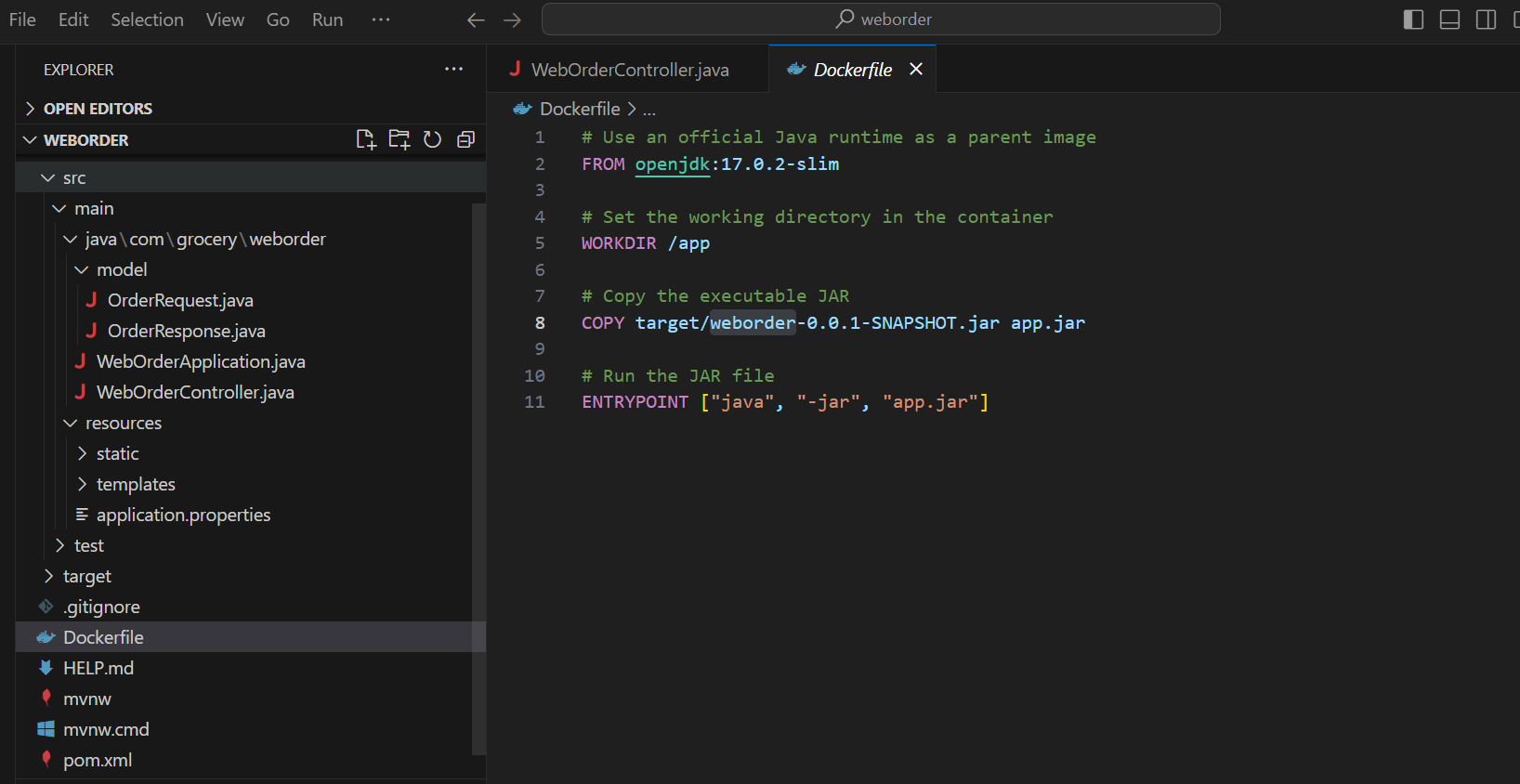
* **deliveryConfirmation** – *Service for the delivery partner to upload a picture and update the final delivery status.*

