DeliVeggie

Description

The application was created as per the architecture mentioned in the document. However, there are some tweaking done in order to simplify and create a structured application. There are 5 main components in this architecture

1. DeliVeggie.Gateway
2. DeliVeggie.Microservice
3. DeliVeggie.Models
4. DeliVeggie.EasyNetQ.RabbitMQ
5. DeliVeggie.UI

DeliVeggie.EasyNetQ.RabbitMQ and DeliVeggie.Models are created as shared libraries that could be leveraged by other applications in the platform. They are packed and pushed into nuget.org. DeliVeggie.EasyNetQ.RabbitMQ contains the EasynetQ implementation for request-response model over RabbitMQ. DeliVeggie.Models, as the name suggest contain model that could be shared across the platform.

DeliVeggie.Gateway is the front facing API service, which can be called be the UI application or act as data endpoint for other application. This application internally request data through rabbitMQ, Which will be server later by DeliVeggie.Microservice.

DeliVeggie.Microservice is the microservice, which will respond to the request coming from the rabbit. This application is a console application, which is made to run forever using infinite loop strategy. I have inserted some dummy records to mongo, at the startup of the application. This is a one time process.

DeliVeggie.UI provides the UI for the application. This is made in the latest Angular framework.

Local Testing

We can test applications locally, by running DeliVeggie.Gateway, DeliVeggie.Microservice and DeliVeggie.UI.

Docker

I have added docker support for all the applications. There is *docker-compose* file located at the root of DeliVeggie.Microservice. You can run all the required applications and necessary infrastructure (Rabbit, Mongo) by invoking the command ***docker-compose up*** from the specific folder.
Note that there might be some differences in the folder naming/ structure, if you have downloaded it from Github. Please ensure the path to the docker file is correct before running the application.

Kubernetes Deployment

I have also added kubernetes deployment support for the applications. Make sure you have kubernetes cluster configured and have *kubectl* installed. There is folder named Deployment at the root of DeliVeggie.Microservice. It contains three *yaml* files. Two of them are for the infrastructure(Mongo, Rabbit). The other one contains deployment details related to other DeliVeggie applications.
The images of DeliVeggie application is pushed to DockerHub, so that it could be downloaded when deploying.

Use the following command to deploy
kubectl apply -f <DEPLOYMENT_FILE>.yaml

Replace <DEPLOYMENT_FILE> with the actual deployment file name.

Start the kubernetes dashboard by giving the command ***kubectl proxy*** and going to

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

or use ***kubectl get pods*** to see the pods in the cli.

GitHub/Docker Link

Backend - https://github.com/Harikrishnan9847/DeliVeggie-Backend

Frontend - https://github.com/Harikrishnan9847/DeliVeggie-Frontend

Docker Repository - https://hub.docker.com/r/harikrishnan1996/deliveggie-gateway

https://hub.docker.com/r/harikrishnan1996/deliveggie-ui

https://hub.docker.com/r/harikrishnan1996/deliveggie-microservice