**Lesson01 Introduction**

**Notes: -**

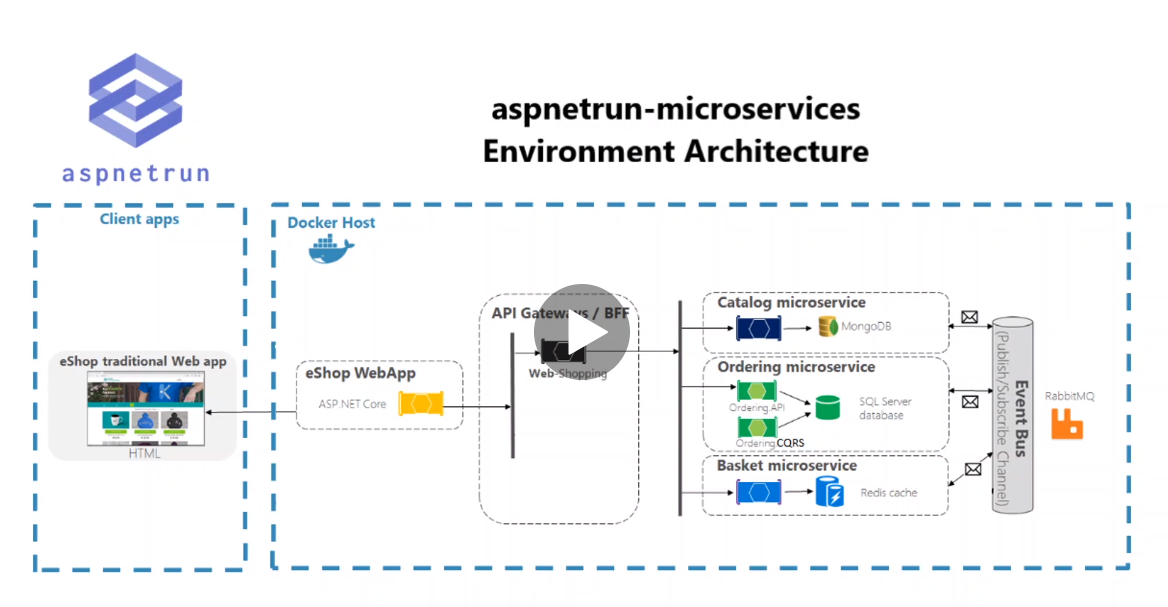
**1-the microservice architecture as below contains the following:-**

**A-SPA**

**B-API Gateway: which will use ocelot to unify all the microservices port and URL’s and redirect to the target microservice**

**C-databases: like MongoDB, SQL server, Redis cache**

**D-using RabbitMQ to communicate between microservices**



**2-the requirements for development environment are below:-**

**A-.Net core 3.x and above**

**B-VS 2019 version 16.0 or above**

**C-Docker desktop**

**D-Postman**

**D-windows 10 64 bit**

**Running the Microservice Solution**

**Steps:-**

**1-on the below URL** [GitHub - aspnetrun/run-aspnetcore-microservices: Microservices on .Net platforms which used Asp.Net Web API, Docker, RabbitMQ, Ocelot API Gateway, MongoDB, Redis, SqlServer, Entity Framework Core, CQRS and Clean Architecture implementation. See Microservices Architecture and Step by Step Implementation on .NET Course w/ discount->](https://github.com/aspnetrun/run-aspnetcore-microservices)

**Download the Microservice solution**

**2-the docker compose file is including what images we should be download form docker hub and how we organize these microservices inside of this compose file.**

**(so, in order to run all microservices we must run the docker compose command)**

**(we see that the docker-compose contains all the images that will be downloaded such as radius, mongo, SQL-Server, and the local web API plus the web app)**

**3-In the docker compose.yml files it contains all the images and the path it will be taken either from online like Mongo , Redius , etc. Or with the local Web API or web app**

**Commands: -**

**1-on command prompt go the folder that contains the docker compose**

**C:\Users\m.enbeh\Desktop\MicroService Architecture on .Net 5\Section01 Introduction\Files\run-aspnetcore-microservices-master**

