**How to deploy the web API in the container instance / azure Kubernetes service (AKS)**

**Links for the container Instance :-**

**The Public links are :-** [20.241.221.60:5000/api/Order/GetAllOrders](http://20.241.221.60:5000/api/Order/GetAllOrders) **(or)** [pharmacywebapi.ddhce2d5gxgkgad0.eastus.azurecontainer.io:5000/api/Order/GetAllOrders](http://pharmacywebapi.ddhce2d5gxgkgad0.eastus.azurecontainer.io:5000/api/Order/GetAllOrders)

**Note :- The links were working during the document creation later it will stop to work as it was created on the free tier subscription.**

**You can refer the screen shots attached in case the links were expired.**

**Deploy to container Instance: -**

**Step 1: -** You should have a running web API on your machine (visual studio) and initially you have to change it into a docker image.

To change it to a image we have to provide a certain details to the web API.

So, the web API contains a docker file in which it contains all the information regarding the sdk, port exposed, project name etc.

**Note: - add the docker file into the publish file in the project**

**Step 2: - You should have the docker engine on our machine running**. So as a possible way I have downloaded the docker desktop on my machine.

After installation you can check version using the following command on cmd.

**Graphical user interface, application, PowerPoint

Description automatically generated**

**Step 3: -** Run the command to build the docker image for your web API project.

To do this navigate the **cmd** path to the path of the project and navigate it further into the publish folder and execute the command.

**docker build -t dotnetapp .**

**step 4: -** You can use the command **docker images** to see the image.

**A screenshot of a computer

Description automatically generated with medium confidence**

From the above photo, **chittabathinikarthik/webapi** is the image name of the web API project.

**Step 5: -** You can push the image to the docker hub. Docker hub is nothing but a repository where we can store our images, and also we can get several public images like ubuntu, alphine and nginx etc at very small size you can just pull it from the hub and spin it in just seconds.

**Note: - You can also run the image In the local machine itself as the docker desktop is available in your pc.**

But as per our task we are gonna run it in container instance and azure kubernetes services in azure

Step 6: - There is a service in the azure called **Container Registry(create a container registry service in azure)** where you can store your images in the azure.

Here we will just login to azure using azure cli and pushes the image from our local machine/docker hub to the azure container registry. Just tag the existing image and provide azure location and push it.

Graphical user interface, text, application, email

Description automatically generated

**Step 7: -** From the above we can see the image is pushed to the azure container registry.

**Step 6: -** **Create a Container instance service in azure**. While creating itself in the process steps we can see the Image options there like azure container registry, docker hub you can pull image as per your choice.

The attached photo shows the available options to pull the image.

To see the docker hub option we can click on other registry option from the attached photo and can proceed by providing the required details of the image in your docker hub or you can just go with the azure container registry option too.

Graphical user interface, application

Description automatically generated

After review + create we can see the details on the container instance page to access the web API.

It provides the public Ip address/ FQDN (Fully qualified domain name) we can just use it to retrieve data as per our requirements.

We have to use the port number here that we exposed to get the result. I have exposed the port 5000 so I was using it.

Graphical user interface, application

Description automatically generated

From the above pic you can see the public ip address and FQDN.

**The Public links are :-** [**20.241.221.60:5000/api/Order/GetAllOrders**](http://20.241.221.60:5000/api/Order/GetAllOrders) **(or)** [**pharmacywebapi.ddhce2d5gxgkgad0.eastus.azurecontainer.io:5000/api/Order/GetAllOrders**](http://pharmacywebapi.ddhce2d5gxgkgad0.eastus.azurecontainer.io:5000/api/Order/GetAllOrders)

**Note :- The links were working during the document creation later it will stop to work as it was created on the free tier subscription.**

**You can refer the screen shots attached in case the links were expired.**

**Text

Description automatically generated with medium confidence**

This was on of the way to create the container and spin up the images on the containers. But it needs a lot of exposure to the containers topic we have to check the pods running and it needs some maintenance. But we can use the AKS we was the enhanced way of maintaining or using the containers.

Deploy to azure Kubernetes service (AKS): -

**Step 1: -** Follow the same steps as per the container instance task till pushing the image to the azure registry.

**Step 2: -** Create a Azure Kubernetes Service (AKS) and then connect kubectl to the Kubernetes cluster.(using the code provided there over in the aks cluster) using the azure cli.

Create a service principal so that the azure aks can directly get the image from the azure container registry.

**Kubectl: - Kubectl is a Kubernetes command line tool.**

**Service principal: - An azure principal is a security identity used by user-created app, services and automation tools to access specific azure resources.**

**Step 3: -** Then add the app.yaml and service.yaml files. These files provide the basic information to the kubectl.

In app.yaml information like where image is located, which port is exposed, what is the name, api version, deployment type all the meta deta is provided in the app.yaml file.

In service.yaml it is a load balancer that makes us to access the webapi and it contains the information like port exposed,

App version, type, name etc

**Step 3: -** we can now use the command to check the deployments, services, pods and the nodes running as shown in the fig with the below commands.

Text

Description automatically generated with medium confidence

You can see all the deployments, nodes, services, pods running from the above figure.

The links for the service running on the aks are: -

http://20.84.28.152: 5000/api/Order/GetAllOrders

This is all about the deployment of Web API on AKS.

Thank You