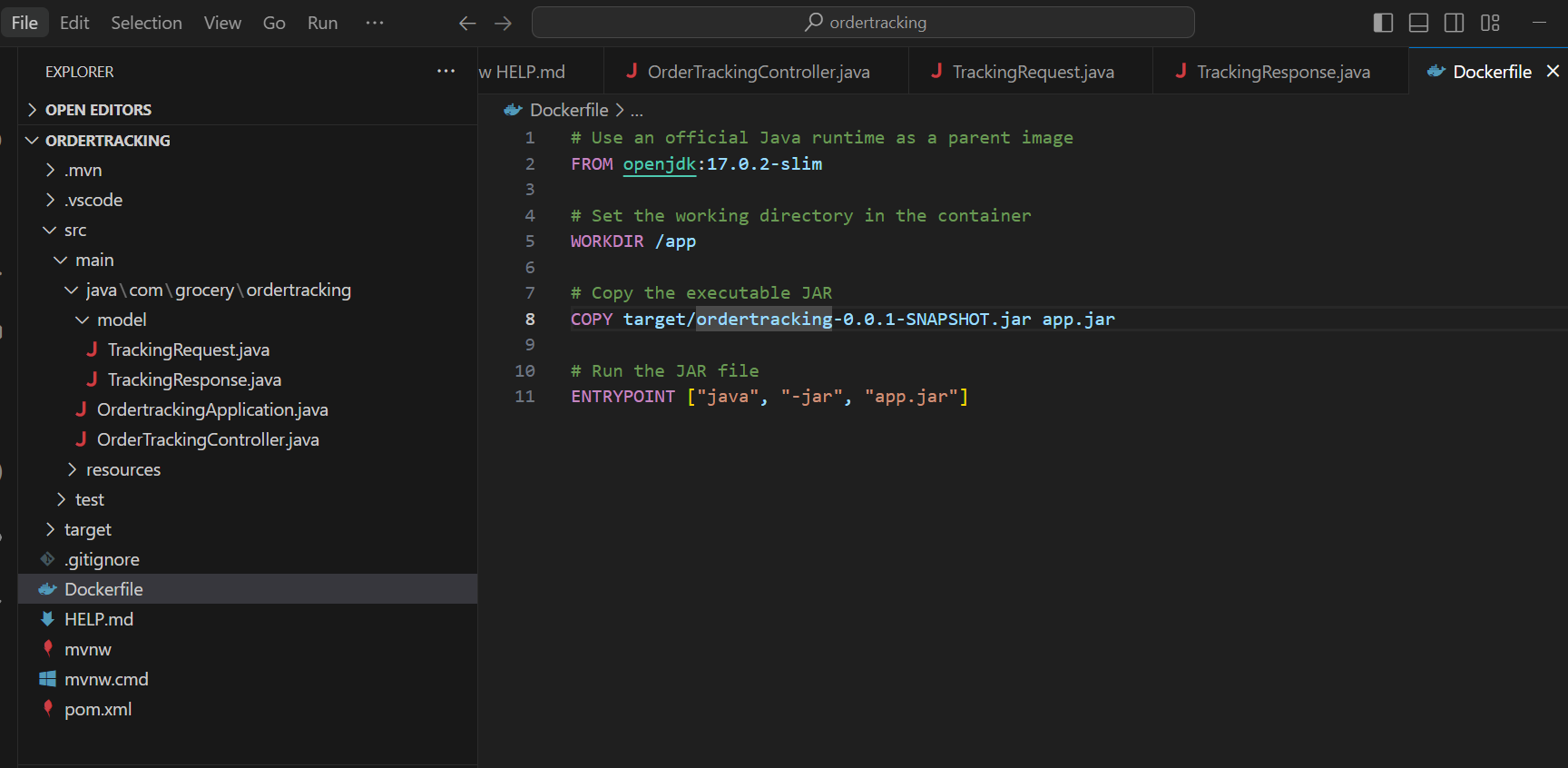
This service will accept the following parameters 🡪 orderId, deliveryStatus

