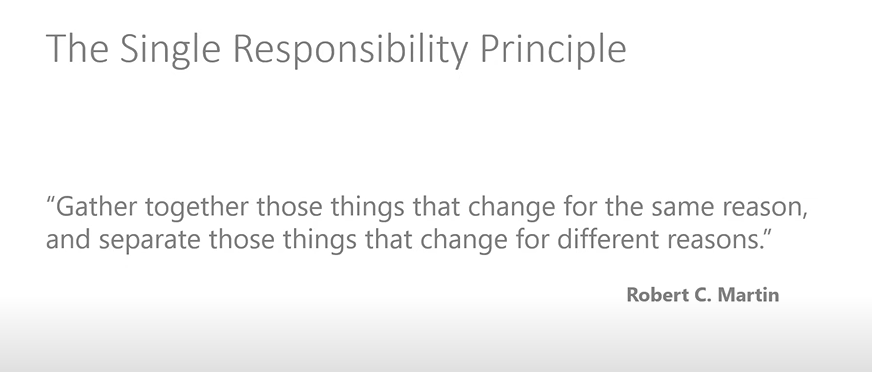
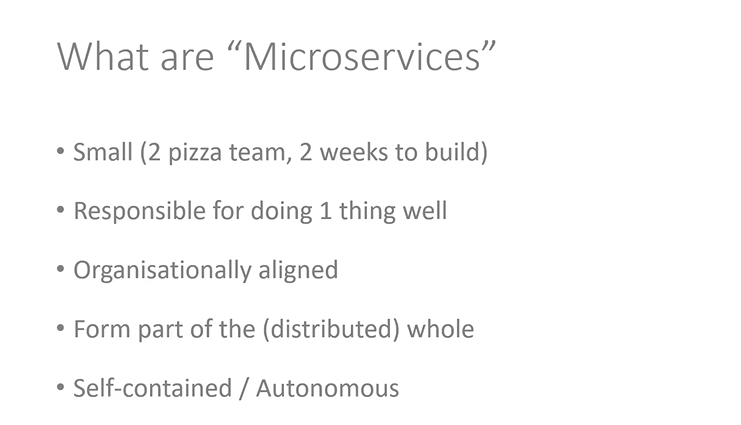
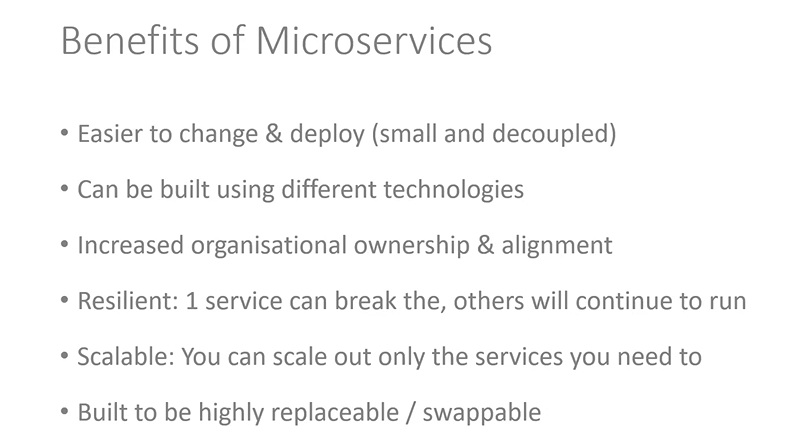
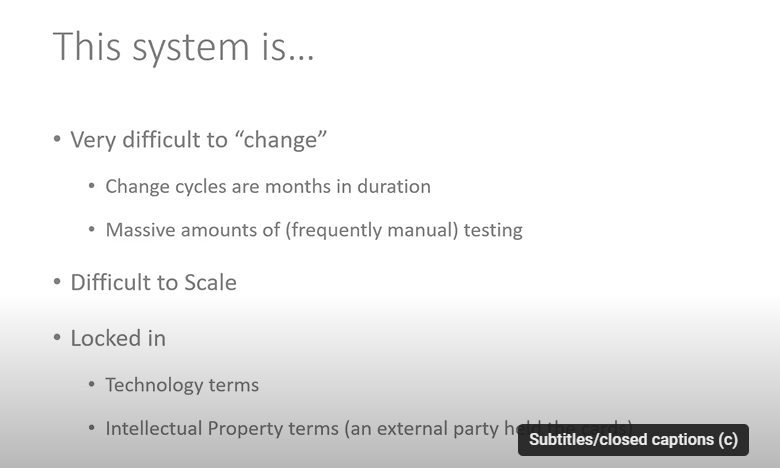
**Microservices**

