Building Microservices on .Net which used Asp.Net Web API, Docker, RabbitMQ,Ocelot API Gateway, MongoDB,Redis,SqlServer

**Create Repository in GIthub**

* Created repository “aspnetcore-microservices” in Github (<https://github.com/yogesh-narkhede>)
* Clone repository using Visual Studio.

**Create Solution “aspnetcore-microservices”**

**Catalog Service:**

* Created folder structure for src/Services/Catalog.
* Created new project Catalog.API under src/Services/Catalog.
* Created Catalog.API profile under Properties => Debug with URL: <http://localhost:5000>

**MongoDB:**

* Right clicked on Solution and Open in Terminal.
* Go to hub.docker.com and searched for Mongo to get mongo pull image command
* Run Command: docker pull mongo
* Show running images: docker ps
* Show stopped images : docker ps –a
* Run mongo docker image using command: docker run -d -p 27017:27017 --name shopping-mongo mongo
* Run command to check logs: docker logs –f shopping-mongo
* Interactive terminal for MongoDB connection: docker exec -it shopping-mongo /bin/bash
* List command to show all files: ls
* To run mongo command: mongo
  + use CatalogDb
  + db.createCollection(‘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}])
  + db.Products.find({}).pretty()
  + db.Products.remove({})

**Add Nuget Packages:**

* Visit website to get install command: <https://www.nuget.org/packages/MongoDB.Driver/>
* Open package manager and execute command: Install-Package MongoDB.Driver
* Update-Package -ProjectName Catalog.API

**Implementation:**

* Created Entities folder and Product class with all fields.
* Created Data folder and ICatalogContext, CatalogContext and CatalogContextSeed classes with implementation.
* Created Repositories folder and ProductRepository interface and class with implementation.
* Created CatalogController with CRUD APIs implementation.
* Add docker-compose
  + Right click on project and Add “Container Orchestrator Support”.
  + Select Docker Compose.
  + Select Operating System as Linux
  + Added/updated catalogdb & catalog.api configuration in docker-compose.yml and docker-compose.override.yml

**Run docker-compose:**

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

**Down docker-compose:**

docker-compose -f .\docker-compose.yml -f .\docker-compose.override.yml down

**Docker commands:**

* docker ps –aq
* Stop all running conatiners: docker stop $(docker ps -aq)
* Remove all containers: docker rm $(docker ps -aq)
* Remove all images: docker rmi $(docker ps -aq)
* Remove unnamed resources: docker system prune
* Show running images: docker ps
* Show stopped images : docker ps –a
* Start existing images: docker start <containerId>
* Stop running container: docker stop <containerId>
* Remove container: docker rm <containerId>
* Show docker images: docker images
* Remove container image: docker rmi <containerId>

**Test swagger in browser:**

<http://localhost:8000/swagger/index.html>

**Mongo GUI options for MongoDb Docker Image**

* <https://hub.docker.com/r/mongoclient/mongoclient>
* Run command in Terminal: docker run -d -p 3000:3000 mongoclient/mongoclient
* Open url in browser: <http://localhost:3000/>

**Basket Service:**

* Created folder structure for src/Services/Basket.
* Created new project Basket.API under src/Services/Basket.
* Created Basket.API profile under Properties => Debug with URL: <http://localhost:5001>

**Redis Database:**

* Right clicked on docker-compose and Open in Terminal.
* Go to hub.docker.com and searched for Redis to get redis pull image command
* Run Command: docker pull redis
* Run redis docker image using command: docker run -d -p 6379:6379 --name aspnetrun-redis redis
* Interactive terminal for Redis connection: docker exec -it aspnetrun-redis /bin/bash
* Run command to enter into cli: redis-cli
  + set key value
  + get key
  + set name Yogesh
  + get name

**Add Nuget Packages:**

* Visit website to get install command: https://www.nuget.org/packages/Microsoft.Extensions.Caching.StackExchangeRedis/
* Open package manager and execute command: Install-Package Microsoft.Extensions.Caching.StackExchangeRedis
* Update-Package -ProjectName Basket.API

**Implementation:**

* Created Entities folder and ShoppingCart and ShoppingCartItem classes with all fields.
* Created Repositories folder and BasketRepository interface and class with implementation.
* Created BasketController with CRUD APIs implementation.
* Register BasketRepository interface and class in startup class.
* Add docker-compose
  + Right click on project and Add “Container Orchestrator Support”.
  + Select Docker Compose.
  + Select Operating System as Linux
  + Added/updated basketdb & basket.api configuration in docker-compose.yml and docker-compose.override.yml
  + Added portainer configuration to pull portainer/portainer-ce docker image in docker-compose.yml and docker-compose.override.yml
  + Run command: docker-compose -f .\docker-compose.yml -f .\docker-compose.override.yml up -d
  + Access portainer using url: <http://localhost:9000>

**Discount Service:**

* Created folder structure for src/Services/Discount.
* Created new project Discount.API under src/Services/Discount.

Created Discount.API profile under Properties => Debug with URL: [http://localhost:5002](http://localhost:5001)

**PostgreSQL Database:**

* Right clicked on docker-compose and Open in Terminal.
* Go to hub.docker.com and searched for postgres to get postgres pull image command
* Run Command: docker pull postgres
* Added discountdb configuration in docker-compose.yml and docker-compose.override.yml
* Added pgadmin configuration in docker-compose.yml and docker-compose.override.yml
* Run command: docker-compose -f .\docker-compose.yml -f .\docker-compose.override.yml up –d
* Access pgadmin using url: <http://localhost:5050/>
* Login with email address and password mentioned in docker-compose.override.yml file.
* Add Server
  + Set Name: DiscountServer
  + Go to connection tab
  + Enter discountdb as hostname
  + Enter username and password which is mentioned in docker-compose.override.yml file.
* Queries:
  + CREATE TABLE Coupon(

ID SERIAL PRIMARY KEY NOT NULL,

ProductName VARCHAR(24) NOT NULL,

Description TEXT,

Amount INT

);

* INSERT INTO coupon (ProductName, Description, Amount) VALUES ('IPhone X', 'IPhone Discount', 150)
* INSERT INTO coupon (ProductName, Description, Amount) VALUES ('Samsung 10, 'Samsung Discount', 100)
* SELECT \* FROM public.coupon ORDER BY id ASC

**Add Nuget Packages:**

* Open package manager and execute commands:
  + Install-Package Npgsql
  + Install-Package Dapper
  + Update-Package -ProjectName Discount.API

**Implementation:**

* Created Entities folder and Coupon class with all fields.
* Created Repositories folder and DiscountRepository interface and class with implementation.
* Created DiscountController with CRUD APIs implementation.
* Register DiscountRepository interface and class in startup class.
* Add docker-compose
  + Right click on project and Add “Container Orchestrator Support”.
  + Select Docker Compose.
  + Select Operating System as Linux
  + Added/updated discount.api configuration in docker-compose.yml and docker-compose.override.yml
  + Run command: docker-compose -f .\docker-compose.yml -f .\docker-compose.override.yml up -d

**Resources:**

* <https://github.com/>
* <https://hub.docker.com>
* <https://www.nuget.org/packages>
* <https://docs.portainer.io/v/ce-2.11/start/intro> (admin/admin@123)
* <https://www.pgadmin.org/>