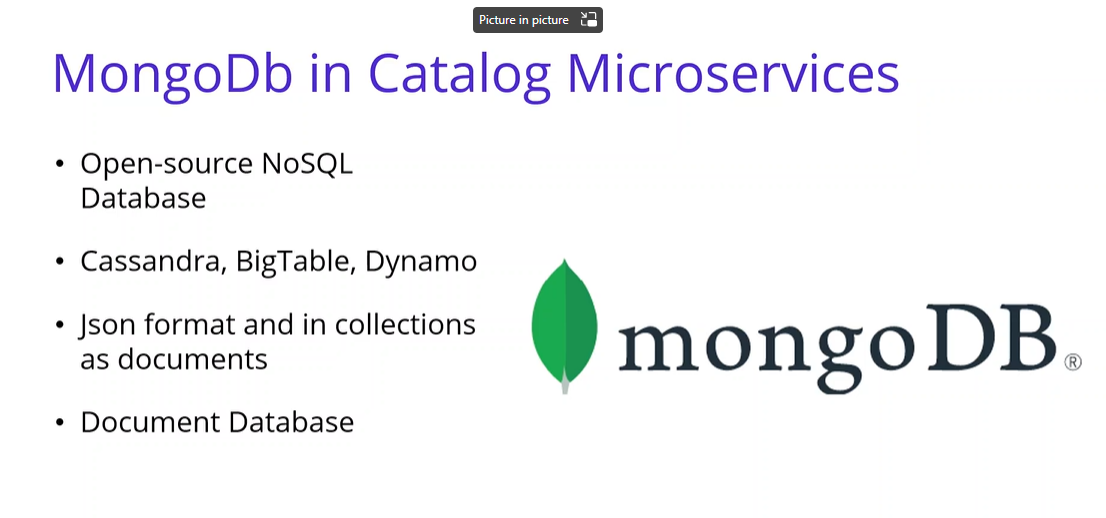
**Develop Micro services With Mongo DB**

**Notes:-**



**We see that mongoDb is an open data source that is designed for easy development and it’s the data on JSON format and its contains collection and inside documents**

**Collection 🡪 tables**

**Document 🡪 rows**

**Steps**

**1-we go to docker hub and install docker**

**2-we install docker and search for mongo and then install mongo**

**docker pull mongo //it will create image for mongo**

**docker ps //to get list of containers**

**docker images //to get list of images**

**//to create new instance of image mongo on port 27017 based on mongoDb image**

**docker run -d -p 27017:27017 --name shopping-mongo mongo**

**//to get basic info of the container shopping-mongo**

**docker logs -f shopping-mongo**

**//to get access to internal mongoDb bash portal to execute commands**

**docker exec -it shopping-mongo /bin/bash**

**(We can create collection, edit collection, list collection, etc…)**

**show dbs //it will get list of databases**

**use CatalogDb //it will switch and create new database CatalogDb if not exist**

**db.createCollection('Products') //it will create collection called Products**

**//it will create documents inside this collection called Products**

**db.Products.insertMany([{ 'Name':'Asus Laptop','Category':'Computers', 'Summary':'Summary', 'Description':'Description', 'ImageFile':'ImageFile', 'Price':54.93 }, { 'Name':'HP Laptop','Category':'Computers', 'Summary':'Summary', 'Description':'Description', 'ImageFile':'ImageFile', 'Price':88.93 } ])**

