

How to deploy SpringBoot WebAPI to Azure Kubernetes(AKS)

IMPORTANT NOTE: see this repo for creating the Docker image (DockerFile) and Kubernetes manifest files (deployment.yml and service.yml):

https://github.com/luisccoco/SpringBoot_Sample4-Deploy-WebAPI-to-LocalKubernetes

1. Create the SpringBoot WebAPI with VSCode

https://github.com/luisccoco/SpringBoot_Sample2-created-WebAPI-with-VSCode

2. Create the Dockerfile

```
# Start with a base image containing Java runtime
FROM openjdk:11-jdk-slim as build

# Add Maintainer Info
LABEL maintainer="your_email@example.com"

# Add a volume pointing to /tmp
VOLUME /tmp

# Make port 8080 available to the world outside this container
EXPOSE 8080

# The application's jar file
ARG JAR_FILE=target/demoapi-0.0.1-SNAPSHOT.jar

# Add the application's jar to the container
ADD ${JAR_FILE} demoapi.jar

# Run the jar file
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/demoapi.jar"]
```

3. Create the Kubernetes manifest files (deployment.yml and service.yml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demoapi-deployment
spec:
```

```

replicas: 1 # The number of Pods to run
selector:
  matchLabels:
    app: demoapi
template:
  metadata:
    labels:
      app: demoapi
  spec:
    containers:
      - name: demoapi
        image: myregistryluiscoco1974.azurecr.io/springbootapi:latest # Replace with your D
        ports:
          - containerPort: 8080

apiVersion: v1
kind: Service
metadata:
  name: demoapi-service
spec:
  type: LoadBalancer # Exposes the service externally using a load balancer
  selector:
    app: demoapi
  ports:
    - protocol: TCP
      port: 80 # The port the load balancer listens on
      targetPort: 8080 # The port the container accepts traffic on

```

4. Set Azure Subscription

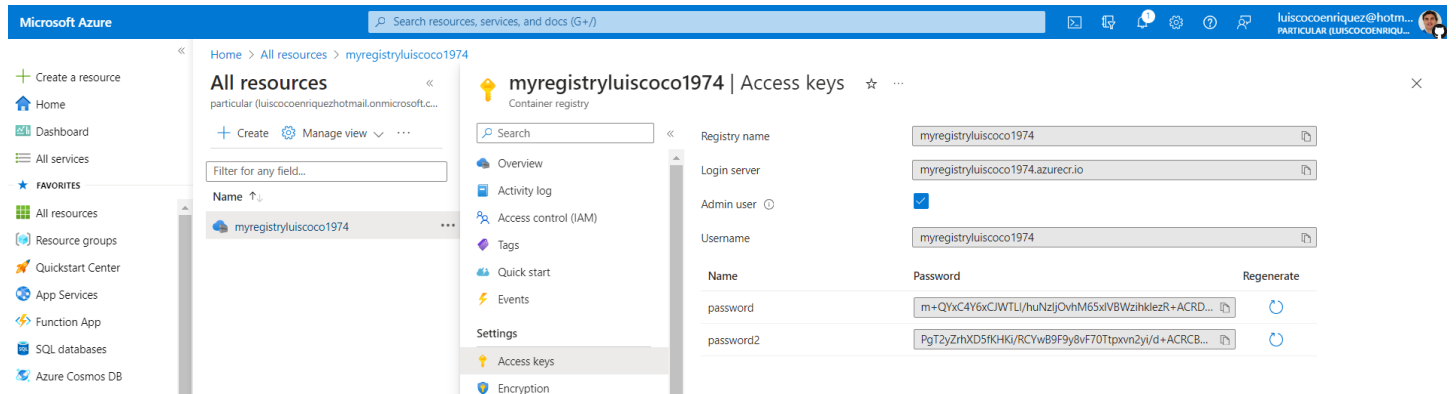
```
az account set --subscription "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

5. Create Azure Container Registry ACR

```
az acr create --resource-group myRG --name myregistryluiscoco1974 --sku Basic --location weste
```



Set as ACR Admin User



6. Create a Docker image and run it

Execute the following command to build the Docker image:

```
docker build -t springbootapi:latest .
```

Tag the Image for Azure Container Registry (ACR). Once the image is built, you'll need to tag it with the Azure Container Registry's address.