**2-run the below command to download 8 images as below  
Redius**

**Sql server**

**RabbitMQ**

**Mongo**

**API Gateway**

**Basket API**

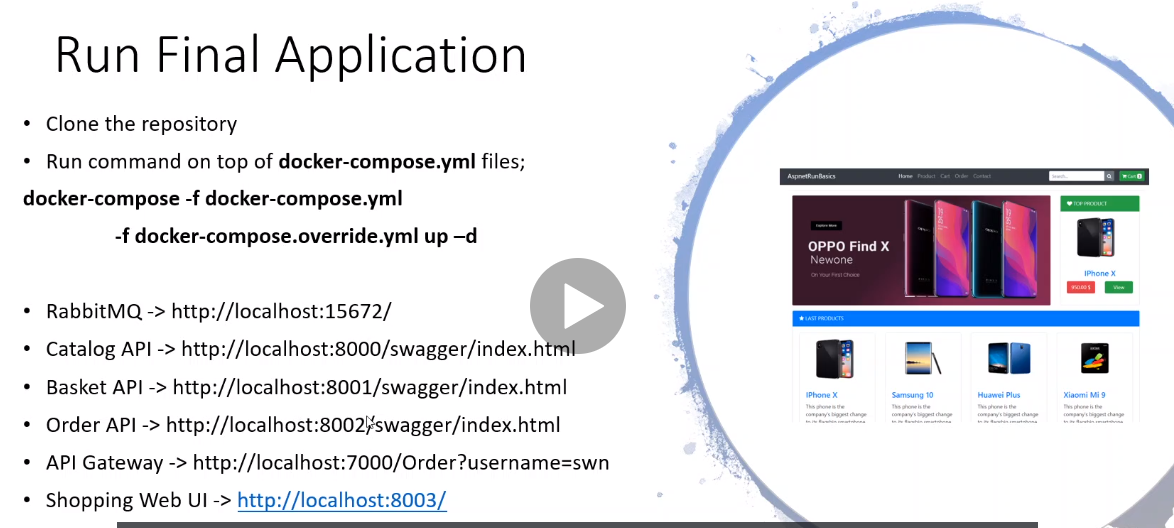
**Catalog API**

**Ordering API**

**Web App**

**docker-compose -f docker-compose.yml -f docker-compose.override.yml up -d**

**//it will take the docker compose.yml that contains the docker containers that will**



**Result: -**

**Swagger API :** [**http://localhost:8000/swagger/index.html**](http://localhost:8000/swagger/index.html)

**Basket API :** [**http://localhost:8001/swagger/index.html**](http://localhost:8001/swagger/index.html)

**Ordering API :** [**http://localhost:8002/swagger/index.html**](http://localhost:8002/swagger/index.html)

**API Gateway : http://localhost:7000/Order?username=swn**

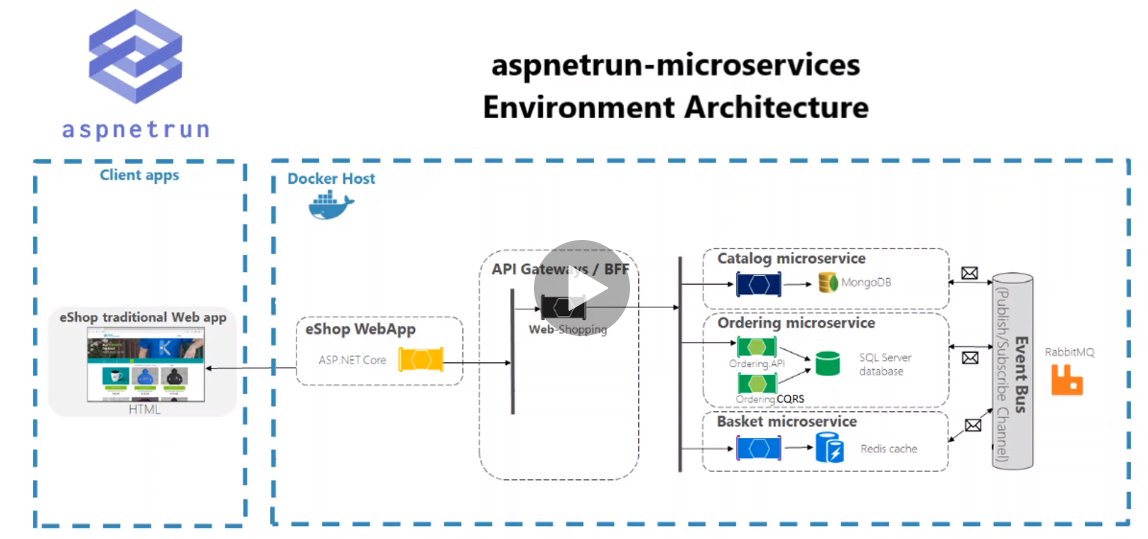
**RabbitMQ :** [**RabbitMQ Management**](http://localhost:15672/)

**WebAPP : http://localhost:8003/**

**The Operation as following: -**

**1-when you open the web app it will hit the web API Gateway which redirect to the Basket API which is send the basket to RabbitMQ which is consumed by the Order microservice**

**The communication between Basket API and ordering API through RabbitMQ through event driven architecture**

\*

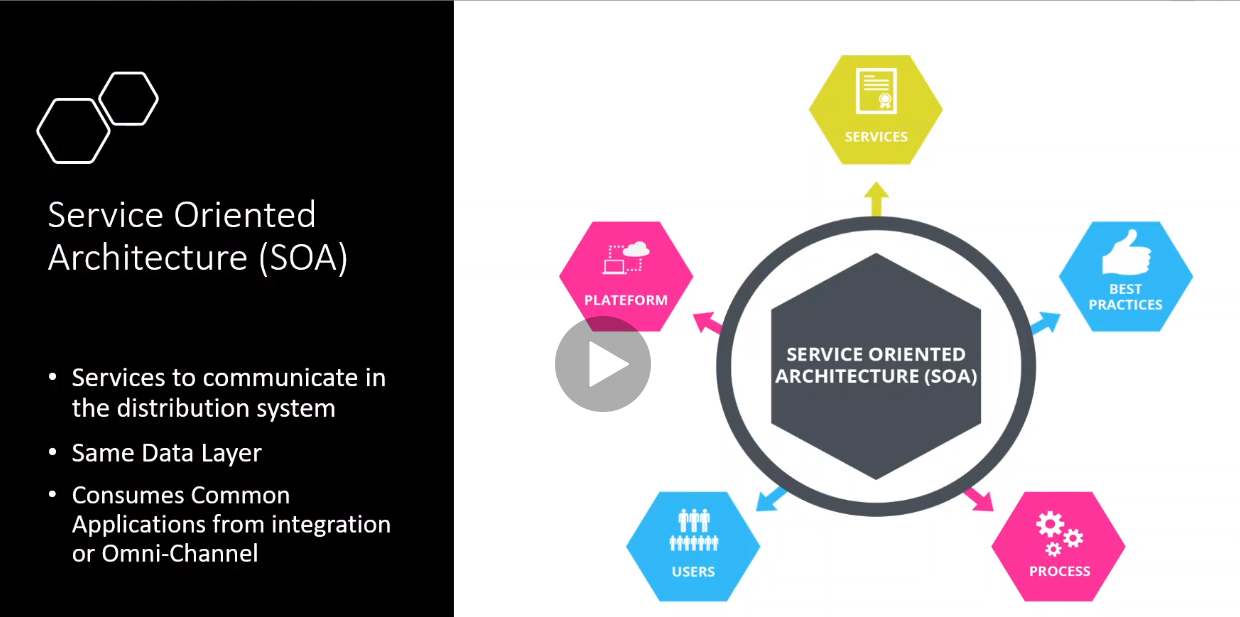
**\*Products coming from the Catalog microservices which using MongoDB and we call the catalog microservice through Ocelot API gateway.**

