

SIT722:TASK 5.2C: DEPLOYING A MICRO-SERVICE TO A AZURE MANAGED KUBERNETES CLUSTER

Step 1: The outcome of resource group implies that Kubernetes Service and Container Registry have been created.

The screenshot shows the Azure portal's main interface. In the top navigation bar, there are icons for 'Create a resource', 'Kubernetes services', 'All resources', 'Container registries', 'Resource groups', 'Storage accounts', 'Quickstart Center', 'Virtual machines', 'App Services', and 'More services'. Below this, the 'Resources' section lists three items under 'Recent': 'week7220t' (Container registry), '722wees5' (Kubernetes service), and 'deakunit' (Resource group). Each item has a 'Type' and 'Last Viewed' column. At the bottom of the 'Resources' section, there are links for 'See all', 'Subscriptions', 'Resource groups', 'All resources', and 'Dashboard'.

Step 2: The following set of commands are executed-

- az --version <Taken from task 5.1P, executed but missed this screenshot>

```
PS D:\Mydrive\SIT722\Task5\1> az --version
az 2.6.0 * 
core          2.6.0 * 
telemetry     1.1.0 
Dependencies: 
msal           1.38.0 
azure-identity 23.1.1 
Python location: 'C:\Program Files\Microsoft SDKs\Azure\CLI2\python.exe' 
Extensions directory: 'C:\Users\hp\az\extensions' 
Python (Windows) 3.11.0 (tags/v3.11.0:db85d1, Feb  6 2024, 22:01:32) [MSC v.1937 64 bit (AMD64)] 
Legal docs and information: aka.ms/azcliinfo
```

- az aks get-credentials --resource-group <Resource_Name> --name <Kubernetes_cluster_name>

```
PS D:\Mydrive\SIT722\Task5\5> az aks get-credentials --resource-group deakunit --name 722wees5 --overwrite-existing
Kerberos '722wees5' service context in C:\Users\hp\Azure\config
PS D:\Mydrive\SIT722\Task5\5>
```

- kubectl config current-context

```
PS D:\Mydrive\SIT722\Task5\5> kubectl config current-context
722wees5
PS D:\Mydrive\SIT722\Task5\5>
```

Step 3: Get output for commands below-

- docker build

2. docker tag

3. docker login

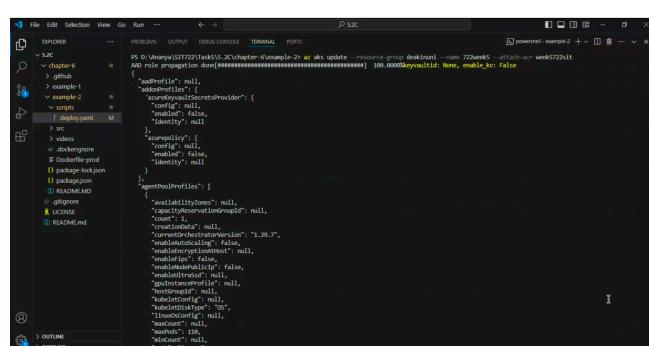


```
PS D:\Aurec\STT221\Task05> docker login week0722it.aurec.io --username week0722it --password jXwW2y8Rf1tbaUd4c7546hW1hBqjACBHEkH  
WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
Login Succeeded  
PS D:\Aurec\STT221\Task05>
```

4. docker push

Step 4: Deals with Kubernetes cluster via the commands specified-

- az aks update



2. kubectl apply

The screenshot shows the VS Code interface with the file 'scripts/deployment.yaml' open in the editor. The terminal at the bottom shows the command 'kubectl apply -f scripts/deployment.yaml' being run, and the response 'deployment.apps/video-streaming created'. The code itself defines a deployment named 'video-streaming' with one replica, selecting pods with the label 'app: video-streaming'.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: video-streaming
spec:
  replicas: 1
  selector:
    matchLabels:
      app: video-streaming
  template:
    metadata:
      labels:
        app: video-streaming
    spec:
      containers:
        - name: video-streaming
          image: week57251st.azurecr.io/video-streaming:1.0
```

3. kubectl get pods

The screenshot shows the VS Code interface with the file 'scripts/deployment.yaml' open in the editor. The terminal at the bottom shows the command 'kubectl get pods' being run, and the response 'NAME READY STATUS RESTARTS AGE'. It lists a single pod named 'video-streaming-69d974868c' in a 'Running' state with an age of 2m. The code remains the same as in the previous step.

```
NAME READY STATUS RESTARTS AGE
video-streaming-69d974868c 1/1 Running 0 2m
```

4. kubectl get deployments

The screenshot shows the VS Code interface with the file 'scripts/deployment.yaml' open in the editor. The terminal at the bottom shows the command 'kubectl get deployments' being run, and the response 'NAME READY UP-TO-DATE AVAILABLE AGE'. It lists a deployment named 'video-streaming' in a 'Ready' state with an age of 2m. The code remains the same as in the previous steps.

```
NAME READY UP-TO-DATE AVAILABLE AGE
video-streaming 1/1 1/1 1/1 2m
```

5. kubectl get services

The screenshot shows the VS Code interface with the file 'scripts/deployment.yaml' open in the editor. The terminal at the bottom shows the command 'kubectl get services' being run, and the response 'NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE'. It lists a service named 'video-streaming' with an external IP of 10.0.237.241 and port 80. The code remains the same as in the previous steps.

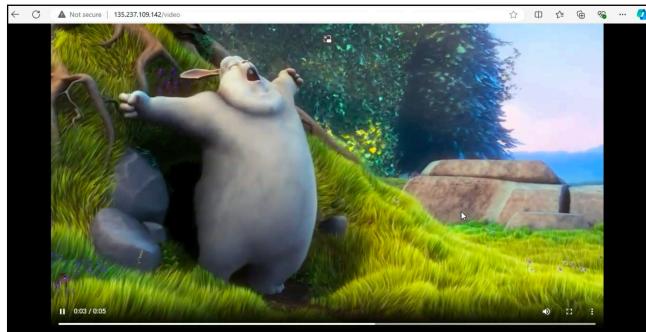
```
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
video-streaming LoadBalancer 10.0.237.241 10.0.237.109:142 80:31575/TCP 8m
```

6. kubectl delete

The screenshot shows the VS Code interface with the file 'scripts/deployment.yaml' open in the editor. The terminal at the bottom shows the command 'kubectl delete -f scripts/deployment.yaml' being run, and the response 'deployment.apps "video-streaming" deleted'. The code remains the same as in the previous steps.

```
PS D:\Users\57251st\Downloads\K8s\5.2\chapter-0\example-2> kubectl delete -f scripts/deployment.yaml
deployment.apps "video-streaming" deleted
```

Step 5: View the streaming video on browser.



Step 6: Resource group is seen to verify that both Kubernetes Cluster and Container Registry resources are deleted.

A screenshot of the Microsoft Azure portal's 'All resources' search results page. The search bar at the top contains several filters: 'Subscription equals all', 'Resource group equals all', 'Type equals all', 'Location equals all', and 'Subscription equals all'. Below the search bar, there is a table with columns for Name, Type, Resource group, Location, and Subscription. The table is currently empty, displaying the message 'No resources match your filters'. There are also buttons for 'Create resource' and 'Clear filters'.