Hospital Management System:

Folder Structure:



Dockerizing the application as:   
1. Docker compose.yml file containing locations of docker files for frontend and backend.

2. Separate container shall be running for MongoDB database.

docker build -f frontend.dockerfile -t hospitalfrontend

docker run -p 80:80 hospitalfrontend-image:0.0.1

**docker build -f backend.dockerfile -t hospitalbackend-image:0.0.1 .**

docker run -p 5000:5000  **hospitalbackend-image:0.0.1**

docker run -p <local host port> : <container port>  **hospitalbackend-image:0.0.1**

OPTIMIZING CURRENT IMAGES USING KUBERNETES:

* Build the image as in format: docker build -f backend.dockerfile -t **maareech/practice-web-apps:1.0.0** . **maareech/practice-web-apps:1.0.0 being username/repo name in docker hub & tagging is imp.**
* Push the image in the repo created in docker hub as: docker push maareech/practice-web-apps:1.0.0

Push only the custom images and not the database ones.

Dockerhub recognizes only tags, hence tags shall be different for different images else the images shall be overwritten

**maareech/practice-web-apps:1.0.0 backend image**

**maareech/practice-web-apps:2.0.0 frontend image**

Quick Look:  
  
  
Docker can reuse layers from the first image in the second image due to identical content. This reduces push time by avoiding the need to upload reused layers.

Deployment in Kubernetes:  
  
Pre-requisite: Kubectl & Kubernetes installed in docker desktop

Create Kube maniffest files for each deployment & service objects eg: 3 files for frontend, backend and database; respectively.   
Contains container, podd details and their Service objects connecting the pods

Enable Kubernetes in docker desktop for a single node cluster "docker-desktop"

Cmds to initialize the same:

1. kubectl config use-context docker-desktop

2. kubectl cluster-info ; to verify cluster info

3. kubectl config get-contexts ; verify context

4. kubectl get nodes ; get nodes details

Run the 4 cmds in that order

Deploy the manifest files in kube:

kubectl apply -f mongodb-deployment.yaml

kubectl apply -f backend-deployment.yaml

kubectl apply -f frontend-deployment.yaml

kubectl get all ; to verify the manifests deployed ;

kubectl apply -f K8S (folder name) to deploy all at once

Accessing the frontend:

kubectl get svc frontend ;

returns the details related to the service object deployed:

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

frontend NodePort 10.96.232.125 <none> 80:30007/TCP 10m

Delete resources: kubectl delete -f frontend-deployment.yaml

Stop & delete minikube cluster: minikube stop & minikube delete

Troubleshooting:

1. kubectl logs <pod name> -- checks logs of the pods during start/restart

2. curl for individual urls(localhost:30007 for frontend, backend:5000 and mongodb:27017) for each connection - works in bash within gitbash

3. if frontend does not resolve backend pod:

check if backend pods and svcs are running;

for dns resolution check:

kubectl exec -it <frontend-pod-name> -- sh and ctrl plus D to exit the terminal

nslookup backend ; in that order

we