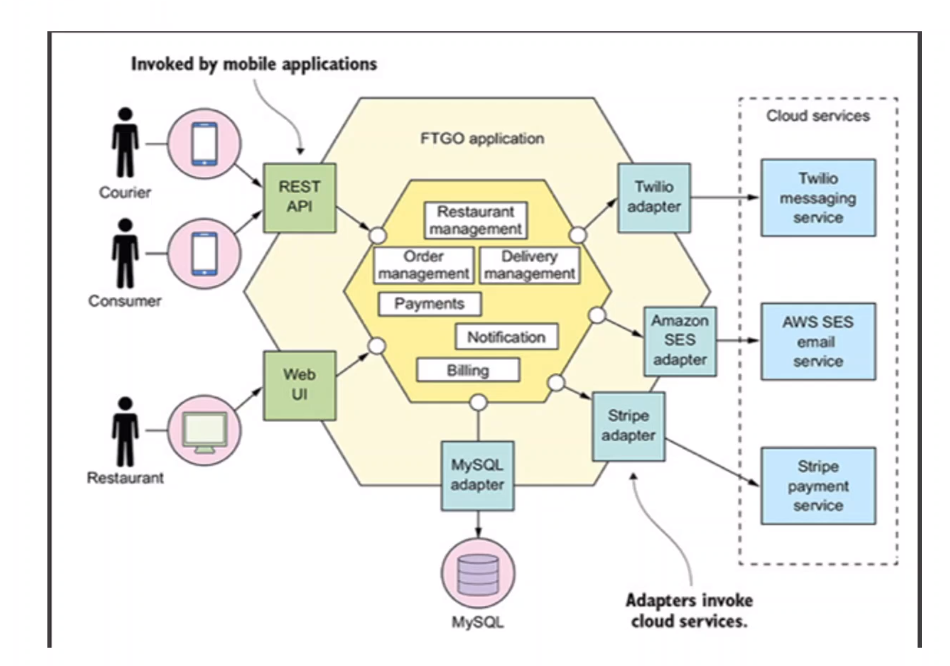
**\*Carts coming from the Basket microservices which using Radius DB and we call the Basket microservice through Ocelot API gateway.**

**\*Orders coming from the Order API which using SQL server which consume the RabbitMQ queue the send the queue message when checkout and submit the orders you listed.**

**Lesson02 What is the Service Oriented Architecture (SOA)?**

**Notes: -**





**1-Service-Oriented Architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.**

**2-The services communication can happen in two ways:**

**A-through passing data or through two**

**B- more services coordinating an activity**

**3-there are three Roles on the SOA**

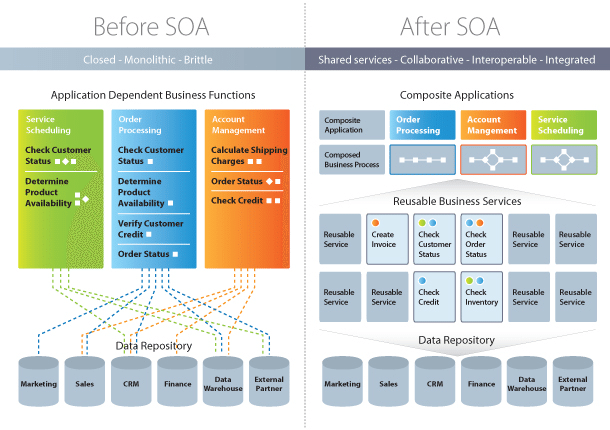
**A-servie provider works in conjunction with the service registry**

**B-service borker** **makes information regarding the service available to those requesting.**

**C-service requester locates entries in the broker registry and then binds them to the service provider.**

**D- It’s important to note that architectures can “operate independently of specific technologies,” which means they can be implemented in a variety of ways, including messaging, such as ActiveMQ;**

**Why SOA is important**



**1-SOA provide the reusable code , Service-Oriented Architecture also allows for using multiple coding languages because everything runs through a central interface.**

