**ARB Application**

**ARB Application MicroService**

* **ARBWeb**
* **ARBGateway**
* **ARBDashBoard**
* **ARBUserManagement**

**Prequisites: (Asp . Net Core Web Application)**

* Docker (aspnetcore-build:2.0, aspnetcore:2.0, nginx)
* Kubernetes (Ingress-service)
* Visual Studio 2017 (Asp .Net Core Application)
* **.** Net Core Sdk 2.0 (Ubuntu)

**STEP1: Install docker in ubuntu machine**

https://docs.docker.com/

After installing ,check for the docker version using the below command.

#sudo docker --version

**STEP 2: Create a docker file required for the application**

**DockerFile:(Angular JS)**

FROM nginx

COPY ./ /usr/share/nginx/html

EXPOSE 80

**DockerFile:(.Net Core Application)**

FROM microsoft/aspnetcore-build:2.0 AS build-env

WORKDIR /app

# copy csproj and restore as distinct layers

COPY \*.csproj ./

RUN dotnet restore

# copy everything else and build

COPY . ./

RUN dotnet publish -c Release -o out

# build runtime image

FROM microsoft/aspnetcore:2.0

WORKDIR /app

COPY --from=build-env /app/out .

ENTRYPOINT ["dotnet", "Application\_Name.dll"]

**STEP 3: Build the application as a docker image**

# sudo docker build –t <ImageName> .

This command will build the docker image as which name your menting. ‘ . ‘ indicates the current session .

Note:

Execute this command where the docker file for the application is located.

Image will be available at the container.

**To view the built image.**

#sudo docker images

**To remove the built image.**

**#** sudo docker rmi <Docker\_image\_ID>

**STEP 4: Download and Install docker local registry**

# sudo docker run -d -p 5000:5000 --restart=always --name registry registry:2

**STEP 5: Create tag for the built image**

# sudo docker tag <ImageName>:latest localhost:5000/< ImageName>

**STEP 6: Push the image into the local docker registry**

#sudo docker push localhost:5000/ <ImageName>

This command will push the image (localhost:5000/< ImageName>) into the local docker registry.

**STEP 7: Create a secret name for registry**

#sudo kubectl create secret docker-registry <secret Name> --docker-server=<your-registry-server> --docker-username=<your-name> --docker-password=<your-pword> --docker-email=<your-email>

This command will create a secret name for registry.

**STEP 8: Create .yaml file for kubernetes Deployments**

**Sample Deployment.yaml file**

apiVersion: apps/v1beta1

kind: Deployment

metadata:

name: <Deployment\_Display\_Name>

namespace: <Namespace\_Name>

spec:

replicas: <No\_of\_replicas>

template:

metadata:

labels:

app: <Deployment\_Name>

spec:

containers:

- name: <Container\_Name>

image: <Image\_Repository\_Name>

ports:

- containerPort: <Port>

imagePullSecrets:

- name: <Secret\_Name>

---

kind: Service

apiVersion: v1

metadata:

name: <Service\_Display\_name>

namespace: <Namespace\_Name>

spec:

selector:

app: <Service\_Name>

ports:

- protocol: TCP

port: <Application\_Port>

targetPort: <Container\_Port>

Note : Deploymet\_Name and Service\_Name Must be same.

**STEP 10: Deploy application into kubernetes**

#sudo kubectl apply -f Deployment.yaml

This command will deploy the application into the kubernetes.

Note: Execute this command from where the Deployment.yaml has been located.

**STEP 12: To see the status of the application**

**#**sudo kubectl get deployments

This command will list the application deployed in kubernetes.

#sudo kubectl get services

This command will list the services in kubernetes.

Note : To run this application use Ingress-Service.yaml to reach our host machine