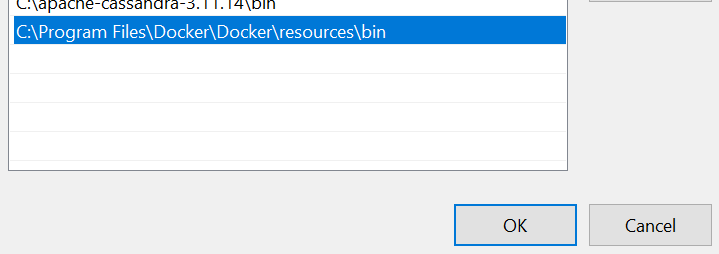
**Docker**

Install the docker and add the path to set environment variable:



Edit path and set docker environment like below:



Check the version:



After installation go to project home directory and create the docker file like “Dockerfile” and add the configuration like below:

FROM openjdk:<java version>

EXPOSE <port number to expose your application>

ADD target/<war file with extension> <image name>

ENTRYPOINT ["java","-jar","/<image name>"]

E.g.

FROM openjdk:8

EXPOSE 8086

ADD target/DockerDemoProject-0.0.1.war demo-docker-app

ENTRYPOINT ["java","-jar","/demo-docker-app"]

**Go to project home directory and open command prompt to execute the docker file**:

(Before executing the below command make sure the war/jar has already created in target folder)

**docker build -t <image name> .** (. Uses for root directory)

After creating an image verify the image on docker desktop support. And then run the image on container:

docker run -p <container port>:<image port> <image name>

Now, Hit the end points and verify the response.

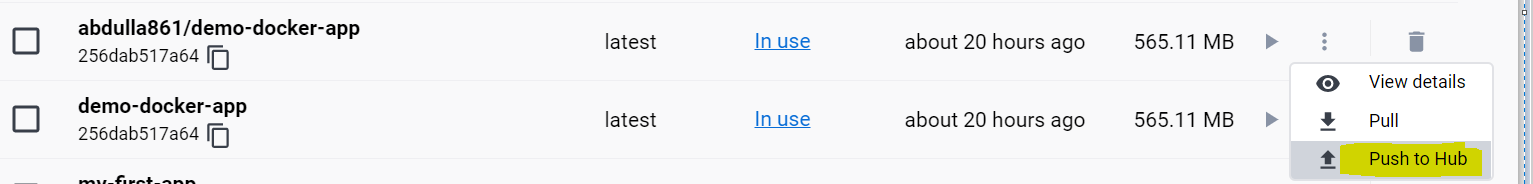
To list the images on command window, use below command:

docker image ls

**Push and pull the image from docker hub**:

Create an account on docker hub and create the repository and use the same credential for docker desktop support:

And push the image to docker hub from docker desktop support:



Or by below command:

docker push <hub-user>/<repo-name>:<tag>

**Kubernetes (Digital ocean)**

Download kubectl (binary) and add the path into the environment path variable.

Create and account on digital ocean and provide the payment detail. After deducting some amount, it will be reversed back that amount.

1. Before starting Kubernetes cluster make sure the docker image has already pushed to docker hub.
2. Create Kubernetes cluster on digital ocean and after creating download the config file.
3. Create the folder “.kube” into the user folder like C:/user/<user-name> and put the downloaded cofig file.
4. Create the .yml file and add the configuration. See attached “k8s-pod.yml”
5. Open command prompt window from project home directory and Create the name space:

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> create ns <namespace name>

1. Get the list of name space:

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> get ns

1. Set context

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> config set-context --current --namespace=<namespace name>

1. Apply the .yml file to create pods and replica set:

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> apply -f <yml file name with .yml extension>

1. Get the list of pods that you have created in previous step:

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> get pods

1. Check the logs of application:

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> logs <pods-name>

1. Finding service to hit the endpoints of API. Create and service yml file. See the attached “k8s-deployment-svc.yml”
2. Apply the service yml file to discover the service and create the load balancer:

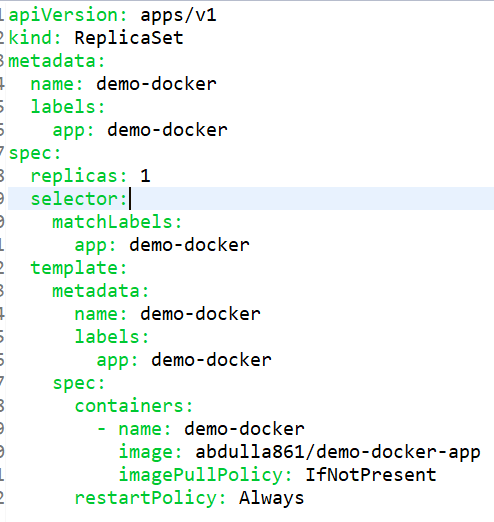
kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> apply -f <yml file name with .yml extension>

1. Verify on digital ocean the load balancer has been created and able to get the external IP. Excute the below command to get the external IP on command window:

kubectl --kubeconfig=C:/Users/mabdulla2/.kube/<config file name> get svc

1. Now, Hit the endpoints and verify the API.

k8s-pod.yml example:



k8s-deployment-svc.yml example for service