**2-with using SOA its allow business to limit the amount of analysis required when developing custom solutions**

**3-in SOA its important to be able to scale buisness to meet the needs of the clients**

**4-SOA can work with / without cloud computing**

**5-SOA recduce cost by reduce code change**

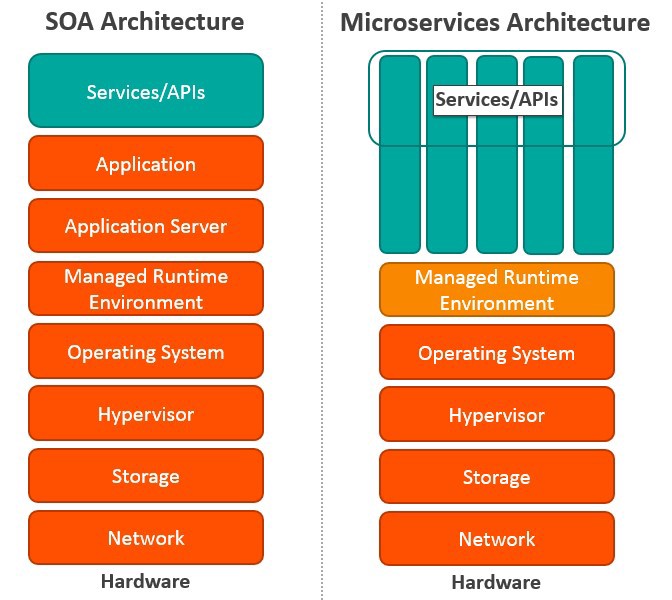
**Difference between SOA and Sass**

**Notes:-**

**1-SOA provide enhancment for the system you built but the complexity is high ,**

**2-SASS provide enhancment for the system but its provide avaible but hidden from users which provide flexibility for the sytem you built by SASS (software as service)**

**Difference between SOA and Microservices**



**Notes:-**

**1- Microservices, also known as Microservice Architecture, is an architectural style that structures an application as a collection of small autonomous services, modeled around a business domain.**

**2- There are four basic types of services:**

**A-Functional service: these define core business operations**

**B-Enterprise service: these implement the functionality defined by the functional services**

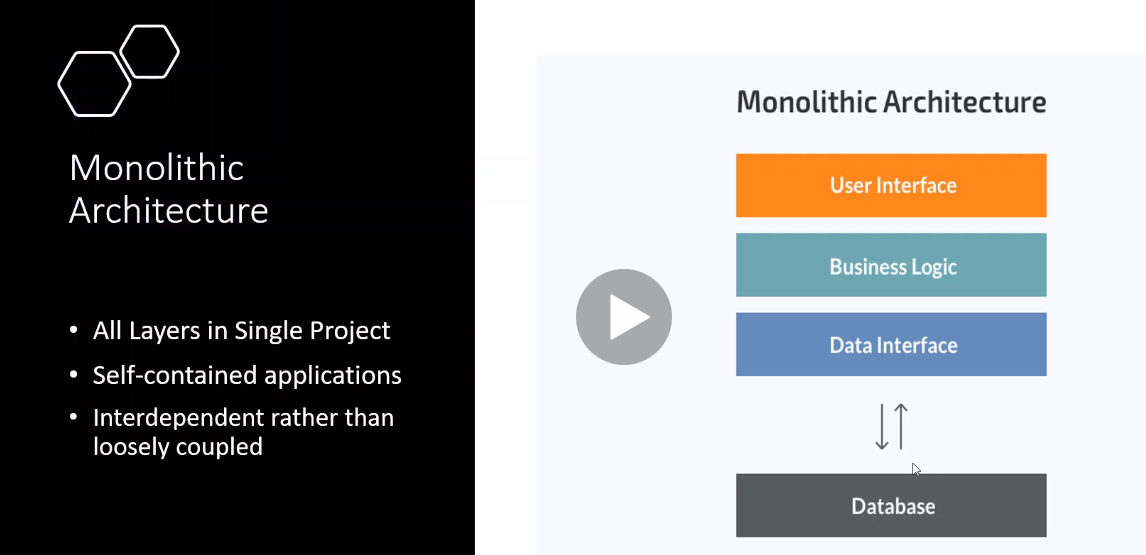
**C-Application service: these are confined to specific application content**

**D-Infrastructure service: implements non-functional tasks such as authentication, auditing, security, and logging**

**3-As you can see, each of these services builds on the one before it, creating a system that is not only easy to use, but provides you with a variety of ways to manage your business. As with any functionality, it’s a matter of figuring out what works best for you and your business.**

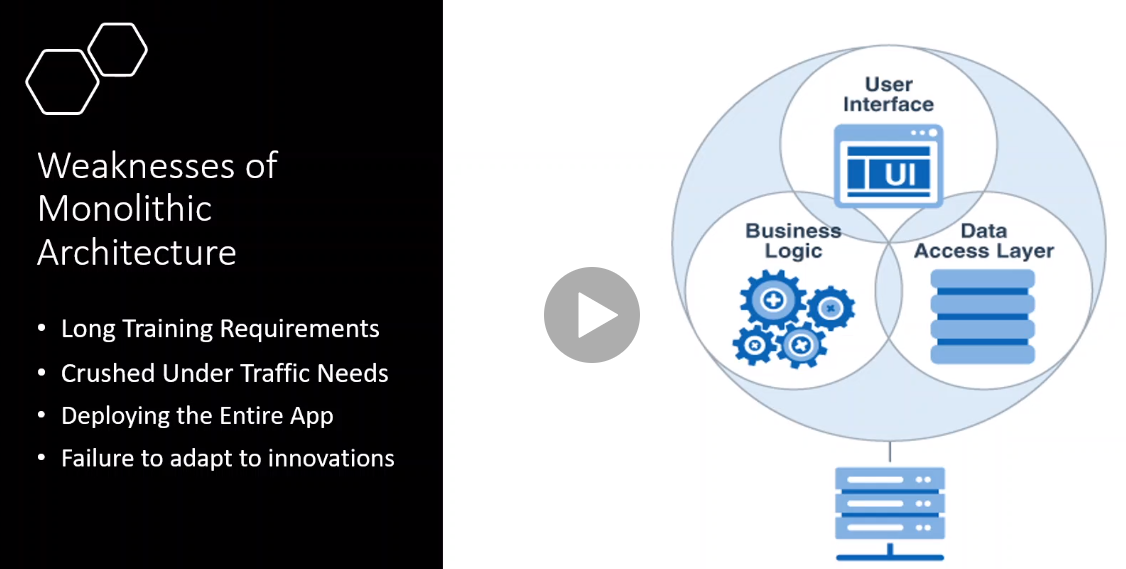
**Lessson03 Monolithic Architecture and whats the weakness and strengths**