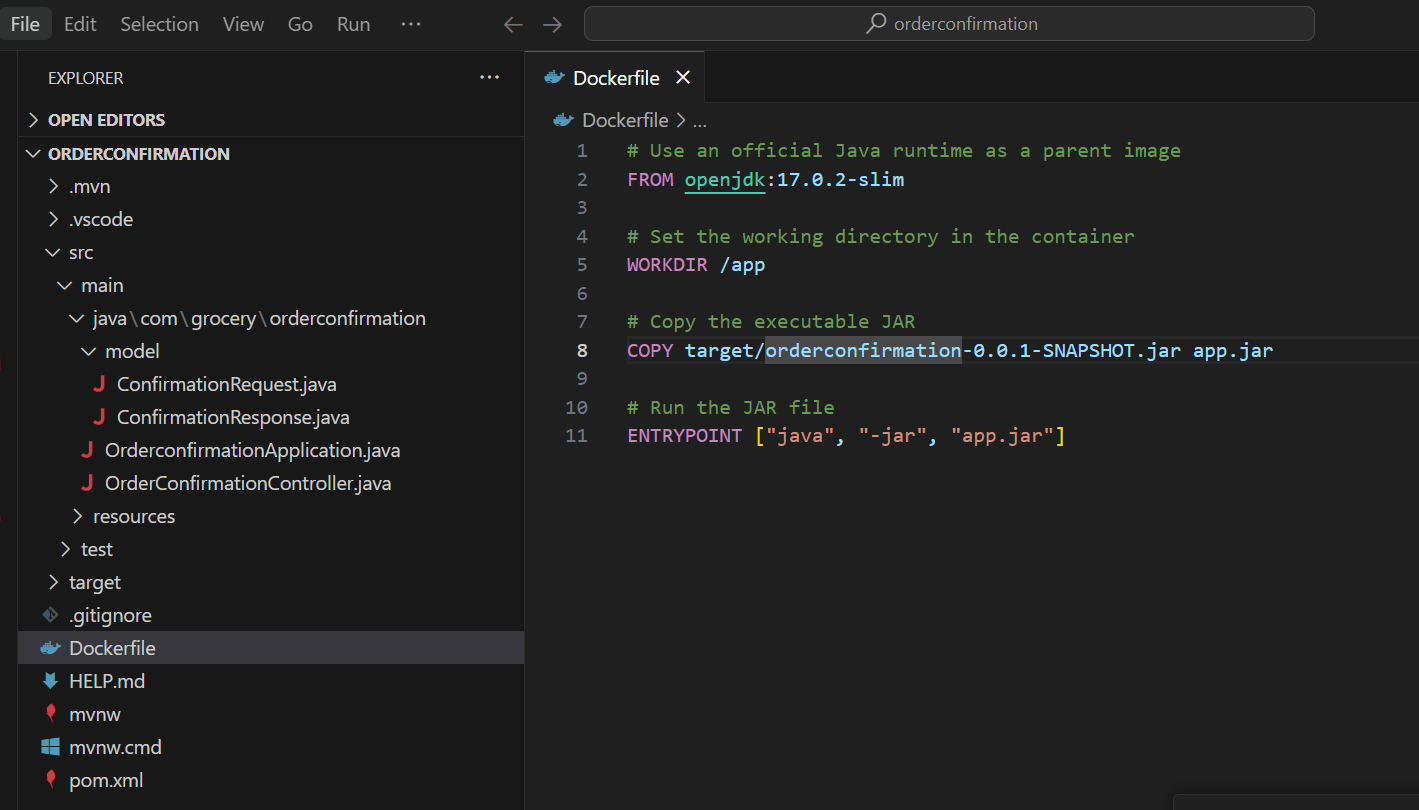
And the service will provide the following output parameters 🡪 orderId, orderStatus, message