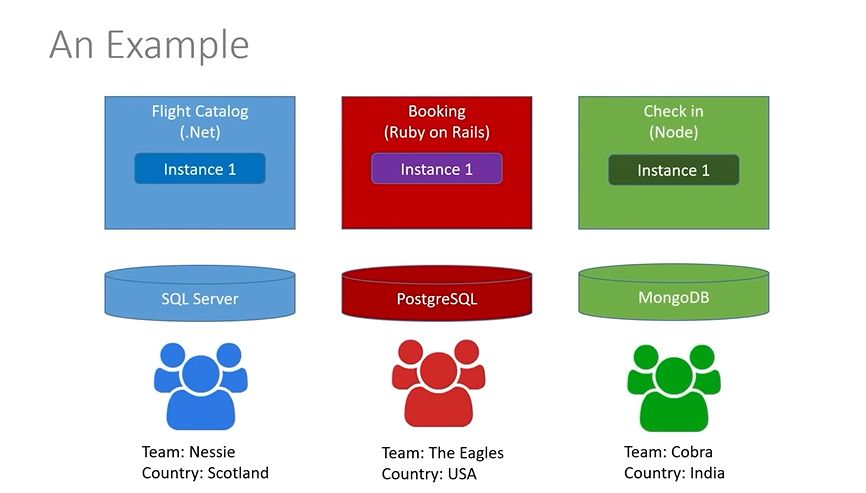
****

****

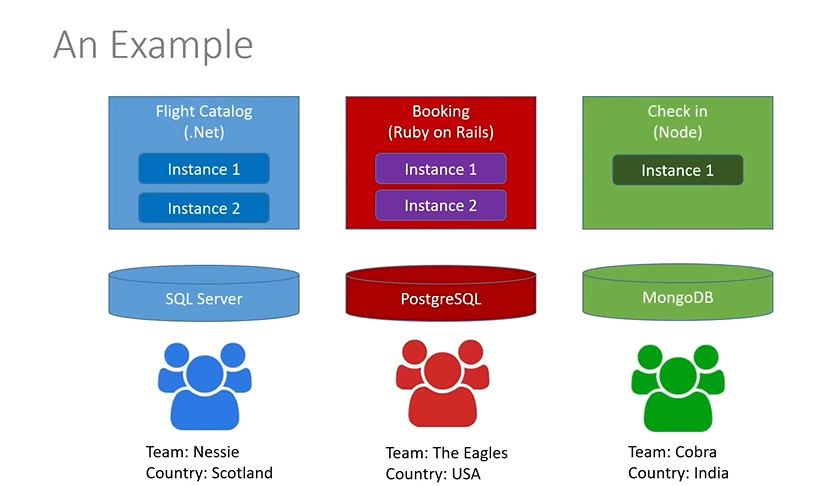
****

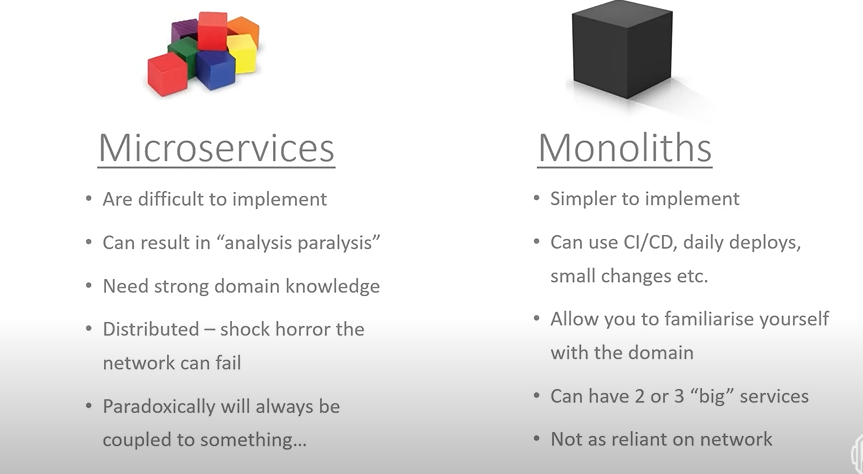
**Monoliths are**

****

****

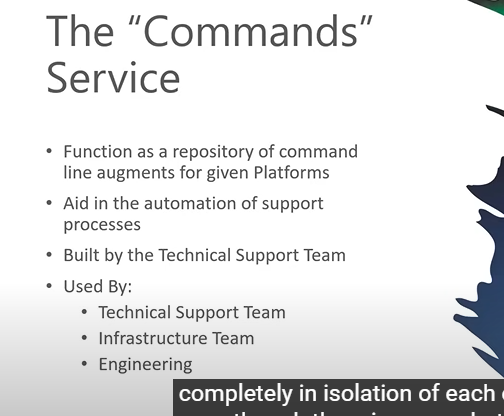
**More traffic on browsing flights and booking then scale individually**