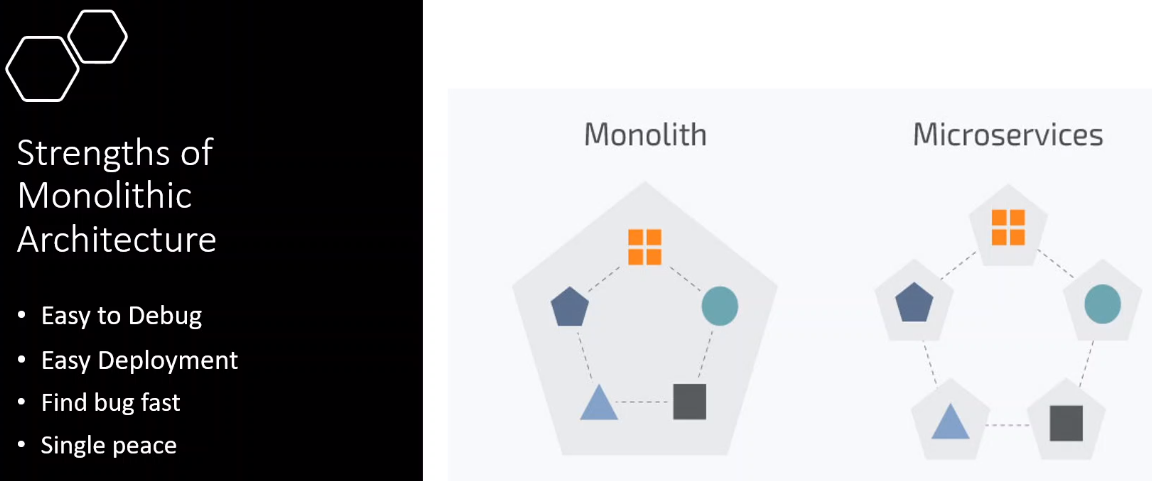
**Notes: -**

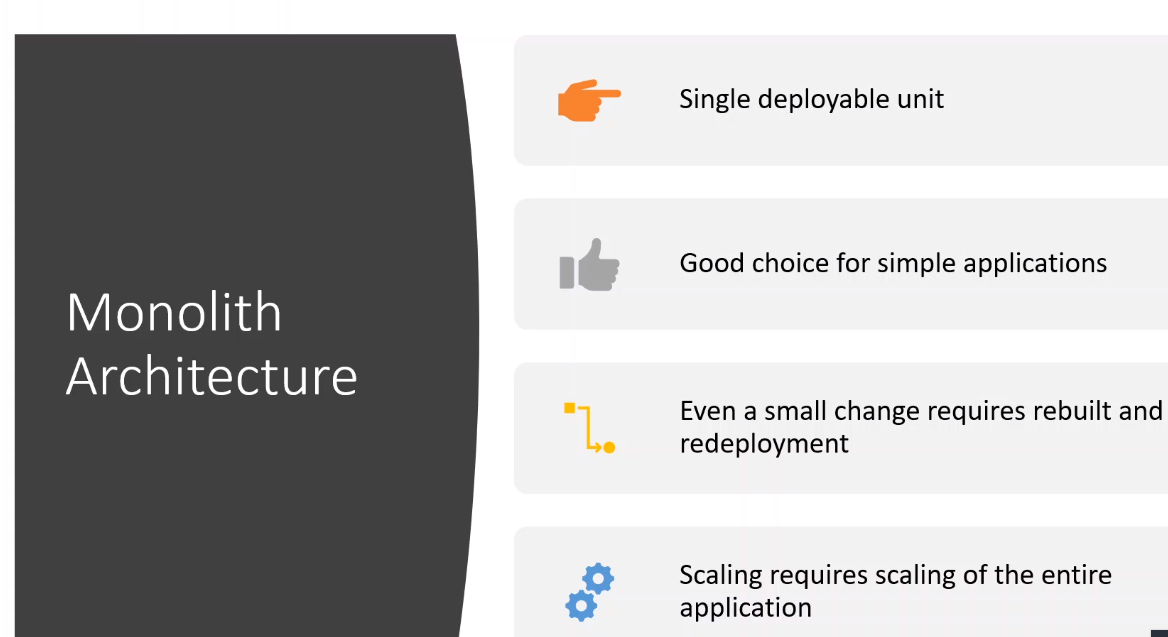


**1-this structure is not loosely coupled for the other systems**

**2-this structure is single piece self-contained each one depends of each other**

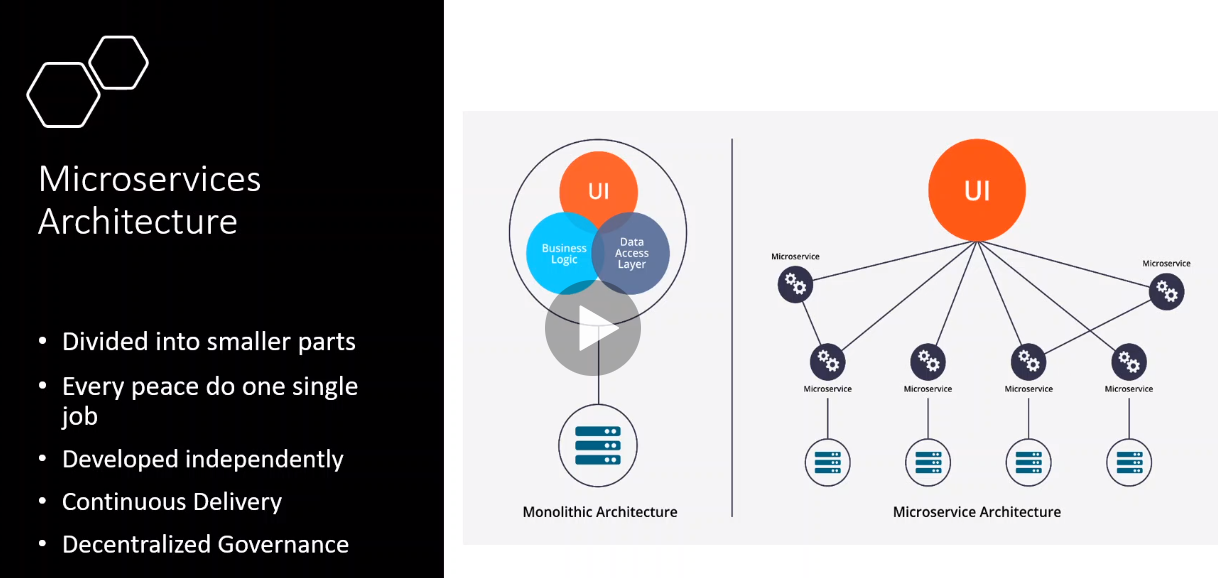






**Lesson04 Microservices architecture and**

**Notes: -**

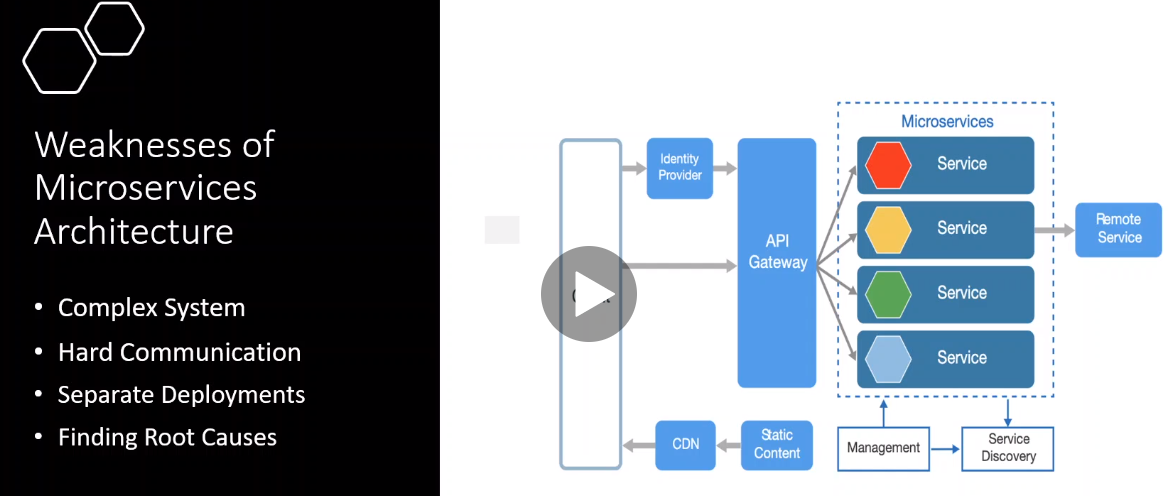


**1-Microservices are stateless which means that it does not keep in state of an object just send the object into queue and the consumer consume the queue taken**

**2-Microservices is low dependence on each other services, if we update specific services, we don’t need to upload the services that depend on that service because it using communication such as RabbitMQ as service bus with queue between sender and receiver.**

