AZURE
KUBERNETES

SERVICE DEMO

INDEX

Introduction to Docker

Docker is an open-source platform that automates application deployment and management within lightweight, portable containers.

It allows applications and their dependencies to be packaged into a single unit, ensuring consistency across different environments.

Introduction to Kubernetes

Kubernetes is a powerful container orchestration platform that automates the deployment, scaling, and management of containerized applications.

It provides tools for load balancing, monitoring, and maintaining the availability of applications

So, what is Azure Kubernetes Service (AKS) ?

Azure Kubernetes Service, or AKS, is Microsoft Azure's managed Kubernetes service.

It simplifies the deployment and management of containerized applications, offering features like automatic scaling, rolling updates, and easy integration with Azure services.

Objective of the Demo

Our **goal** is to demonstrate how to deploy a simple web application to AKS.

Key Learning Points:

Containerization(docker, k8s)
Azure Container Registry (ACR)
AKS
LoadBalancers

Content of the course https://github.com/martimolanes/demoAzure

Build Our Web App

We'll create a straightforward web application using Python and Flask.
This app will respond with "Hello, Azure AKS!" when accessed.

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, Azure AKS!"

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=80)
```

```
<html>
▶ <head> ••• </head>
••• <body>Hello, Azure AKS!</body> == $0
</html>
```

now we can access the webapp at:
http://localhost:80

Containerize It

To ensure our app runs consistently, we'll containerize it using Docker.

A Dockerfile will define the containerization process, and we'll build the Docker image with specific commands.

```
# Use the official Python image from Docker Hub
FROM python:3.8-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

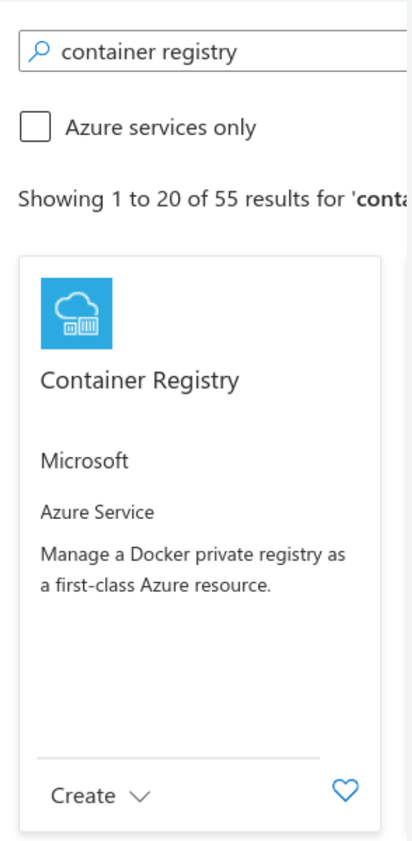
```
demoAzure main* 1m13s
venvpython2 ❯ sudo docker build -t my-aks-app .
```

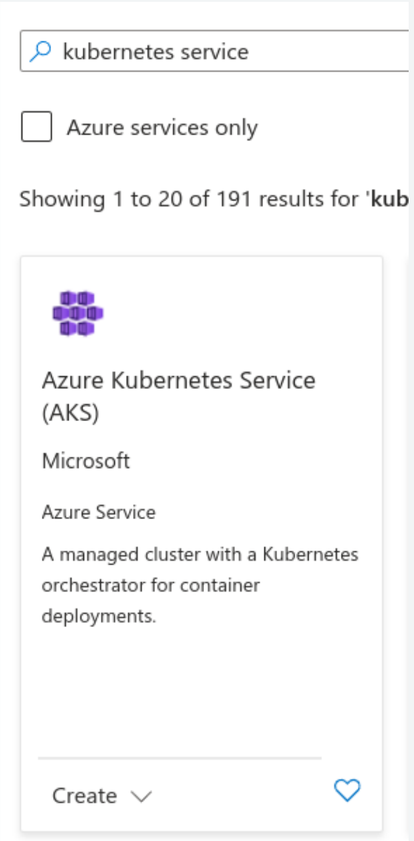
```
demoAzure main* 12s
venvpython2 ❯ sudo docker run -p 4000:80 my-aks-app
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.17.0.2:80
```

now we can access the webapp at:
http://localhost:4000

Create an ACR and AKS

Azure Container Registry (ACR) is like a secure storage for Docker images.
We'll create an ACR to store our Docker image and demonstrate how to push our app's image to ACR.





1 node pool

Role	Description	Scope
AcrPull	acr pull	This resource

```
demoAzure main*<200b>
venvpython2 ❯ az login
```

```
demoAzure main* 5m56s
venvpython2 ❯ az acr repository list --name testmartinho --output table

Result
--------
aks-app
```

```
sudo docker login testmartinho.azurecr.io
sudo docker tag my-aks-app:latest testmartinho.azurecr.io/aks-app:latest
sudo docker push testmartinho.azurecr.io/aks-app:latest
```

Create an ACR and AKS (2)

```bash
kubectl create secret docker-registry acr-secret \
  --docker-server=testmartinho.azurecr.io \
  --docker-username=testmartinho \
  --docker-password= \
  --docker-email=fmanu002@edu.xamk.fi
```

Deploy an Azure Kubernetes Service (AKS)

Azure Kubernetes Service (AKS) simplifies the management of Kubernetes clusters.

File: **deployment.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-aks-app-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-aks-app
  template:
    metadata:
      labels:
        app: my-aks-app
    spec:
      containers:
      - name: my-aks-app
        image: testmartinho.azurecr.io/test-aks:latest
        ports:
        - containerPort: 80
          name: http
      imagePullSecrets:
        - name: acr-secret
```

```bash
kubectl apply -f deployment.yaml
```

```
demoAzure main* 2s
 ❯ kubectl get pods
NAME                                      READY   STATUS              RESTARTS   AGE
my-aks-app-deployment-567d567465-8blp7    1/1     Running             0          37s
my-aks-app-deployment-567d567465-lh8bt    1/1     Running             0          37s
```

Deploy Our LoadBalancer

An app needs to be accessible to users, and that's where LoadBalancers come in.

I'll show you how to deploy a LoadBalancer within AKS and provide Kubernetes YAML files.

File: **service.yaml**

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-aks-app-service
spec:
  selector:
    app: my-aks-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

```bash
kubectl apply -f service.yaml
```

```
demoAzure main*
 ❯ kubectl get svc my-aks-app-service
NAME                  TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)        AGE
my-aks-app-service    LoadBalancer   10.0.70.147   20.166.171.240   80:30400/TCP   21h
```

THANK YOU FOR LISTENING!

Now I've introduced Docker, Kubernetes and how to work in this projects with Azure.

ANY QUESTIONS?

Content of the course https://github.com/martimolanes/demoAzure