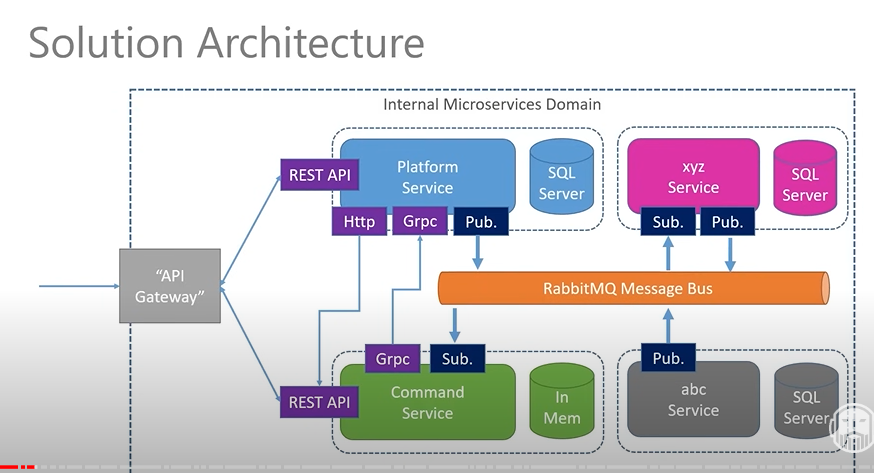
****

****

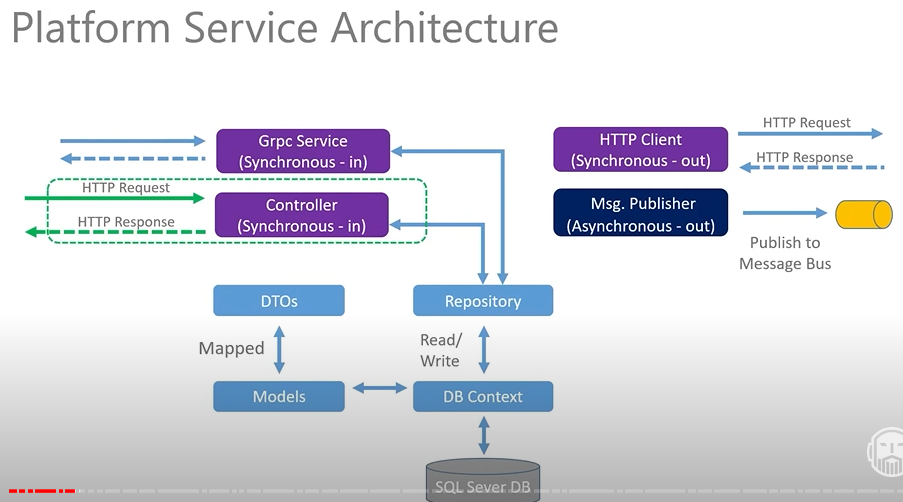
**Services of learning project**

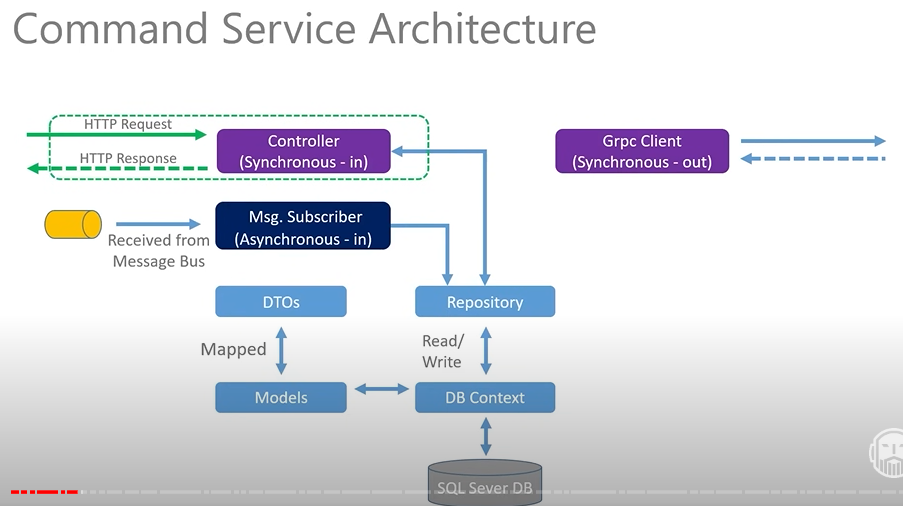
****

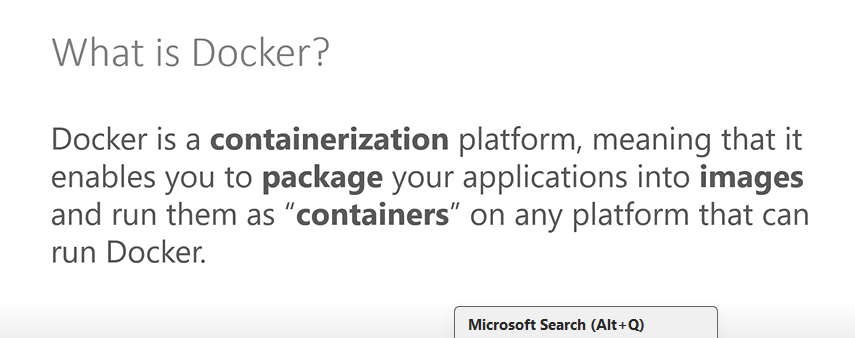
****

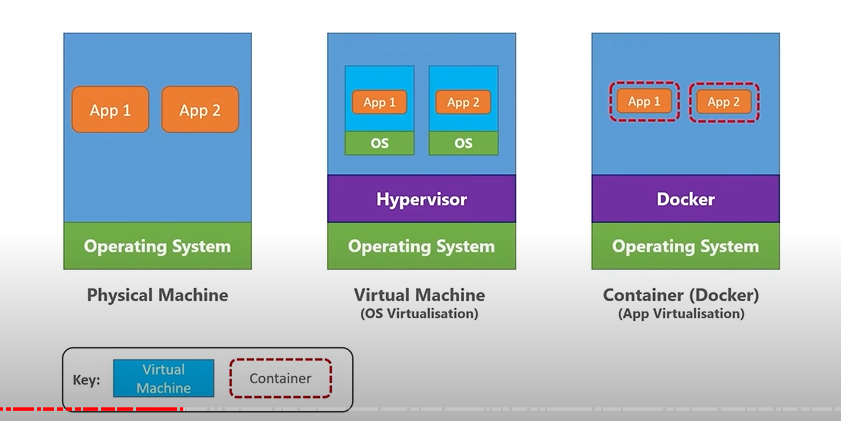
****

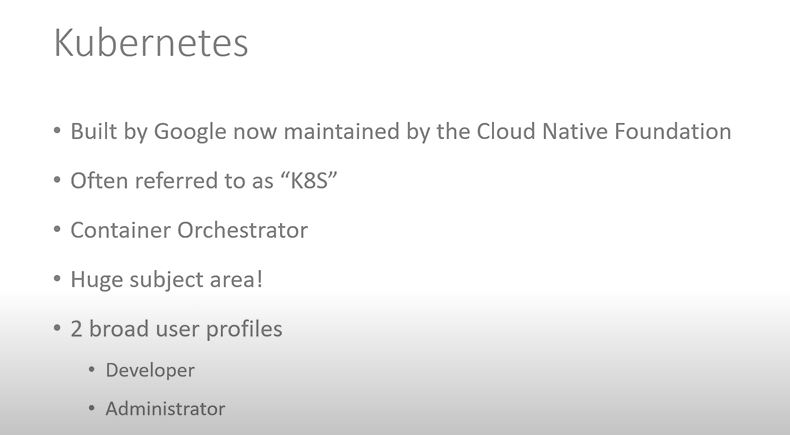
**Use of RabbitMQ Event bus to create a decoupled microservices architecture.**