**3-services are loosely coupled which means independent**

**4-you are focus on targeted microservice only not the whole modules**



**Weakness of Microservice architecture**

**1-complex system in configuration and built**

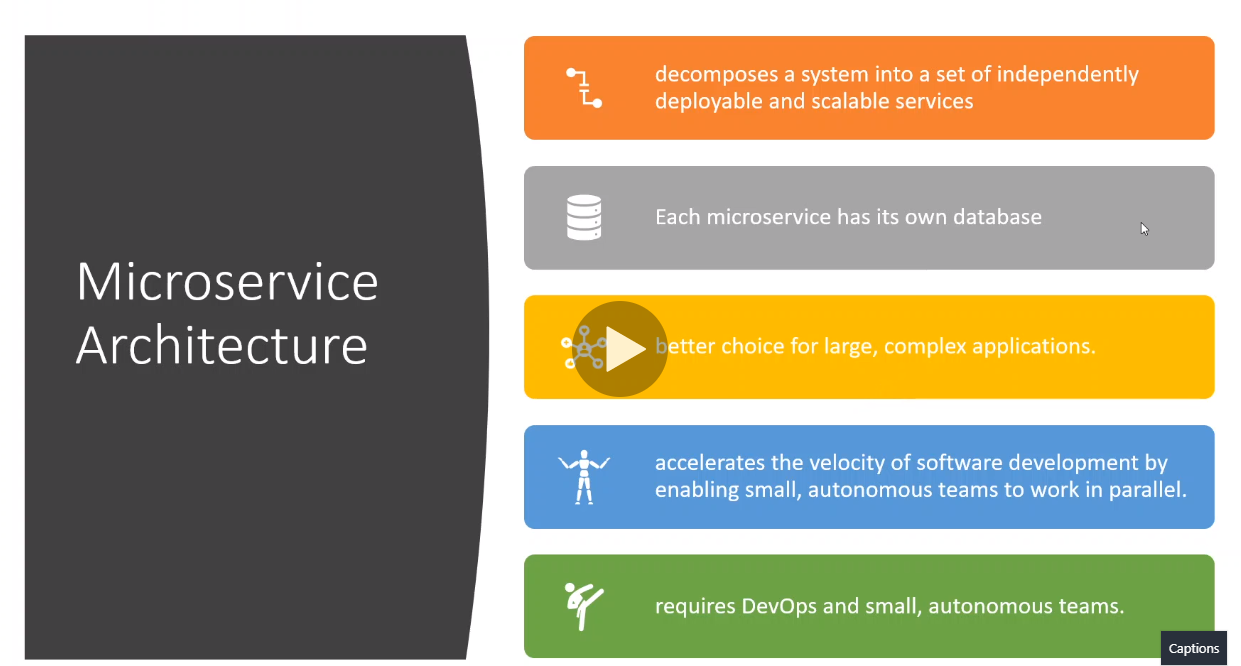
**2-separate deployment in publish each microservice and prepare the production and environment variables and so on.**

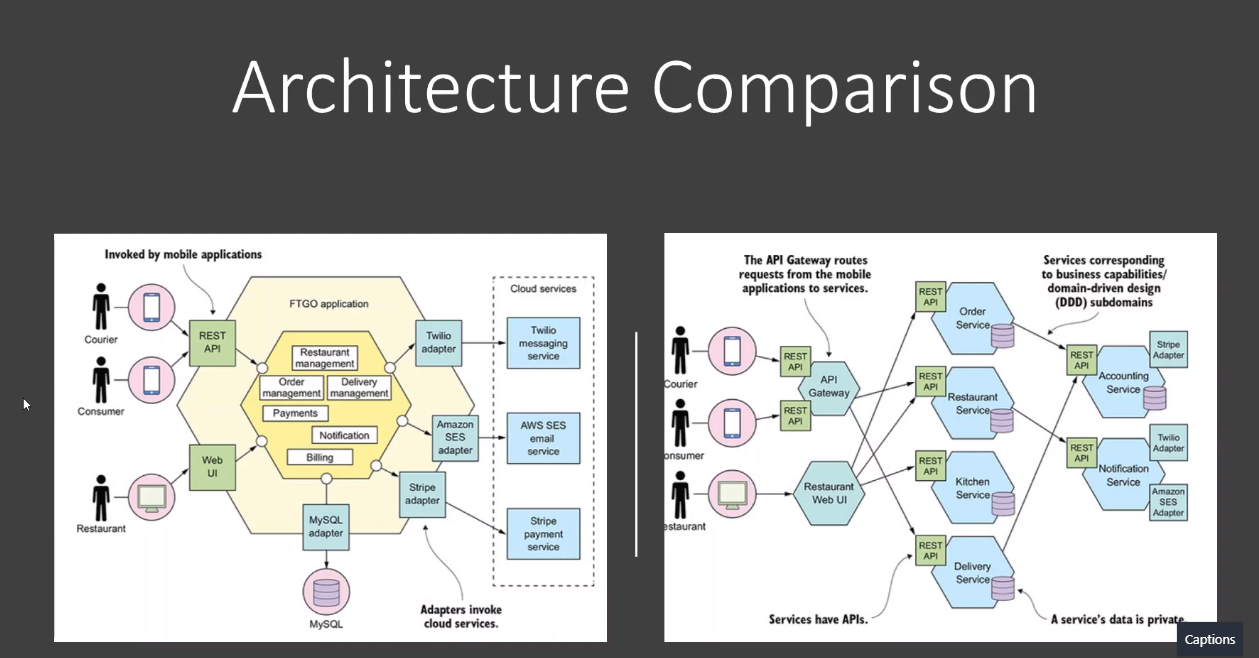
**3-debugging code is difficult because each microservice is loosely coupled which means you have to debug each microservice independently**



