**E-Commerce-Microservice-with-NodeJS-and-RabbitMQ**

Three Services we created for this

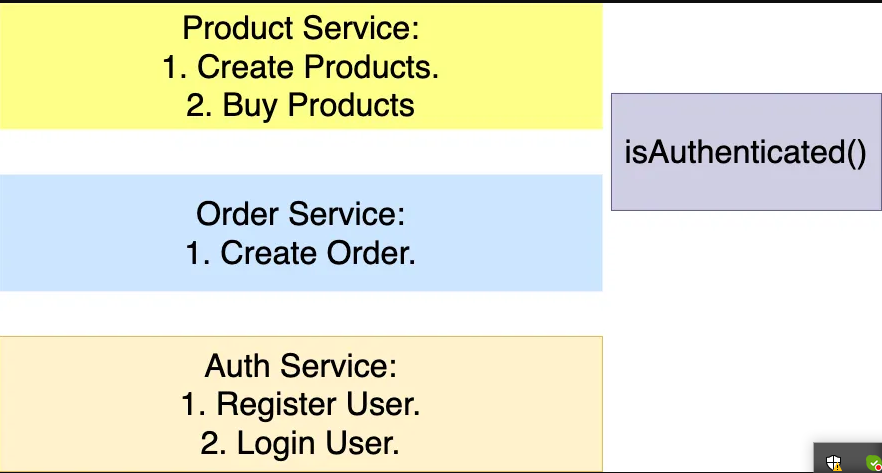
**Services:**

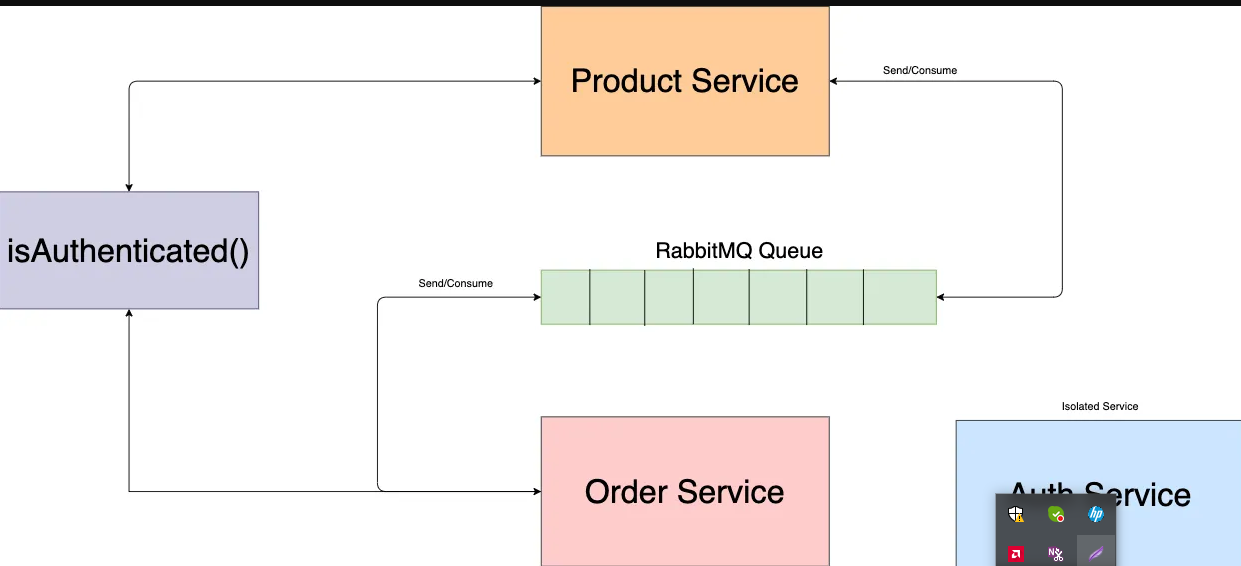
Auth Service

Product Service

Order Service

We using RabbitMq microservices pattern for this approach and communicating as a messaging server using docker





Firstly we install docker desktop in our Machine then we create three folder one product service, order service and auth service. Auth service is just for authenticate the if user is logged in or not to create and buy product and orders.

Then we install

npm i express nodemon amqplib

what is amqplib

amqplib is a library for interacting with RabbitMQ in Node.js. RabbitMQ is a message broker that allows different parts of a distributed system to communicate with each other by sending and receiving messages.

The amqplib library is a popular choice for Node.js developers who want to work with RabbitMQ. It provides an API for creating connections to RabbitMQ servers, declaring and managing exchanges and queues, and publishing and consuming messages. It is designed to be used with the AMQP (Advanced Message Queuing Protocol) protocol, which RabbitMQ supports.

Then in root project cmd where all three project here enter cmd and type this for running rabbitmq microservice using docker

**docker run -p 5672:5672 rabbitmq this 5672 assign in index.js file in all service check the code**

Then we go each project folder and and run with different port and firstly login add token in authorization and create product and then buy product

When you buy a product and hit the url in postman then rabbit channel in product service send to queue in order service and order service consume new order then order will create and they send to queue in product service and product consume the new order you can check in code here.

**What is Docker ?**

Certainly! Docker is a platform that simplifies the process of developing, shipping, and running applications. It uses containerization, a lightweight and portable way to package and isolate software and its dependencies.

Here's a simple breakdown:

**Containerization**: Imagine a container as a standalone, executable package that includes everything needed to run a piece of software, such as the code, runtime, libraries, and system tools. Containers ensure consistency across different environments, making it easy to move and deploy applications.

**Docker Image**: This is like a blueprint or template for a container. It contains all the necessary files and configurations to create a container. Think of it as a snapshot of an application and its environment at a specific point in time.

**Docker Engine**: The core of Docker. It's a lightweight runtime and set of tools that manage containers. It can run on any machine that supports containerization.

**Dockerfile**: This is a set of instructions that tells Docker how to build a Docker image. It specifies what goes into the image, such as the base operating system, application code, dependencies, and configurations.

**Docker Hub**: A centralized repository where you can find and share Docker images. It's like an app store for Docker containers.

In a nutshell, Docker allows developers to package their applications and all dependencies into containers, making it easy to deploy and run the same application consistently across different environments. It simplifies the development and deployment process, making applications more portable and scalable.