****

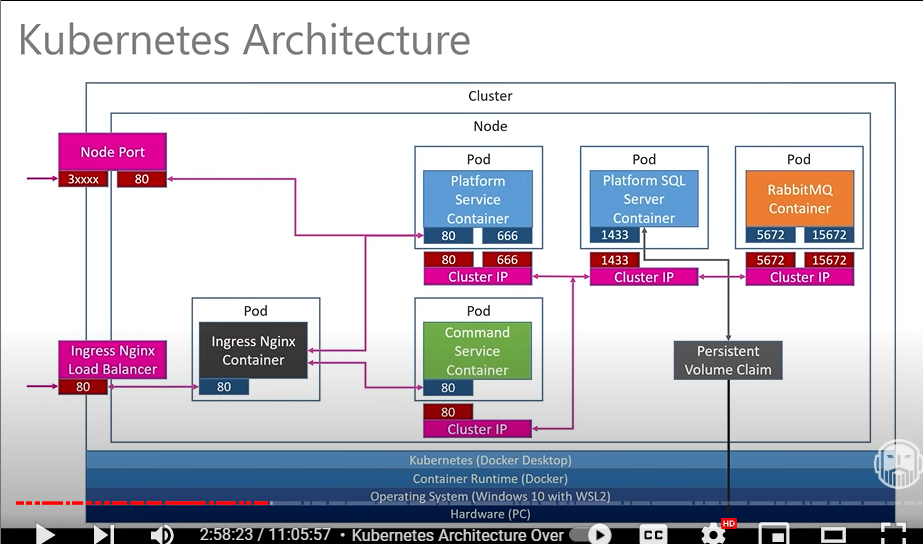
****

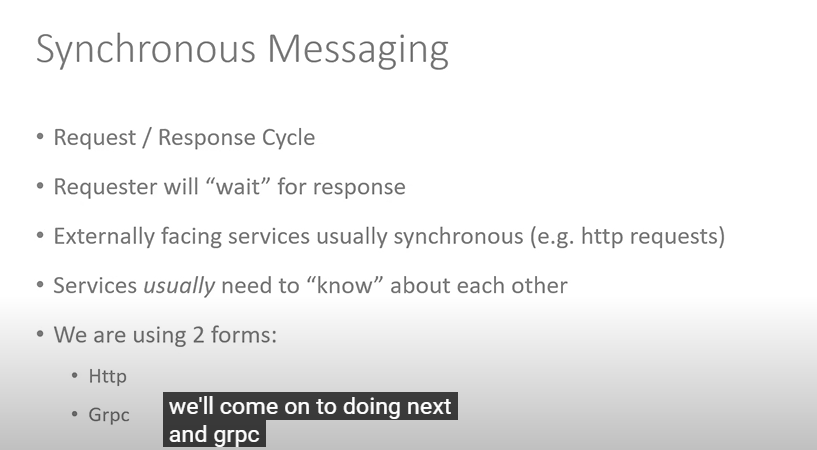
****

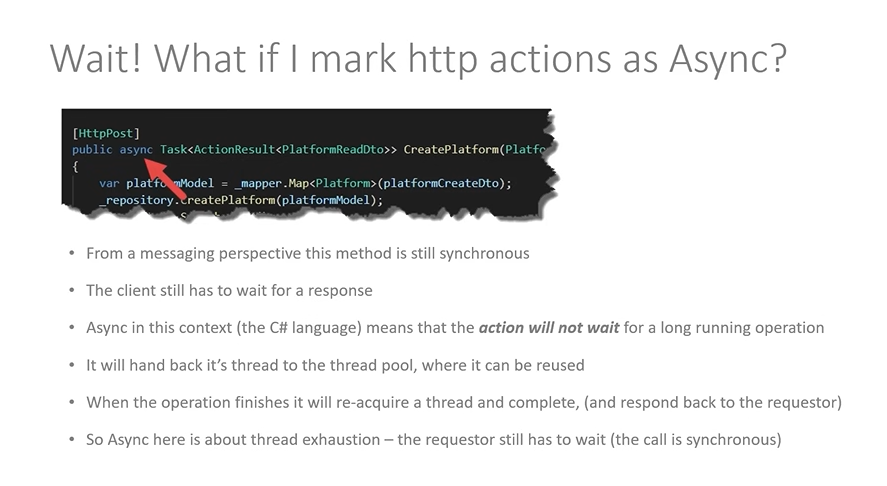
****

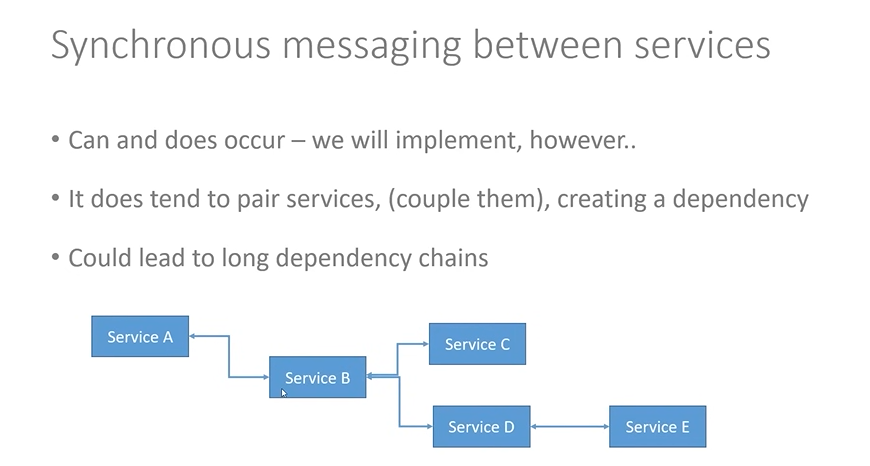
****

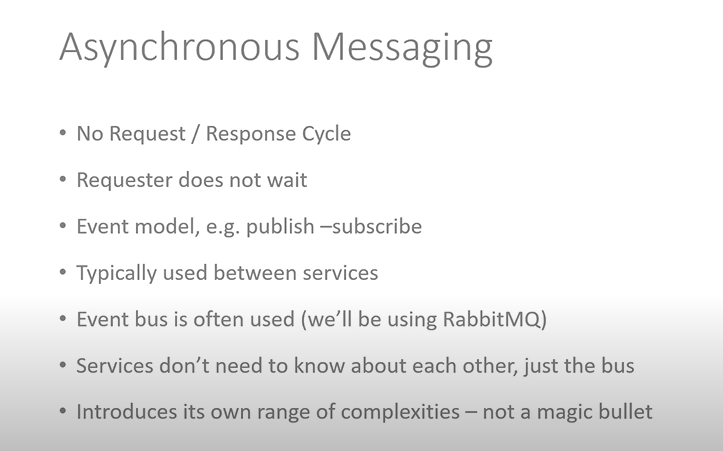
**As the Project was made in .net 5 and I am developing in .net 8 the default port for running