**1-each microservice can scale independently**

**2-each microservice can select different database such as SQL server , mongo , Reduis**



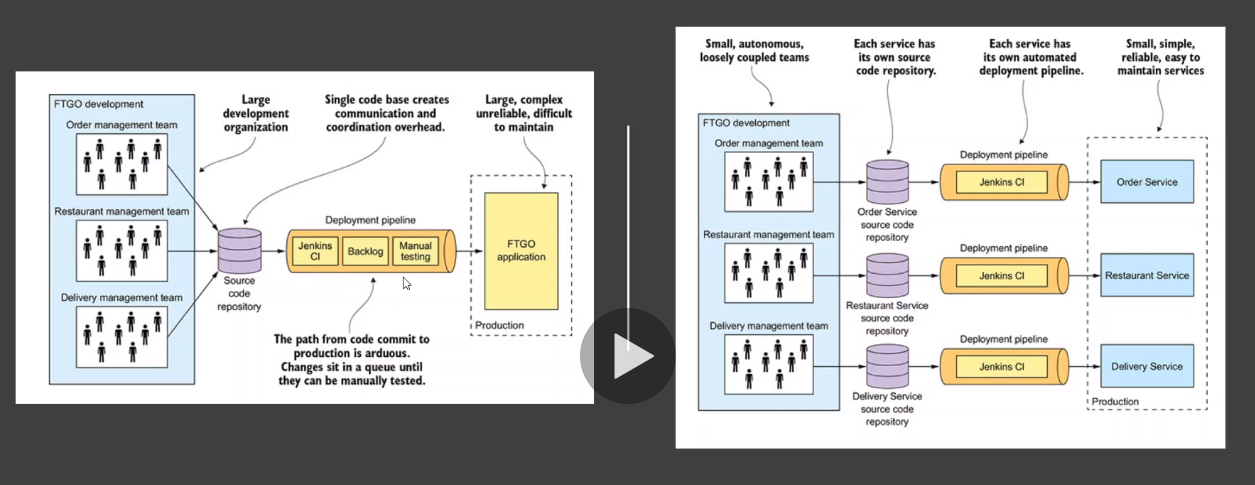


**1-We see that in SOA (service-oriented architecture) we see that all microservices are combined into single modules with single database and working independent with little communication such as RabbitMQ**

**(if you make migration of one microservice , you have to make publish to other microservices because all these microservices are reading from the same database on the same schema)**

**(we see also that the UI web app interact directly with the microservice**

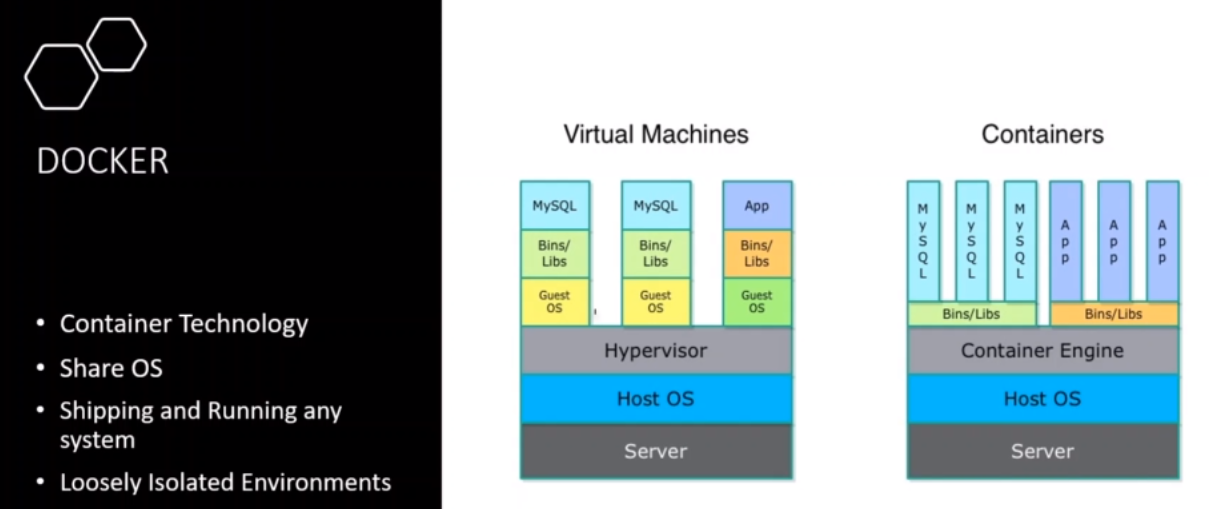
**2-While in Microservice we see that each microservice has its own database plus there are gateway API with ocelot that contains configuration of all the microservices**



**(each microservice have own database or schema which means you can have one database with multiple schema, so when migrate on microservice it will not effect to other microservice because its reading from different schema)**

**Lesson05 Docker started with Docker commands**

**Notes: -**



**1-docker is open source virtualization platform which can install containers with this platform and install test and deploy systems**

**2-one of the benefits of docker that it eliminates why the app working on my computer and not working on the server by making image and make it reusable**

**3-with virtual machine approach we using hypervisor to make multiple virtual machines running on the same machine and you have to make sure that all DLL’S and configuration are set on each VM**

**4-with docker we built images that each one contains all the configuration you need and working on any machine and you can convert it to Linux or windows as you want.**

**5-you can upload your images into docker hub**

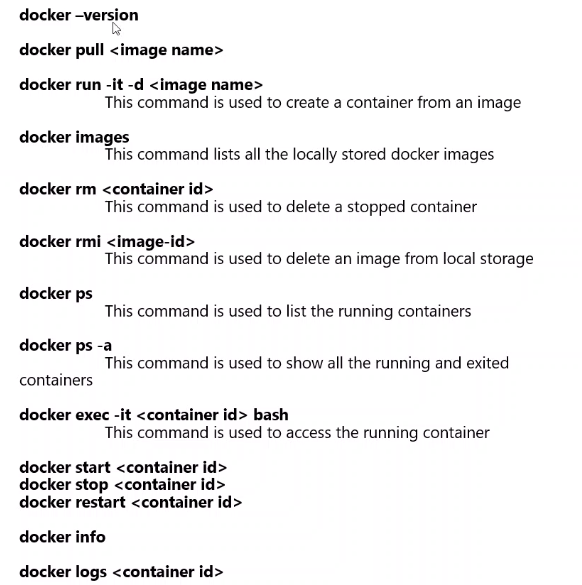
**Docker Architecture**



**1-we have Docker hub which contain the Docker registry which contain all the images online such as radius, mongo, etc.**

**2-when download its download on the Docker host which contain images section and container section**

**(images can be use in multiple containers as below)**



**The best command to create mulitple containers from the docker-compose.yml and docker-compose.yml is below**

**Docker-compose -f docker-compose.yml -f docker-compose.override.yml up-build**