**//it will get list of products**

**db.Products.find({}).pretty()**

**//it will get products based on id**

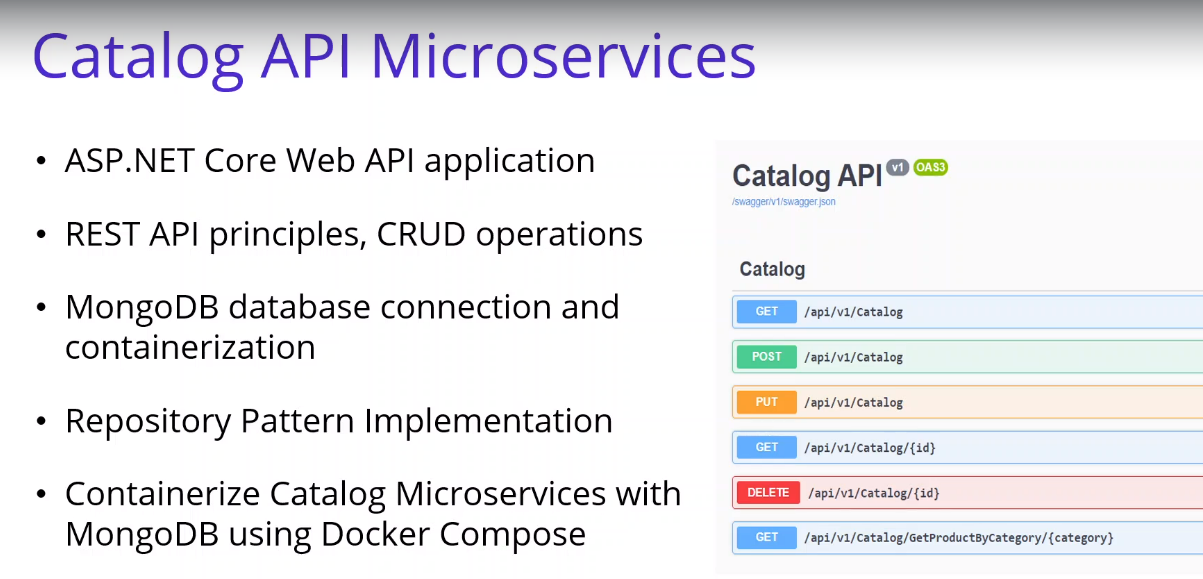
**db.Products.find({\_id : ObjectId("62c0f4e3b5de41af2ec82d2f")}).pretty()**

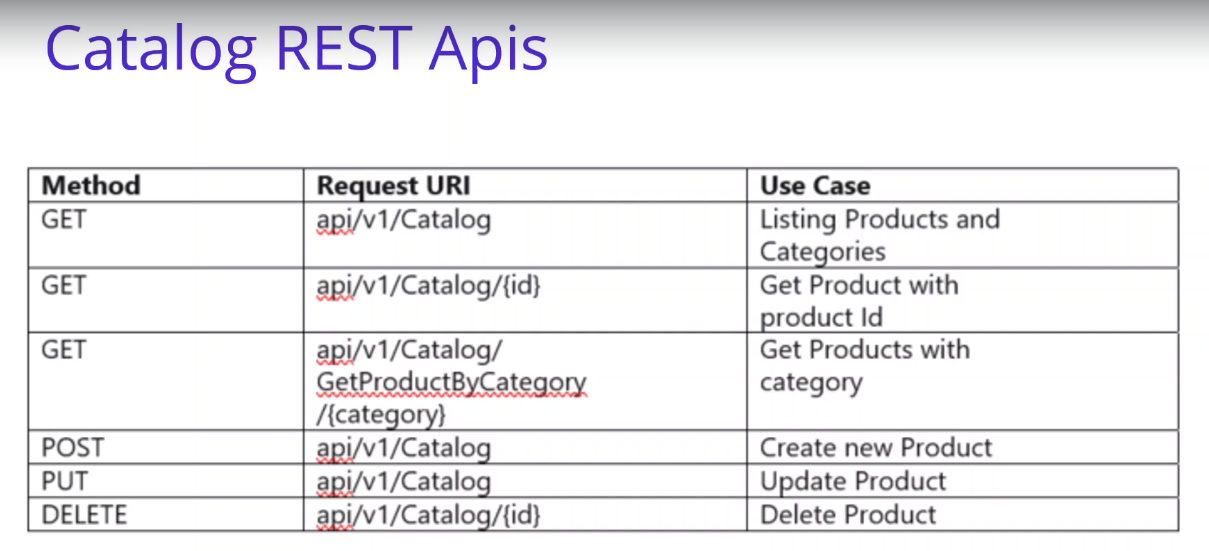
**//it will remove all documents inside collection products**

