# Microservices Interview Crash Course - .NET Core + React

## Final Architecture Overview

This is a modular microservices-based architecture built using .NET Core and React:

1. Product.API - Handles product CRUD, has its own database.
2. Order.API - Manages orders, communicates with Product.API via REST/gRPC, uses RabbitMQ to publish events.
3. Gateway (Ocelot) - Unified entry point. Routes and authenticates requests.
4. RabbitMQ - Used for asynchronous communication (event-driven pattern).
5. Docker - All services are containerized and orchestrated via docker-compose.

Authentication is handled using JWT and validated at the API Gateway level.

Services communicate via REST and gRPC (e.g., Product.API for high performance).

## Revision Sheet

| Topic | Key Points |
|---|---|
| **API Gateway (Ocelot)** | **Central entry point, routes to microservices, supports JWT auth** |
| **JWT Authentication** | **Token-based auth; validated at gateway using DelegatingHandler** |
| **Polly** | **Resilience library for retries, circuit breakers in HTTP/gRPC calls** |
| **RabbitMQ** | **Event-driven communication, publisher-subscriber pattern** |
| **gRPC** | **High performance binary RPC, used between internal services** |
| **CQRS + MediatR** | **Separation of commands and queries, clean architecture** |
| **Docker** | **Containerization of services, docker-compose for orchestration** |
| **Swagger** | **API documentation for each microservice** |