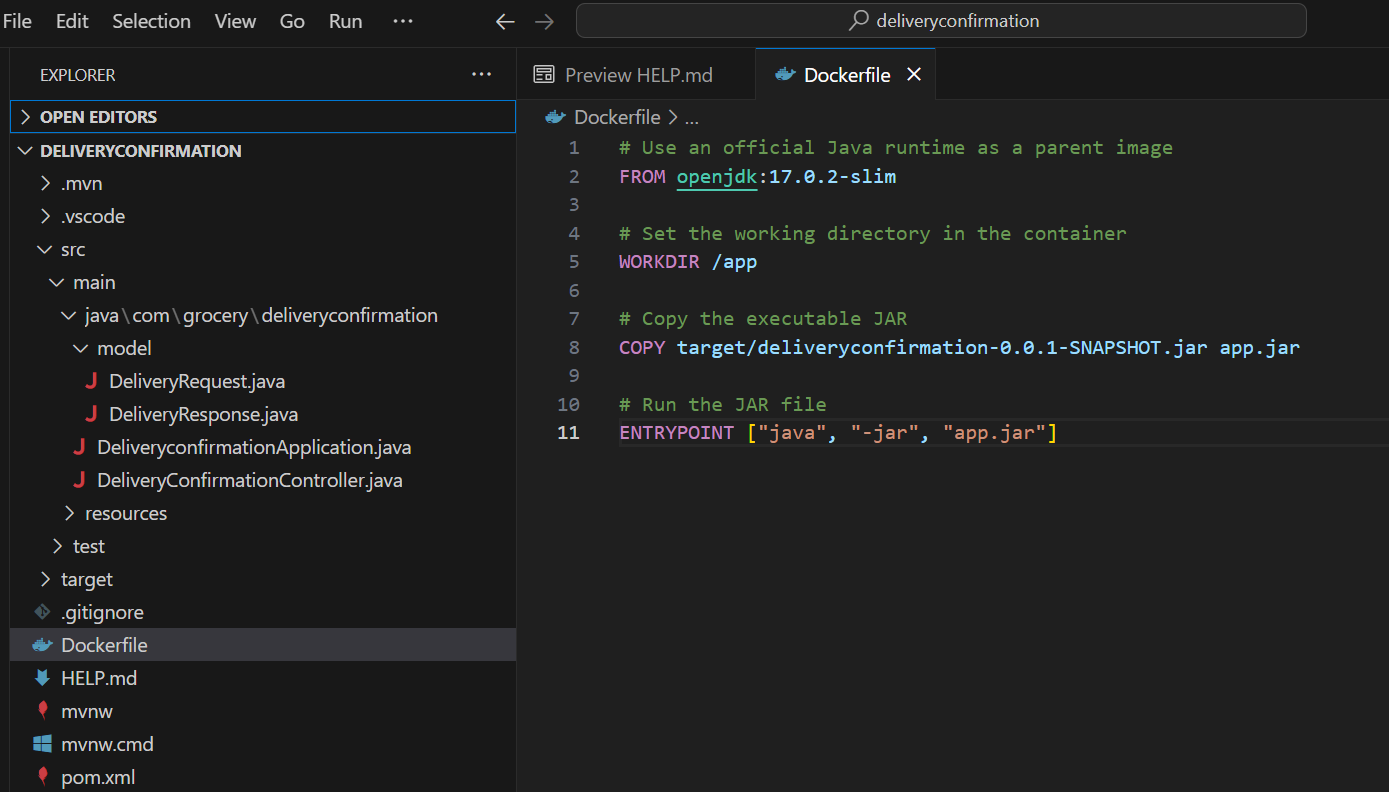
# Source Code-Project structures

The respective services are created using Java SpringBoot and the screenshots of the respective project structure, this is for reference only. The source code is also provided along with this document.