api server is changed from 80 -> 8080 from .net5 -> .net8**

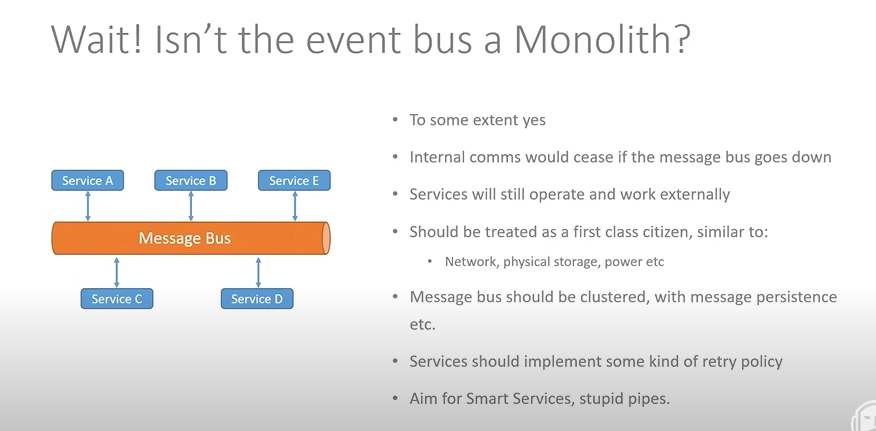
****

****

****

****

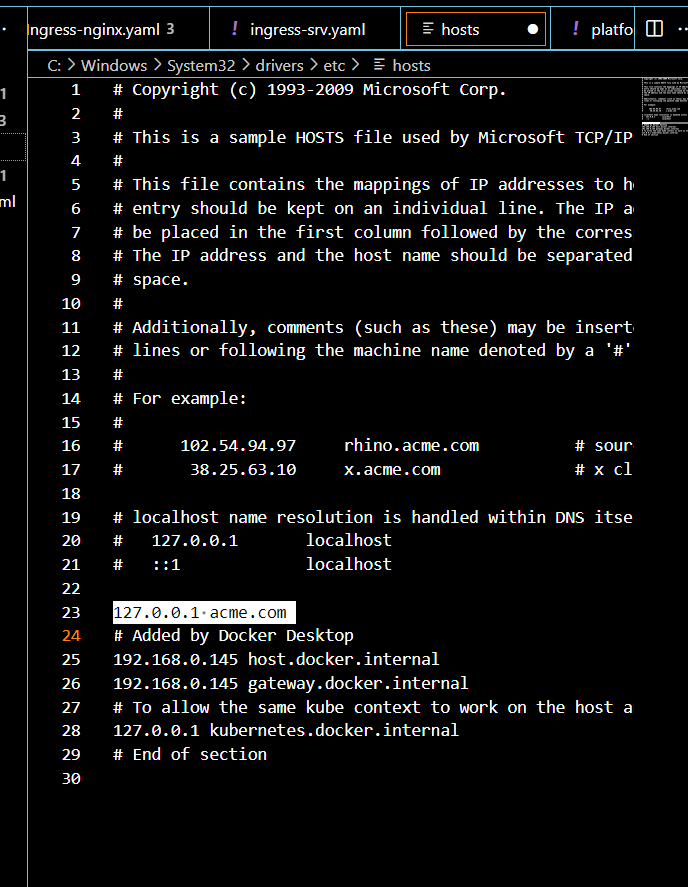
****

****

**We are setting up the API Gateway using an Ingress Nginx Container and Ingress Nginx Load Balancer whose yaml file is procured from following url**

