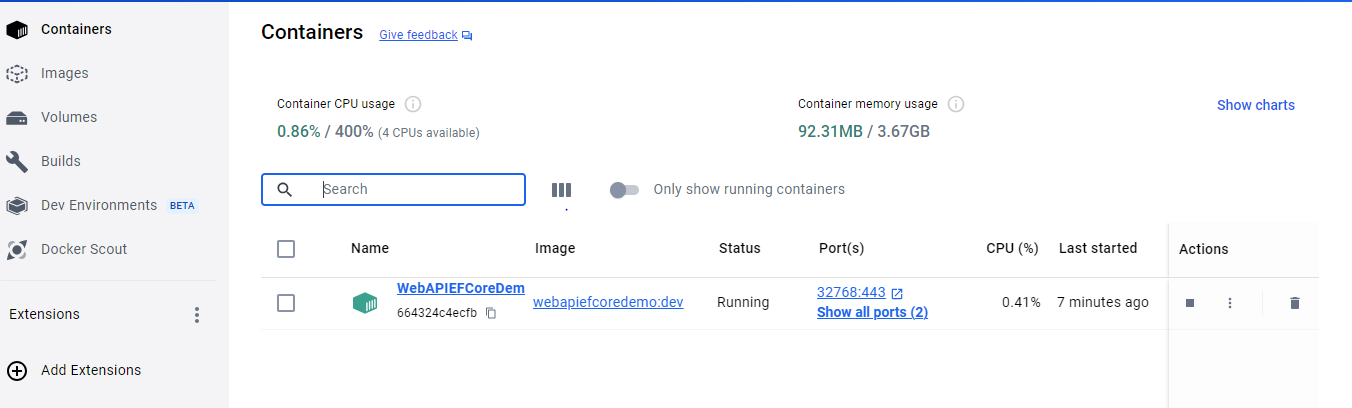
Step1: Created a sample Web API application target framework .net6

Step2: Added docker support file to implement containerization

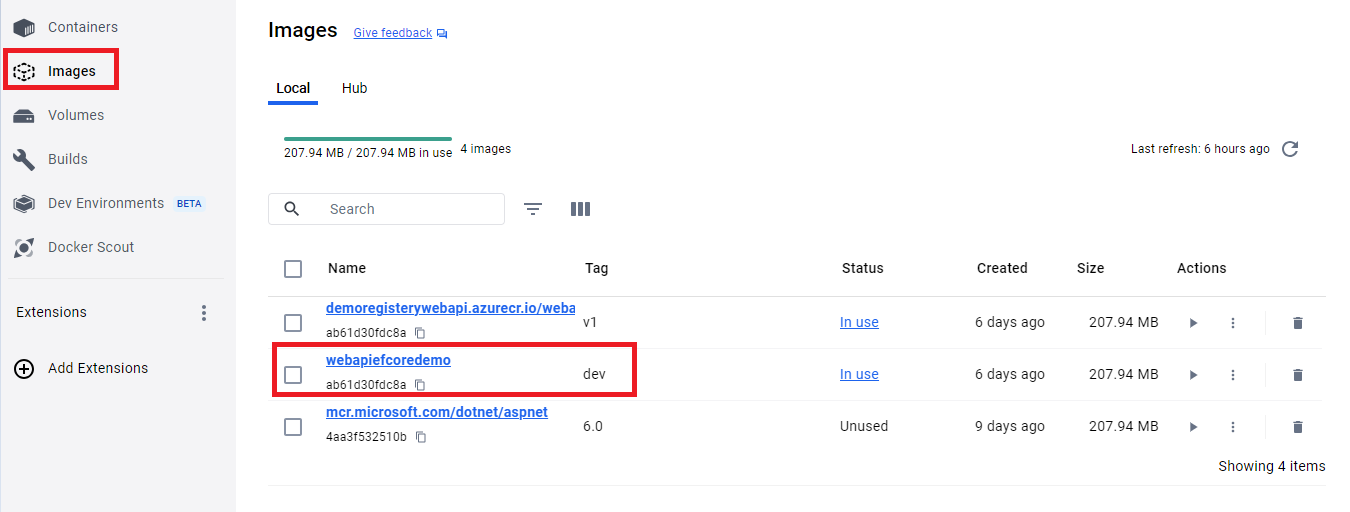
Step3: Build application on Docker profile

Step4: Container and image will be created in docker desktop. Application converted as containerization application.