**WEBORDER:  
**

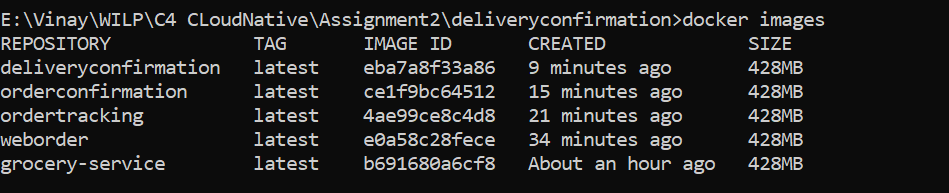
**ORDER TRACKING:  
**

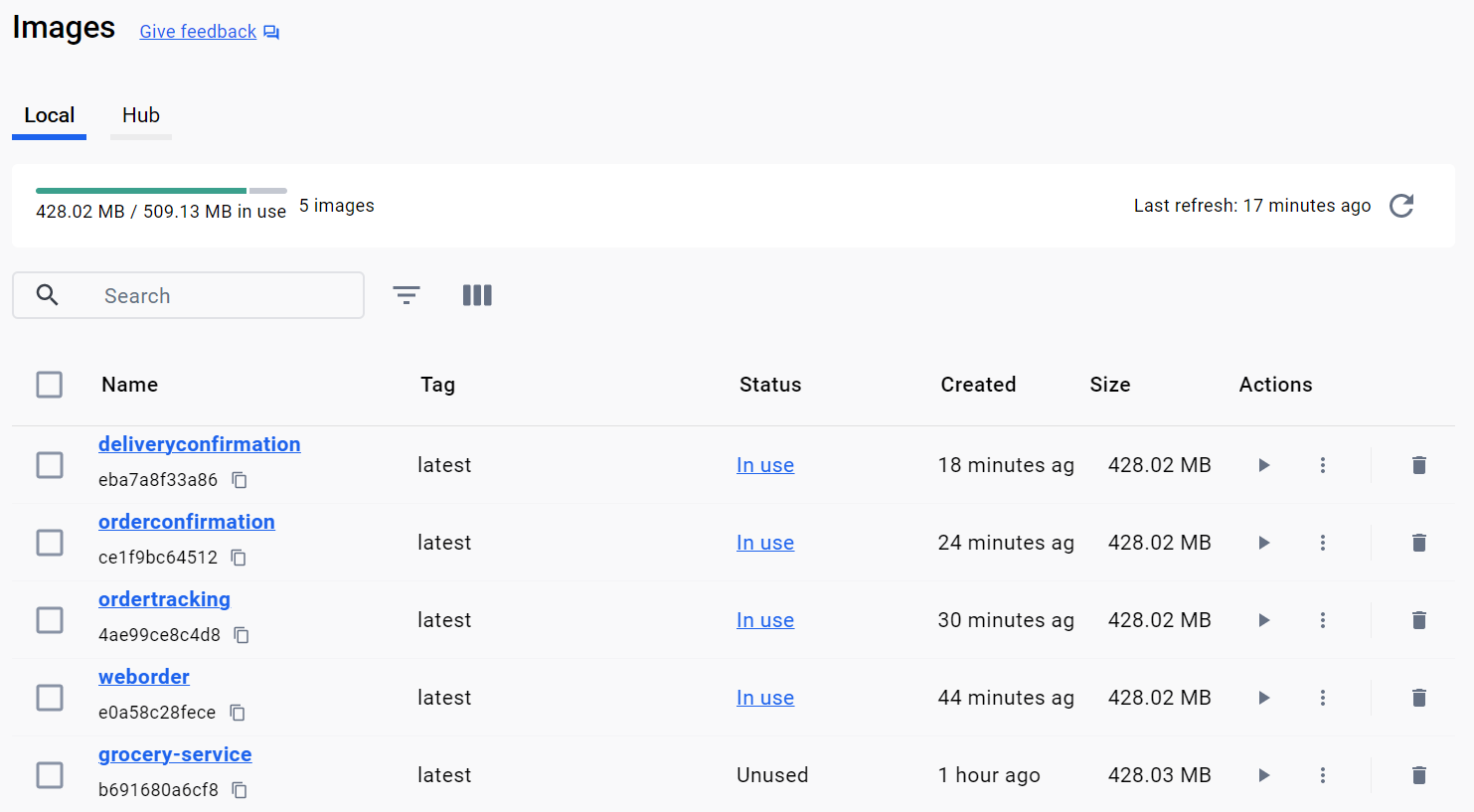
**ORDER CONFIRMATION:  
**

**DELIVERY CONFIRMATION**:

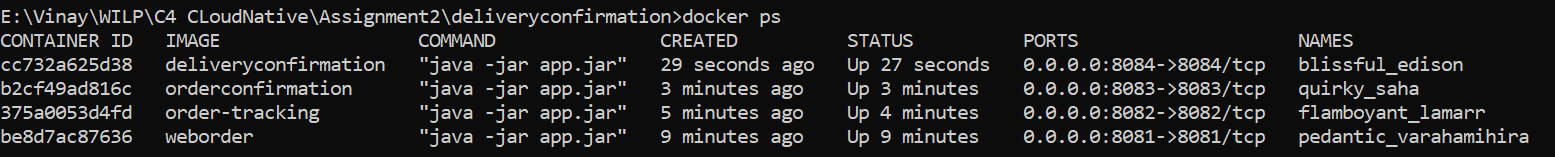