<https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.1/deploy/static/provider/aws/deploy.yaml>

**After Deploying ingress-nginx we have to define a ingress service file where we configure the api gateway to app paths for the requests apart from that we also have to update our host file to loopback to localhost on hitting a specific domain in our case acme.com**

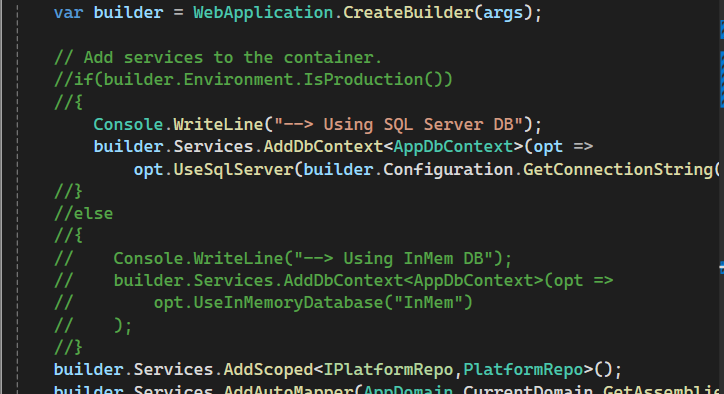
****

**To Setup SQL Server instance we have to create a secret which will be used as an sa username and password. We cannot directly hardcode it as it will be useless.**

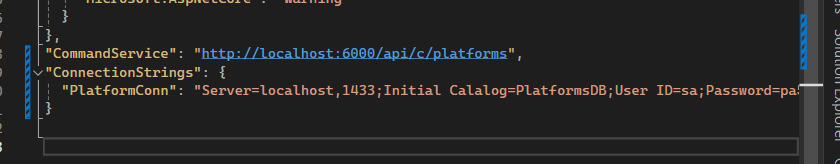
kubectl create secret generic mssql --from-litera  
l=SA\_PASSWORD="pa55w0rd!"  
secret/mssql created

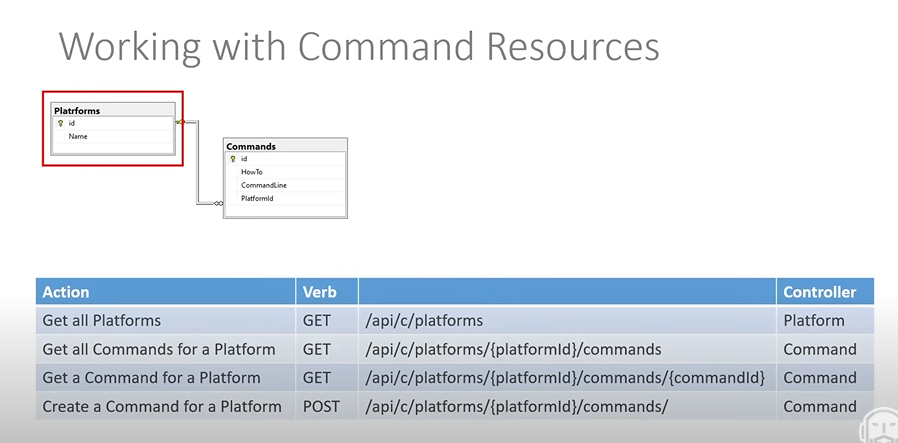
**After this we create a mssql-plat-depl.yaml basically a deployment file to set up sql server as well as mount a sqlserver to a persistent volume claim.**

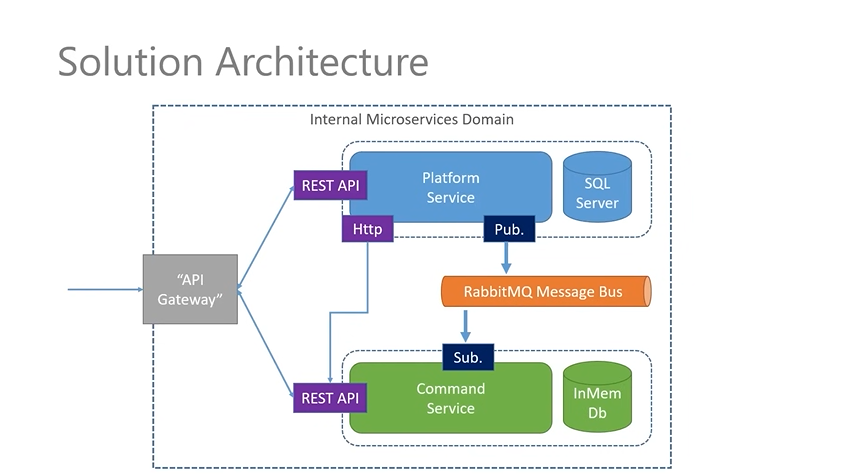
**To Carry out migrations correctly we have to trick EntityFramework to think we are in production do comment out code to check for development and leave only production code to add DbContext**