Containers Tab



Images Tab

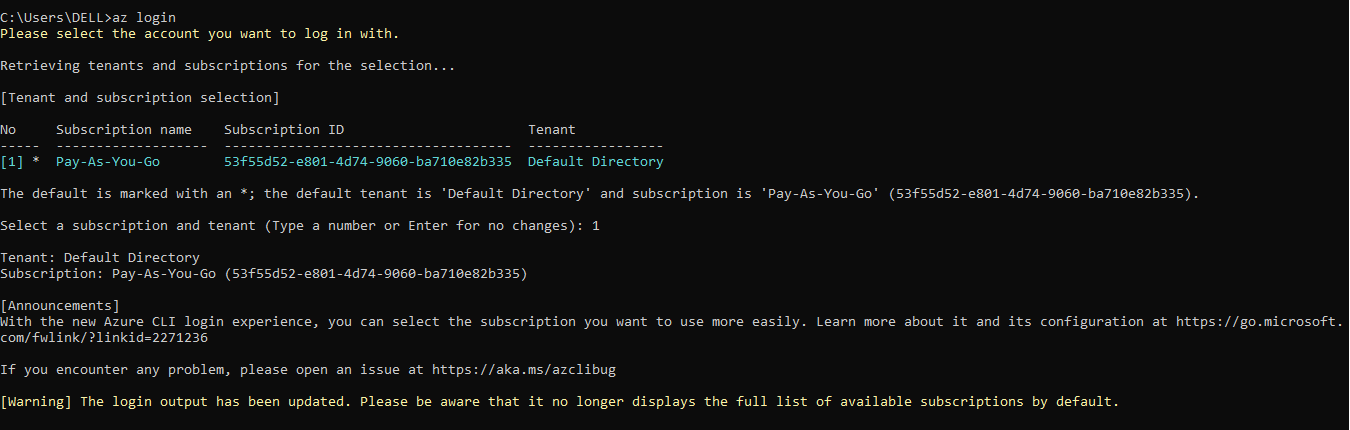


Step5: Push the image to Azure container registry

1. Login to azure portal make sure having a one active subscription
2. Configure Azure CLI, run (command: az) on cmd after configuration to check Azure CLI Configure correctly. We should get below output on cmd.



1. Run a command to login on azure (command : az login). It will ask to choose subscription.



1. Create resource group

Command: az group create - l <<location>> -n <<group\_name>>

Example: az group create -l CentralIndia -n AKSDemo

1. Create registery

Command: az acr create --resource-group <<resource\_group\_name>> --name <<registery\_name>> --sku Basic

Example: az acr create --resource-group AKSDemo --name demoregisterywebapi --sku Basic

1. Login to registery

Command: az acr login –name <<registery\_name>>

Example: az acr login --name demoregisterywebapi

1. Tag image with new tag (this is optional if have to proceed with existing tag). Go to project repository folder then run below command

Command: docker tag <<image\_name\_with\_tag>> <<azure\_container\_registery\_url>>/<<image\_name\_with\_new\_tag>>

Example: docker tag webapiefcoredemo:dev demoregisterywebapi.azurecr.io/webapiefcoredemo:v1

1. Push image to container registery

Command: docker push <<azure\_container\_registery\_url>>/<<image\_name\_with\_tag>>

Example: docker push demoregisterywebapi.azurecr.io/webapiefcoredemo:v1

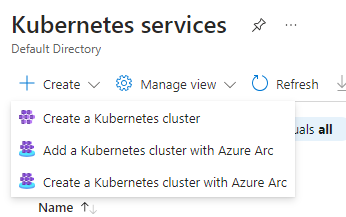
1. Check image pushed correctly by listing down images available in registry

Command: az acr repository list --name <<azure\_container\_registery\_name>> --output table

Example: az acr repository list --name demoregisterywebapi --output table

Step6: Create Azure kubernetes Cluster

1. Login to Azure portal
2. Go to kubernetes Service from search resource
3. Click on create kubernetes cluster



1. Fill out the all necessary information like cluster name, subscription, nodes, and in the integrations tab choose the container registery that we create in step5.
2. Login to cluster to deploy the application AKS
   1. Run the command to set subscription

Command: az account set –subscription <<subscription\_id>>

* 1. Run the command to get into cluster

Command: az aks get-credentials –resource-group <<resource\_group\_name>> --name <<cluster\_name>>

* 1. Run the command to verify we can execute kubectl commands

Command: kubectl get deployments –all-namespaces=true

1. Deploy the yml files
   1. Create pod.yml file and save into project repository

apiVersion: v1

kind: Pod

metadata:

name: weather-forecast-app

labels:

app: weatherforecast-aks

component: weatherforecast-app

spec:

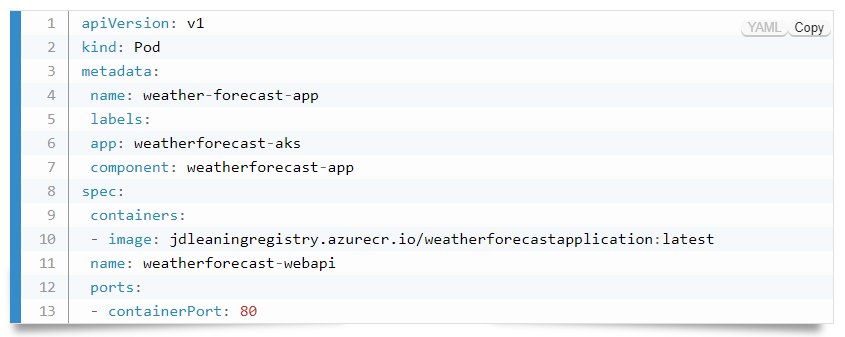
containers:

- image: jdleaningregistry.azurecr.io/weatherforecastapplication:latest

name: weatherforecast-webapi

ports:

- containerPort: 80



* 1. Create service.yml file and save into project repository

apiVersion: v1

kind: Service

metadata:

labels:

app: weatherforecast-aks

name: weatherforecast-aks

spec:

ports:

- port: 8080

protocol: TCP

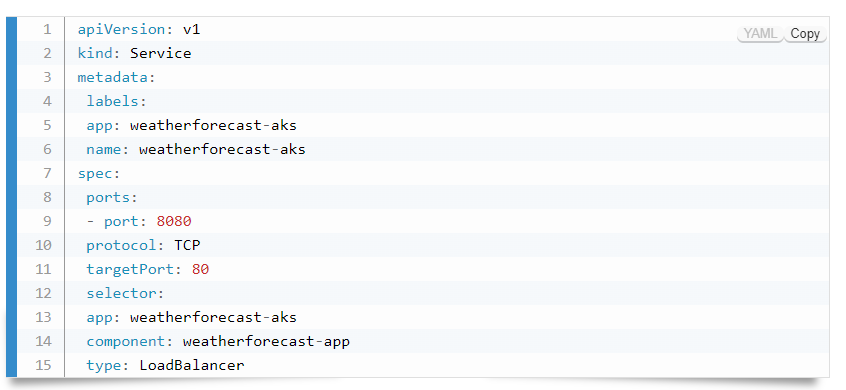
targetPort: 80

selector:

app: weatherforecast-aks

component: weatherforecast-app

type: LoadBalancer



* 1. Deploy the file, Go to project repository folder where yml files present and run below command to deploy both pod.yml and service.yml files

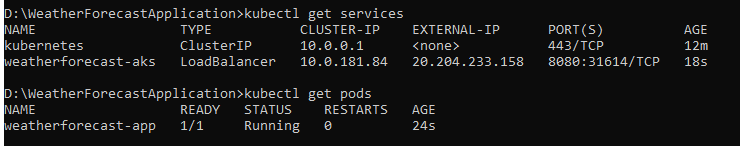
Command1: kubectl create –f pod.yml

Command2: kubectl create –f service.yml

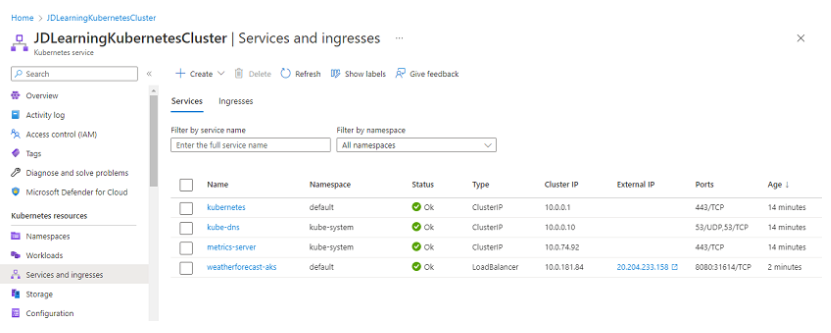
* 1. Verify deployment successfully completed run below command and verify output

Command1: kubectl get services

Command2: kubectl get pods



1. Test the application running
   1. Go to cluster and go inside the service and ingress, search application with application name and copy the External IP Address



* 1. Hit the url: http://<<external\_ip\_address>>:8080/weatherforecast to verify we are getting output