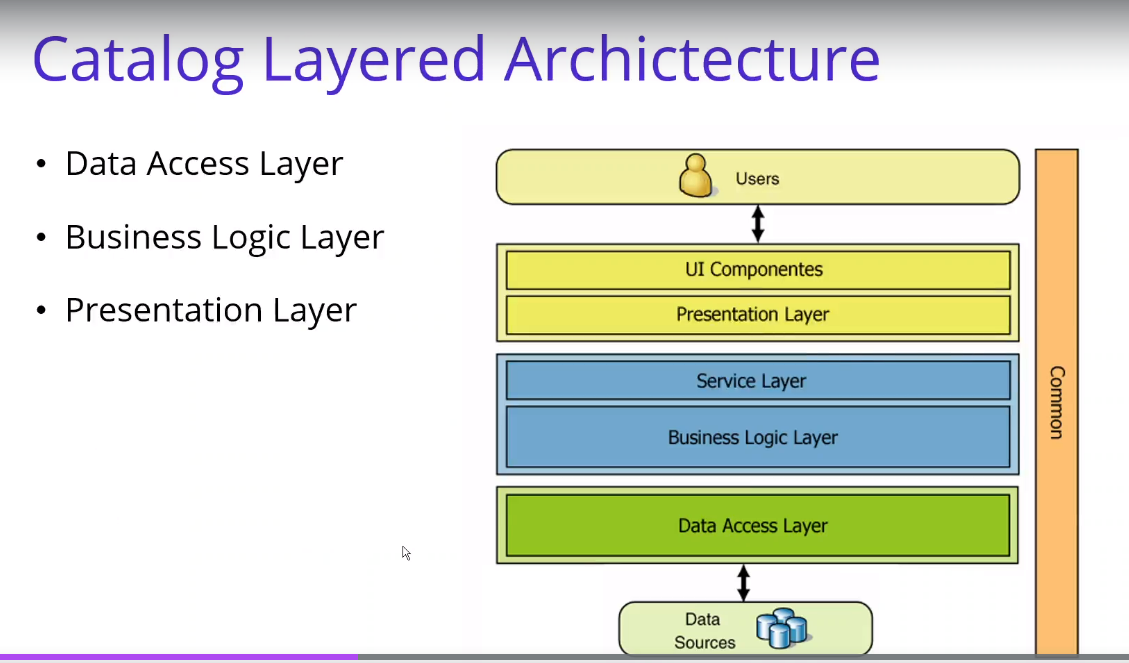
**db.Products.remove({})**

**//it will show product collections**

**show collections**

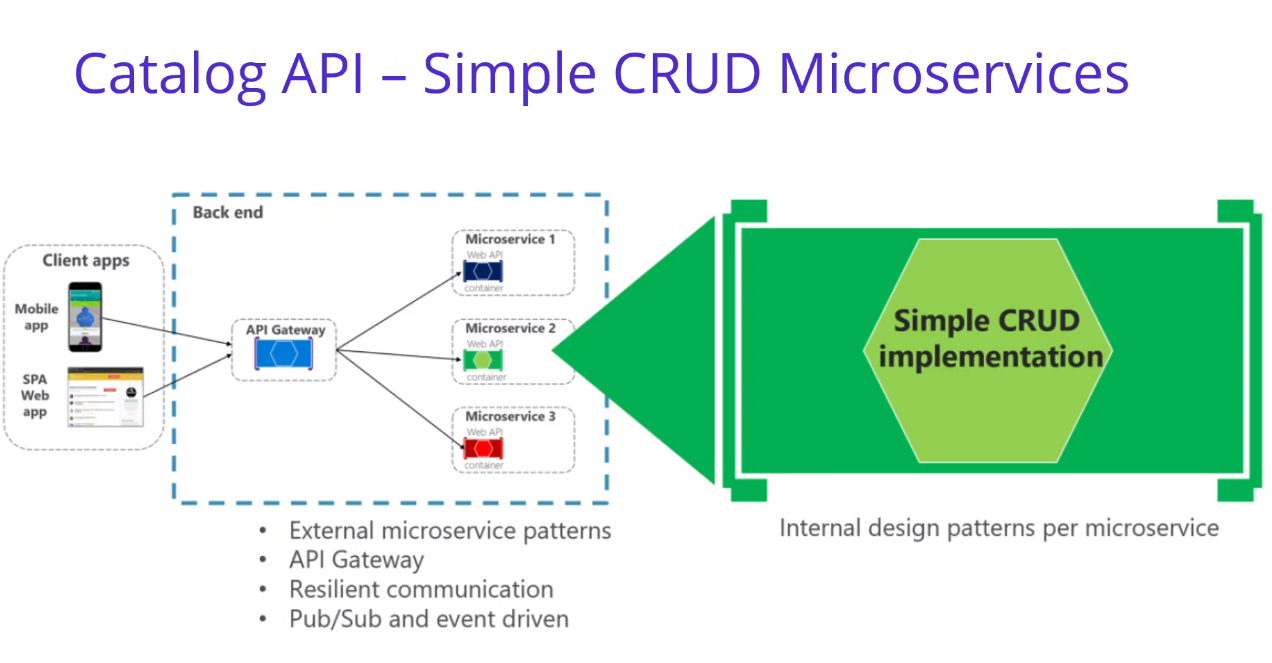




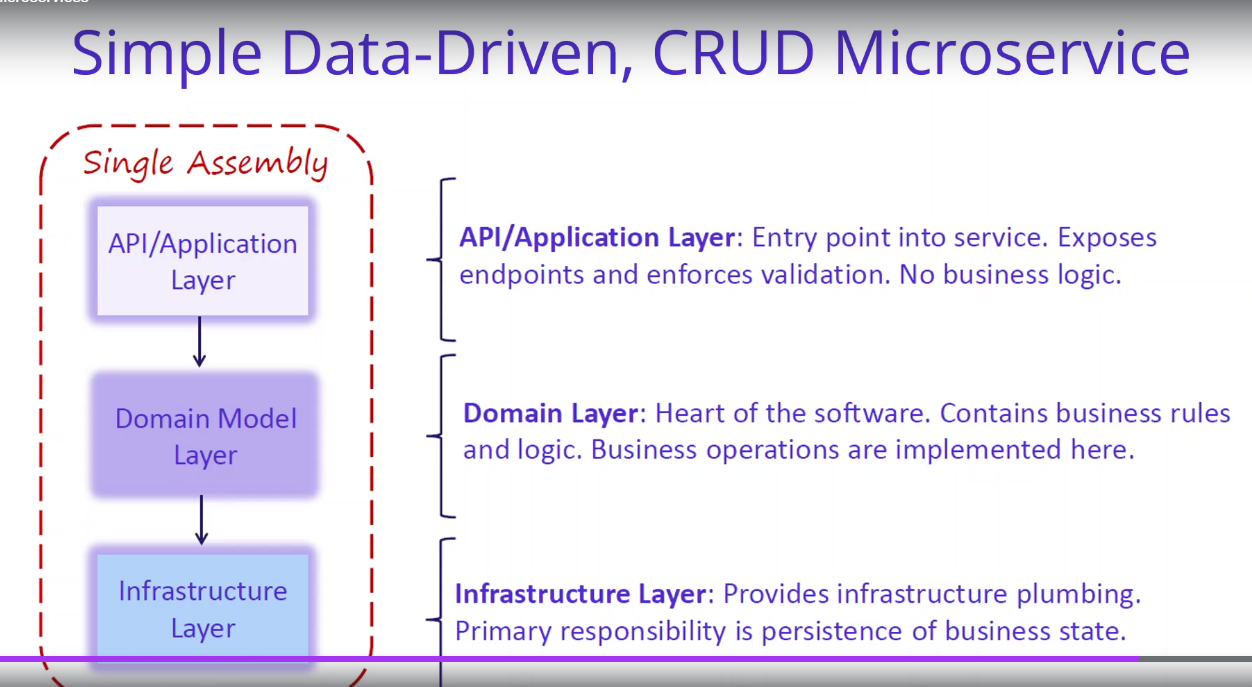


**We use N-layer architecture (motholonic architecture) (old structure traditional)**

**Micro services**



**We use clean micro service which contains API gateway which is responsible to handle request and detect the target micro service it target and publish message and on the target micro service it will subscribe the message and consume and execute it**