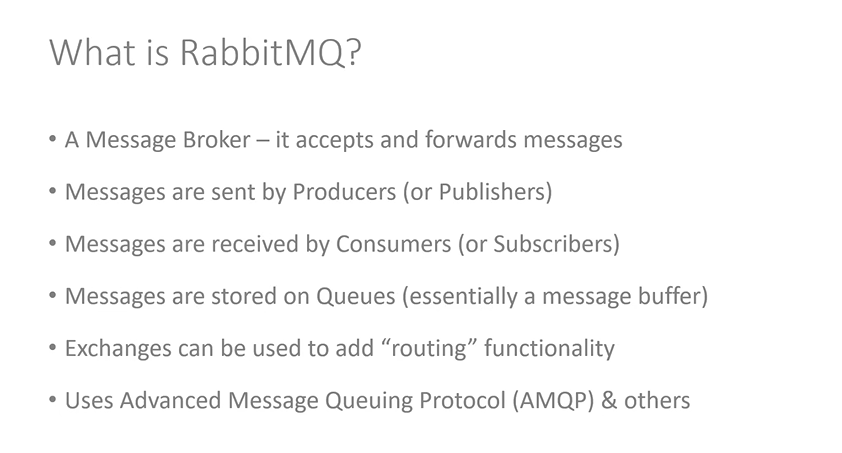
****

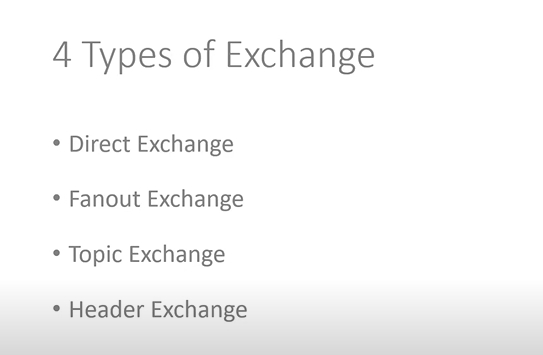
**Also make changes in appsettings.Development.json fileto connect to database**

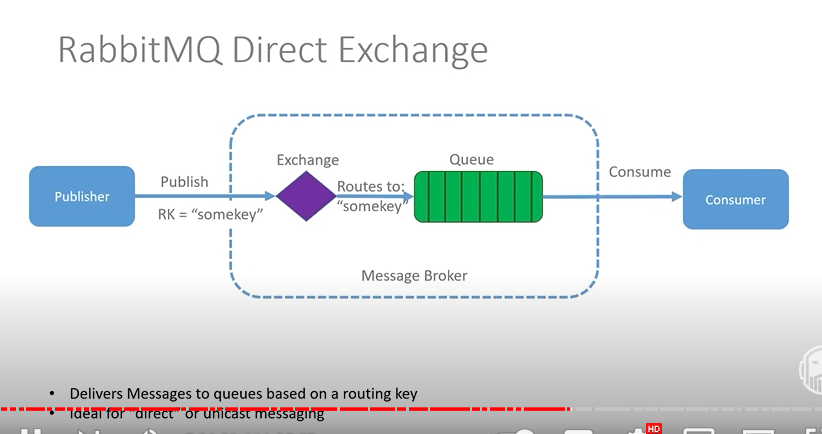
****

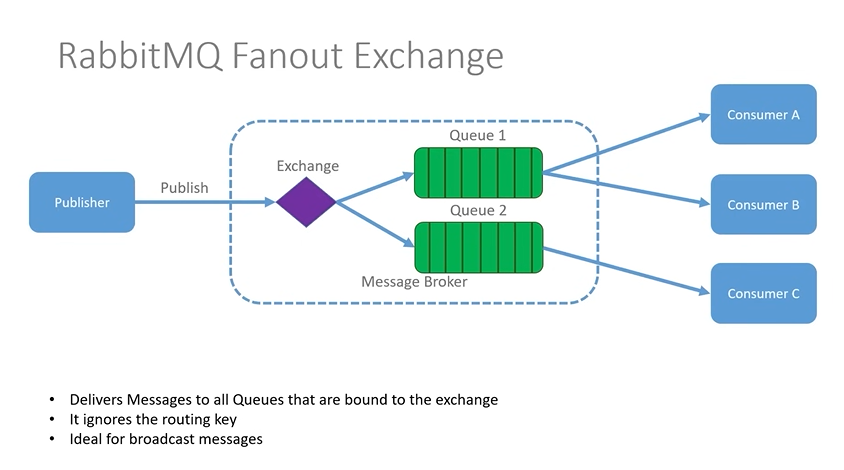
****

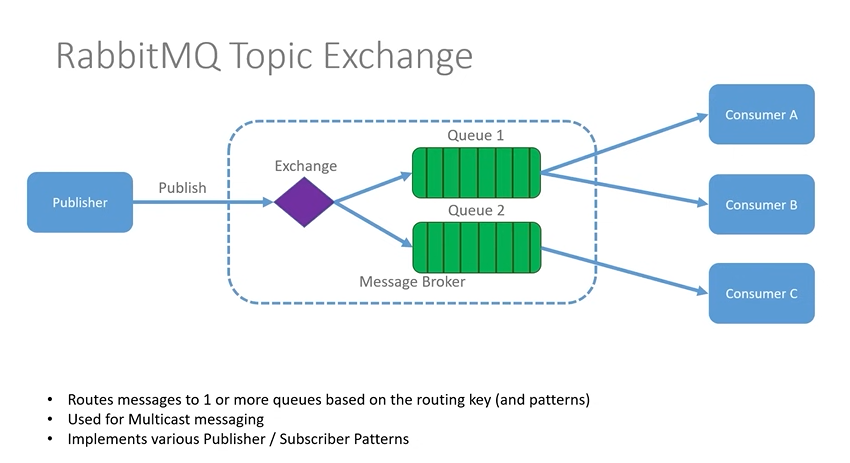
****

****

****

****

****

****

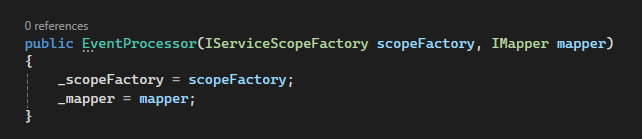
**After Rabbitmq deployment access rabbit mq management server at** [**http://localhost:15672/**](http://localhost:15672/)