Assume your ACR login server is myregistryluiscoco1974.azurecr.io, type this command:

```
docker tag springbootapi:latest myregistryluiscoco1974.azurecr.io/springbootapi:latest
```

Run the Docker container, to start a container from your image, use the docker run command:

```
docker run -p 8080:8080 myregistryluiscoco1974.azurecr.io/springbootapi:latest
```

7. Log in to an Azure Container Registry (ACR) using Docker

7.1. Using Azure CLI

Install Azure CLI: Make sure you have the Azure CLI installed on your machine.

Log in to Azure: Open a terminal or command prompt and log in to your Azure account:

```
az login
```

Follow the instructions to complete the login process.

Get ACR Login Server Name: Retrieve your ACR's login server name. Replace myregistryluiscoco1974 with your actual ACR name.

```
az acr show --name myregistryluiscoco1974 --query loginServer --output table
```

It will return something like myregistryluiscoco1974.azurecr.io.

Login to ACR: Use the Azure CLI to log in to your ACR.

This command automatically uses the credentials stored from your Azure login to authenticate with ACR.

```
az acr login --name myregistryluiscoco1974
```

This command will handle the authentication with Docker for you.

7.2. Using Docker Login Command

Get ACR Credentials: First, retrieve your ACR's username and password. Replace myregistryluiscoco1974 with your actual ACR name.

```
az acr credential show --name myregistryluiscoco1974
```

Note the username and one of the passwords from the output:

```
PS C:\SpringBoot WebAPI> az acr credential show --name myregistryluiscoco1974
{
  "passwords": [
    {
      "name": "password",
      "value": "m+QYxC4Y6xCJWTLI/huNzIj0vhM65x1VBWzihk1ezR+ACRDK1Lb0"
    },
    {
      "name": "password2",
      "value": "PgT2yZrhXD5fKHKi/RCYwB9F9y8vF70Ttpxvn2yi/d+ACRCB+AYZ"
    }
  ],
  "username": "myregistryluiscoco1974"
}
```

Docker Login: Use the docker login command with your ACR's login server URL and the credentials you just retrieved.

```
docker login myregistryluiscoco1974.azurecr.io -u <username> -p <password>
```

```
docker login myregistryluiscoco1974.azurecr.io -u myregistryluiscoco1974 -p m+QYxC4Y6xCJWTLI/h
```

Replace and with the credentials from the previous step.

8. Push the Docker image to Azure Container Registry ACR

```
docker push myregistryluiscoco1974.azurecr.io/springbootapi:latest
```

This is the output we obtain:

```
PS C:\SpringBoot WebAPI> docker push myregistryluiscoco1974.azurecr.io/springbootapi:latest
The push refers to repository [myregistryluiscoco1974.azurecr.io/springbootapi]
632870e20190: Pushed
eb6ee5b9581f: Pushed
e3abdc2e9252: Pushed
eafe6e032dbd: Pushed
92a4e8a3140f: Pushed
latest: digest: sha256:bbc1283153d8ab75f0521d4bfff7b23a5e4d03feb5296ce703e097fcaa45952c6 size:
```

9. Create a new Azure Kubernetes Cluster AKS

```
az aks create ^
--resource-group myRG ^
--name myAKSClusterluiscoco1974 ^
--node-count 1 ^
--enable-addons monitoring ^
--generate-ssh-keys ^
--attach-acr myregistryluiscoco1974 ^
--location westeurope
```

10. How to deploy.NET 8 WebAPI Docker image deploy to Azure AKS

Authenticate with Azure: Make sure you are logged in to Azure CLI and have access to the subscription and resources.

