Tools:

* Kind
* Docker Desktop

Scenario:

You have a project divided in 3 parts frontend, backend and data base needed to be deployed on docker.

Expect to learn:

Communication between docker containers.

GitHub Repository:

<https://github.com/AlphaTab435/Kubernetes-Communication-Between-Pods>

Docker Images:

You can refer to following repository to create docker images and learning about communication between docker containers that will be needed before understanding current topic.

https://github.com/AlphaTab435/Communication-between-Docker-containers

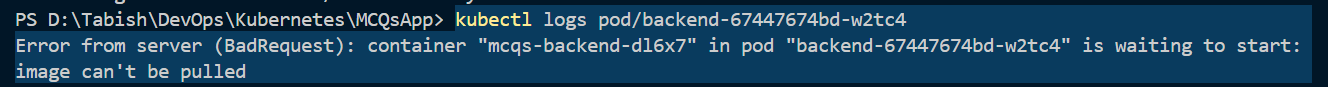
I am going to use images that I already created in that exercise.

Kind:

Using Kind (Kubernetes IN Docker) is a great way to create a local Kubernetes cluster using Docker containers.

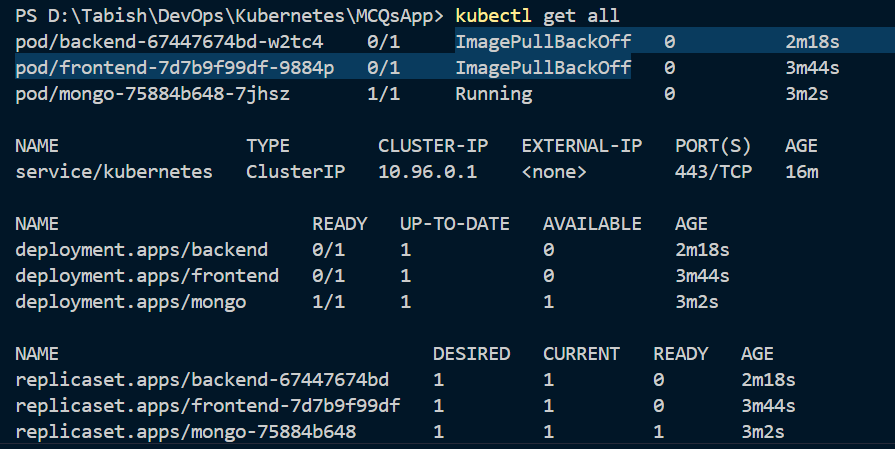
For troubleshoot:

Use kubecl logs podName for debugging



Error:

If you are facing ImagePullBackOff like following image that might be due to you image is not public or you have not pushed it to docker hub



Methods:

Manual creation through cli

Using deployment.yaml file

Manual creation through cli:

kind create cluster // To create Kubernetes cluster

Now creating deployments using docker hub public images

kubectl create deployment mongo --image=mongo

kubectl create deployment frontend --image=muhammadtabish/mcqs\_frontend\_already\_build

Note: backend tightly depends on database so we will deploy backend on later stage

Pod communication:

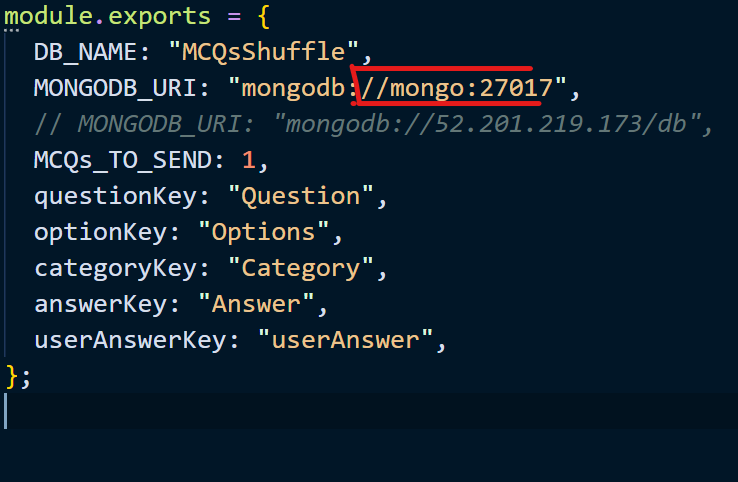
For pod communication we need services.

* ClusterIP # Internal service for accessing from inside the cluster
* NodePort # External service for accessing from outside the cluster

Note:

--name while creating mongo service is important

If you have checked my Communication-between-Docker-containers repository link provided at start of document. You should check backend\src\constants.js



You should use same name for your database service.

Create services:

kubectl expose deployment mongo --name=mongo --port= 27017 --type= ClusterIP

kubectl expose deployment frontend --name= frontend --port= 80 --type= NodePort

if above don’t work try using --target-port like used in below command

kubectl expose deployment mongo --type=ClusterIP --name=mongo --port=27017 --target-port=27017

For backend:

Now we can create backend deployment and service

kubectl create deployment backend --image=muhammadtabish/mcqs\_backend

kubectl expose deployment backend --name= backend --port= 3000 --type= NodePort

Using deployment.yaml file:

Instead of doing all above steps you can use deployment.yaml file that I have provided in github repository after creating cluster

kind create cluster // To create Kubernetes cluster

kubectl apply -f deployment.yaml // To apply yaml cofiguration

Port mapping:

You can use following on windows PowerShell for port mapping:

Start-Job -ScriptBlock {kubectl port-forward service/backend 3000:3000}

Start-Job -ScriptBlock {kubectl port-forward service/frontend 80:80}

Use following to get, stop, remove a job:

Get-Job

Stop-Job Id

Remove-Job Id

Reminder:

Feel free to delete resources after you have done testing. Commands will be similar to following

kubectl delete -f filename

kind delete cluster