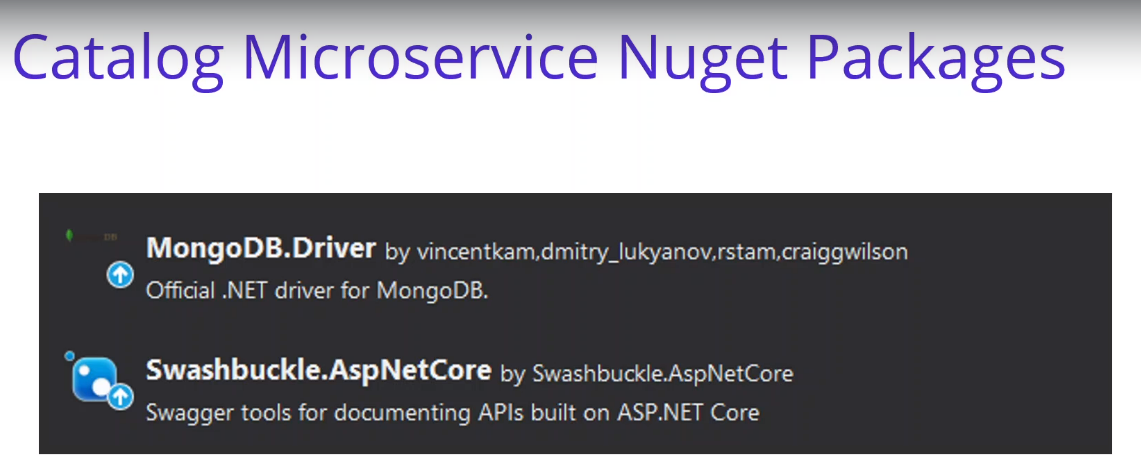


**We are using clean architecture**

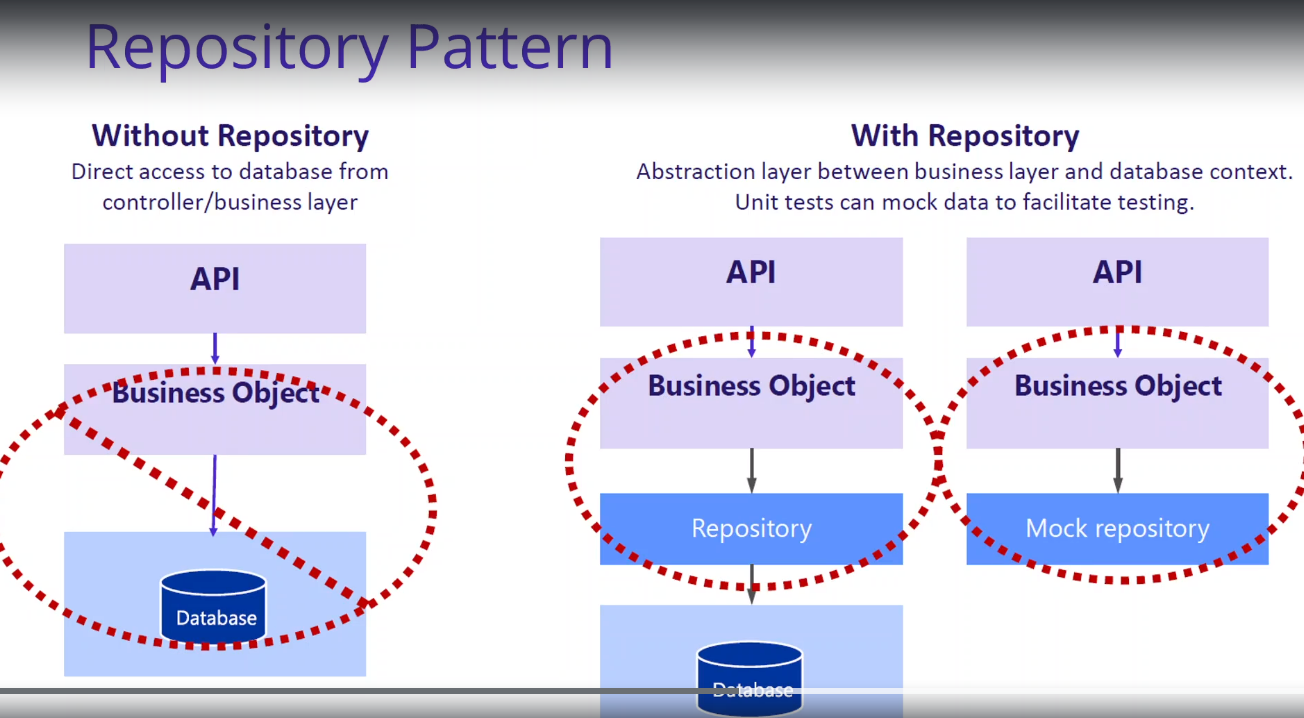
**Within it the API Layer only contains the endpoints and talking with Domain layer within CQRS**

**Domain layer contains the business rules and logic**

**Infrastructure layer responsible to communicate with database**



**Repository Pattern**



**1-Repository pattern is an abstraction between the business layer and database context layer (Middle layer between the above 2 layers).**