# Docker Build and Run status

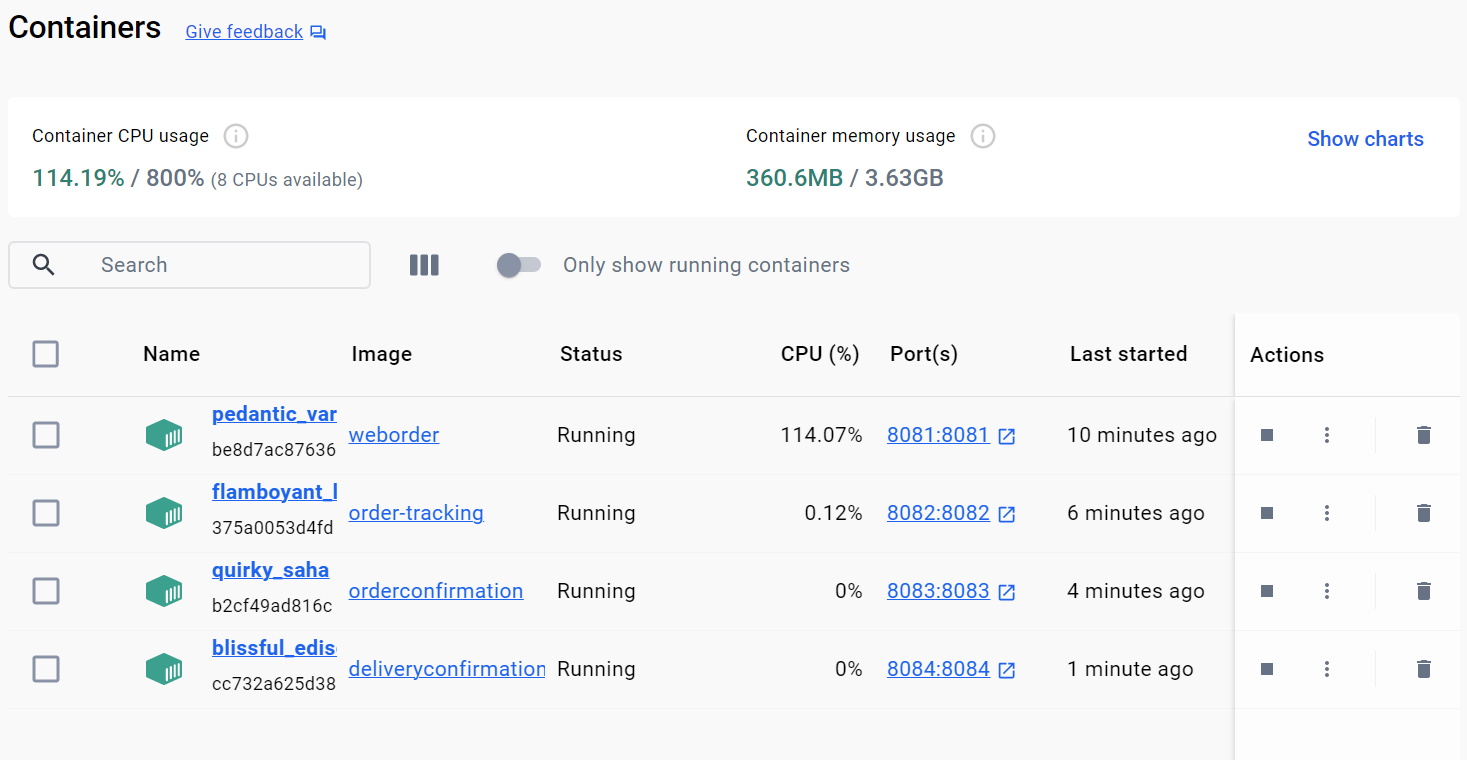
Docker file has been created for each of these services in the respective project structure and docker build has been executed to created their images. Please find below screenshot for reference:



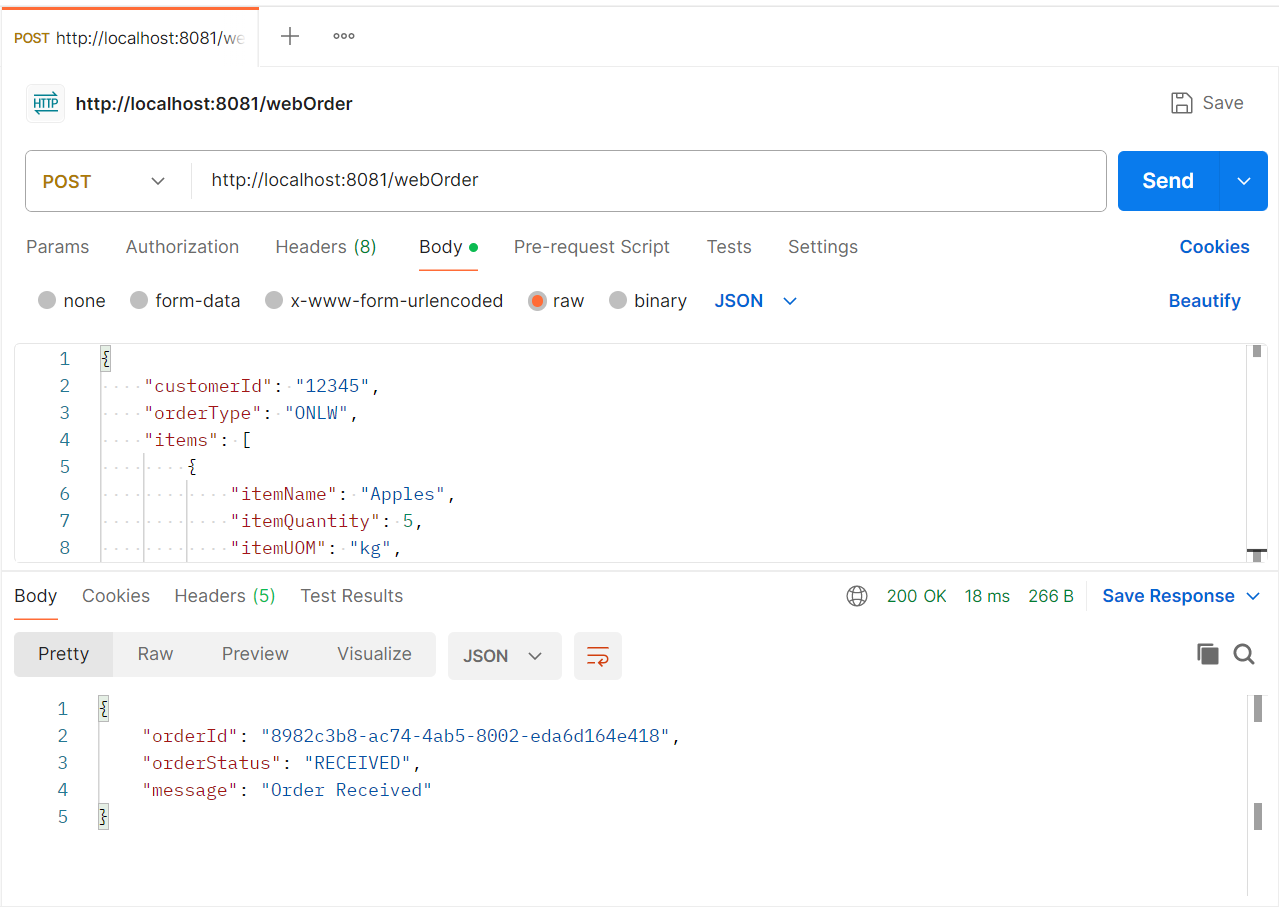
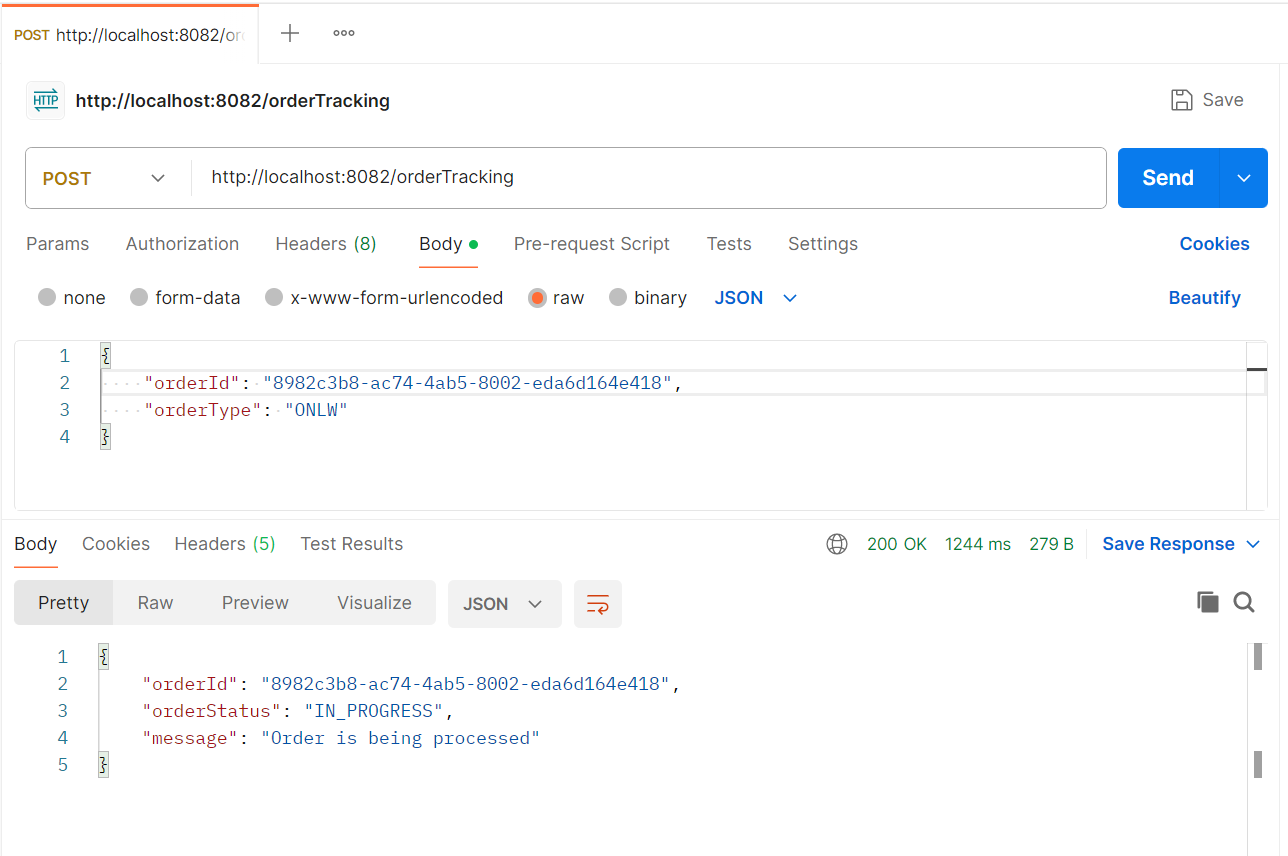
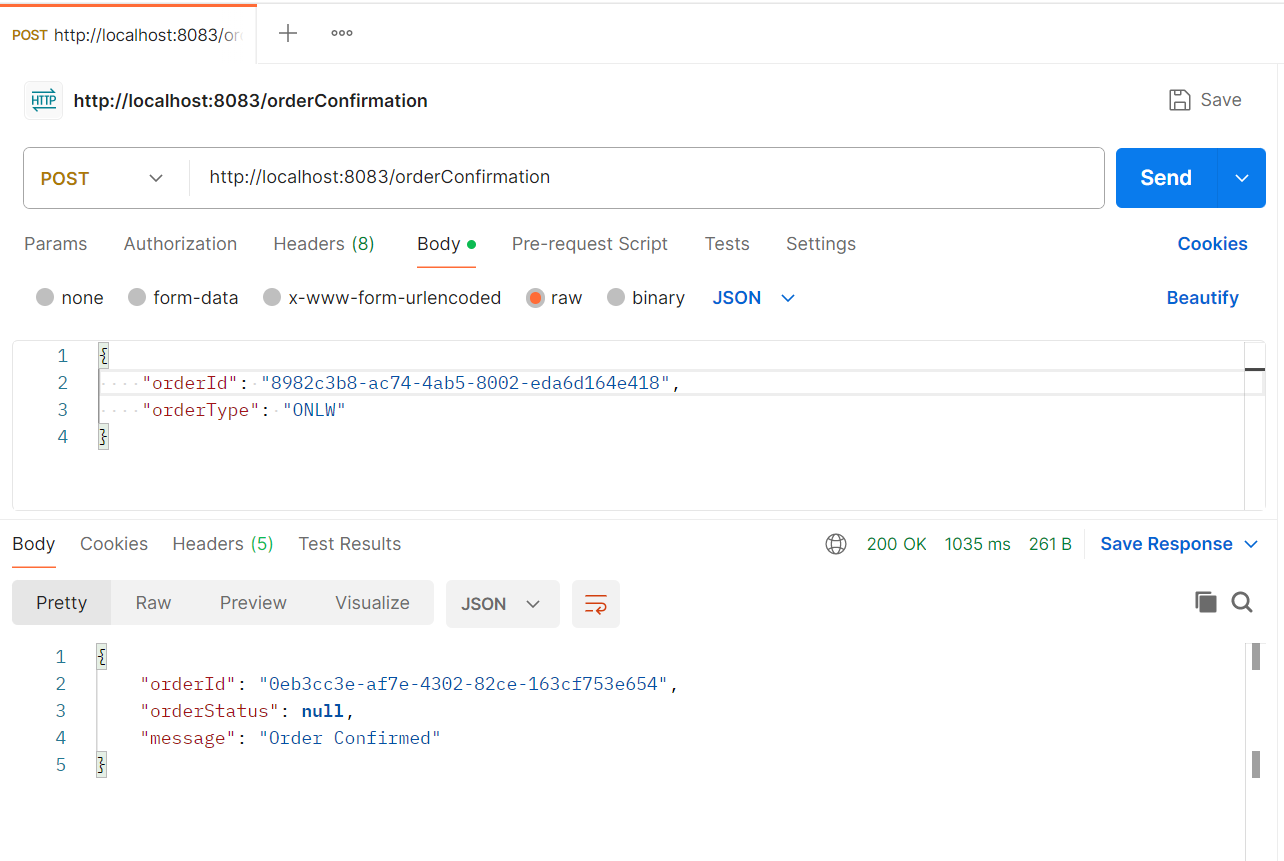


The docker images are then run as containers using the docker command and below are the respective status:





# Postman Test – Screenshots

1. WebOrder:  
   
2. OrderTracking:   
   
3. OrderConfirmation:  
   
4. DeliveryConfirmation:  
   