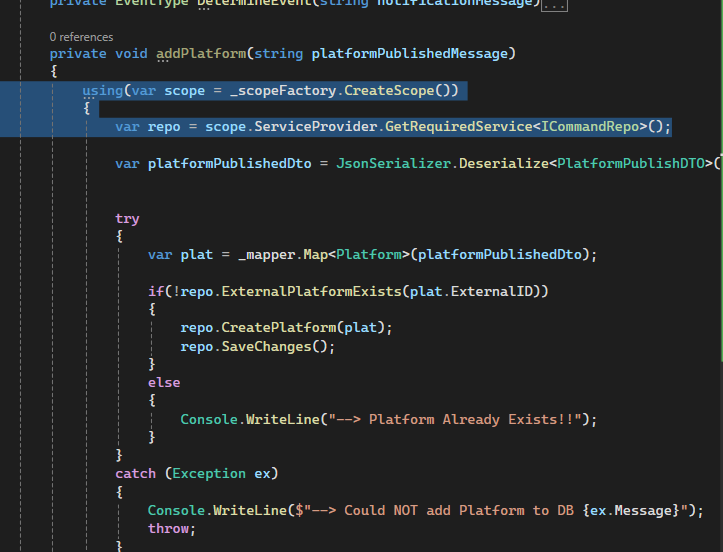
**With userid as guest same pass**

**CommandsService :**

**We will be creating a EventListning service (Singleton) and an EventProcessor service (Singleton). To process the event we will check that event and if it is platform published then we will add it to our dbcontext. But we cannot directly access Repository/dbcontext in our EventProcessor service as there lifetime is smaller that EventListner and EventProcessor.**

**To Get around that use scoped service factory**

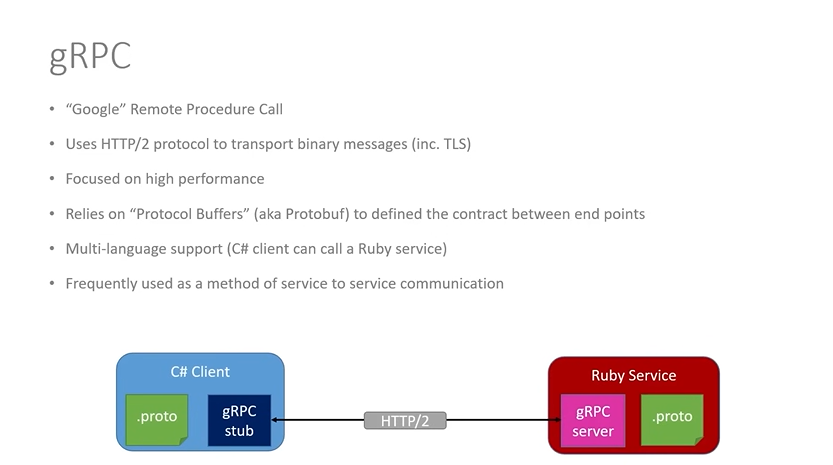
****

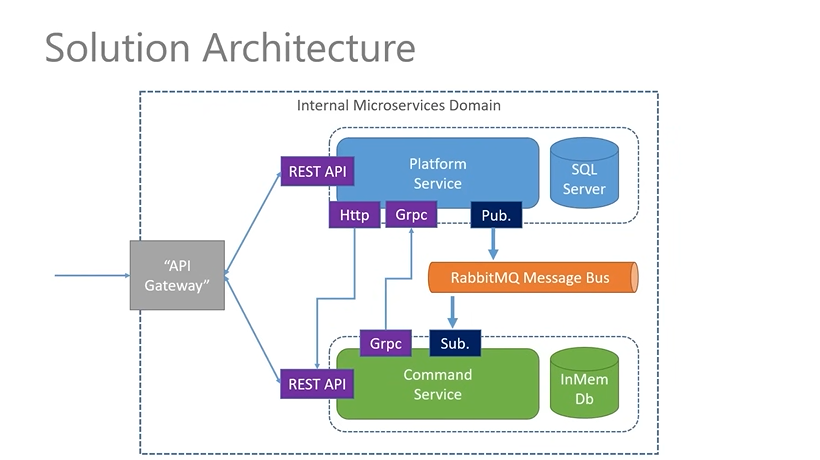
****

**After pushing new built images to docker hub we have to restart the Kubernetes containers**

kubectl rollout restart deployment commands-depl

**Now we have a working solution on Kubernetes deployed. But we do not have the initially seeded platforms(created from migrations) in the commands service as created events were never fired for those. We can solve this using GRPC.**

****

****

**GRPC uses TLS which has HTTPS protocols by default but for our case with service communication internally we are using HTTP so we need to configure our deployment and explicitly tell clusterip service where GRPC endpoint is to use HTTP.**