**2-Centerline the domain object make easy to maintain.**

**3-testing code is in easier way.**

**4-reducre application of code.**

**5-controller don’t care about what database you are using**

**Docker run –d –p 27017:27017 –name shopping-mongo mongo**

**Docker ps //list the Docker working containers**

**3fe91a216514 mongo "docker-entrypoint.s…" 4 hours ago Up 4 hours 0.0.0.0:27017->27017/tcp shopping-mongo**

**Docker stop 3fe9 //stop the Docker container**

**Docker start 3fe9 //start the Docker container**

**Docker Compose**

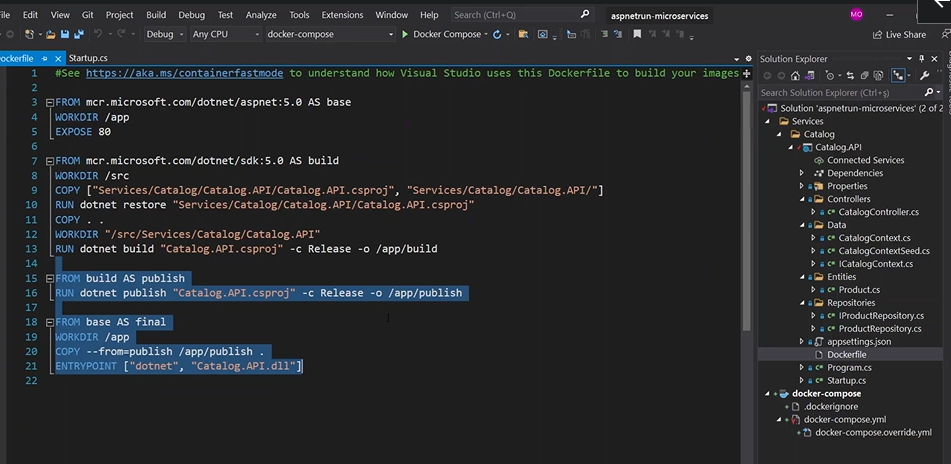
**Notes:-**

**1-Docker compose: its Docker tools that enable to containerize multi-application on one configuration and run and manage communication between them in single command**

**2-on vs > project > right click > Add > Container Orchestrator support > choose Linux**

**(It’s responsible to configure the containers and create and manage it from VS)**

**(Within it you can debug code with the generated containers by this way)**



**It will generate Docker-compose which is looking for the Docker file that contains the configuration for the container**

**Configuration section  
1-Pulling Images**

**2-Running Images into containers**

**//it will pulling the image aspnet 5 from Docker hub**

**//WORKDIR: it’s the destination folder that which the Docker pulling the image**

**FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base**

**WORKDIR /app**

**EXPOSE 80**

**FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build**

**WORKDIR /src**

**//it will build image called Catalog.API.csproj by copy the content of the folder of //src/Catalog.API and then generate image on release mode**

**COPY ["Catalog.API/Catalog.API.csproj", "Catalog.API/"]**

**RUN dotnet restore "Catalog.API/Catalog.API.csproj"**

**COPY . .**

**WORKDIR "/src/Catalog.API"**

**RUN dotnet build "Catalog.API.csproj" -c Release -o /app/build**

**//it will generate container from the target image**

**FROM build AS publish**

**RUN dotnet publish "Catalog.API.csproj" -c Release -o /app/publish**

**//it will running the container from the entry point Cataglo.API.dll > program.cs**

**FROM base AS final**

**WORKDIR /app**

**COPY --from=publish /app/publish .**

**ENTRYPOINT ["dotnet", "Catalog.API.dll"]**

**Volumes: is an important feature which is file system store if any reason the the container of mongo not working it will store on the file system called mongo\_data**

**docker-compose.yml**

**version: '3.4'**

**services:**

**catalogdb:**

**image: mongo**

**catalog.api:**

**image: ${DOCKER\_REGISTRY-}catalogapi**

**build:**

**context: .**

**dockerfile: Catalog.API/Dockerfile**

**volumes:**

**mongo\_data:**

**docker-compose-override.yml**

version: '3.4'

services:

catalogdb:

container\_name: catalogdb

restart: always

ports:

- "27017:27017"

volumes:

- mongo\_data:/data/db

catalog.api:

container\_name: catalog.api

environment:

- ASPNETCORE\_ENVIRONMENT=Development

- DatabaseSettings:ConnectionString=mongodb://catalogdb:27017

depends\_on:

- catalogdb

ports:

- "8000:80"

**//on each container we specify the port for the container and the protocol port that can be access like catalog.api container on port 8000 and access through http port**

**Catalogdb container on port 27017 and access through 27017 port**