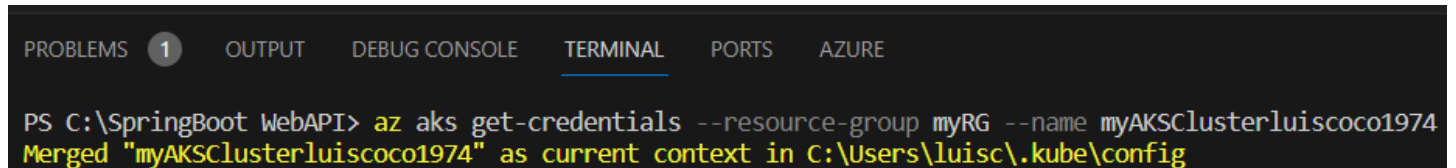
```
az login
```

Set the context to your AKS cluster: You need to get credentials for your AKS cluster and set the current context of kubectl to your cluster.

```
az aks get-credentials --resource-group <YourResourceGroup> --name <YourAKSClusterName>
```

```
az aks get-credentials --resource-group myRG --name myAKSClusterluiscoco1974
```

Replace and with your AKS resource group name and AKS cluster name, respectively.



```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
PS C:\SpringBoot WebAPI> az aks get-credentials --resource-group myRG --name myAKSClusterluiscoco1974
Merged "myAKSClusterluiscoco1974" as current context in C:\Users\luisc\kube\config
  
```

Create a Kubernetes Secret for ACR authentication: This step is crucial for allowing your AKS cluster to pull images from your private Azure Container Registry.

```
az aks update -n <YourAKSClusterName> -g <YourResourceGroup> --attach-acr <YourACRName>
```

```
az aks update -n myAKSClusterluiscoco1974 -g myRG --attach-acr myregistryluiscoco1974
```

11. Deploy the docker image stored in ACR to Azure AKS

```
kubectl apply -f deployment.yml
```

```
kubectl apply -f service.yml
```

12. Verify the Kubernetes deployment

```
kubectl get all
```

This is the output:

```
PS C:\SpringBoot WebAPI> kubectl get all
```

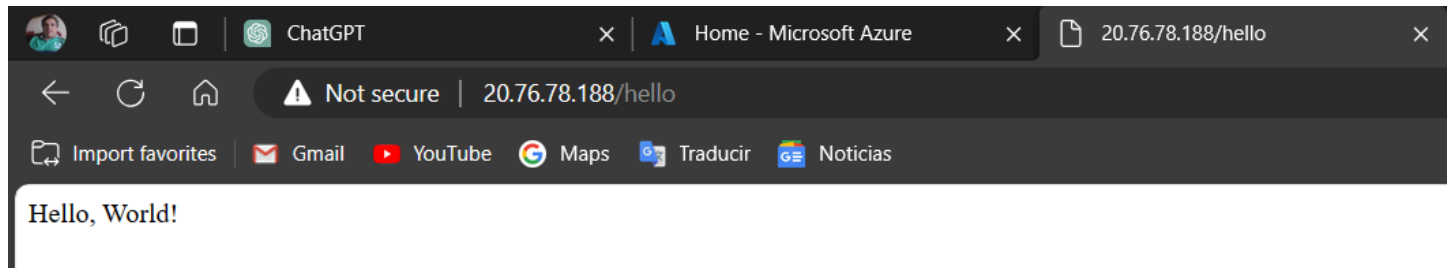
NAME	READY	STATUS	RESTARTS	AGE
pod/demoapi-deployment-65c6d7bd-rt7sx	1/1	Running	0	21s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/demoapi-service	LoadBalancer	10.0.112.84	20.76.78.188	80:31638/TCP	14s

service/kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	7m46s
--------------------	-----------	----------	--------	---------	-------

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/demoapi-deployment	1/1	1	1	21s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/demoapi-deployment-65c6d7bd	1	1	1	21s



Verify the Deployment. To ensure your deployment is running, use:

```
kubectl get deployments
```

For detailed information on the deployed pods, use:

```
kubectl get pods
```

To see the LoadBalancer IP address run this command:

```
kubectl get services
```