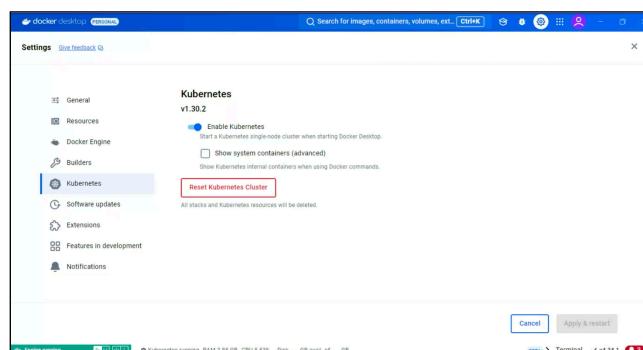


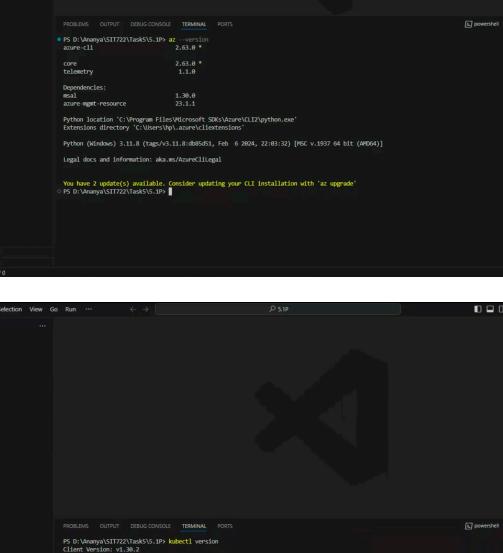
SIT722: TASK 5.1: DEPLOYING A MICRO-SERVICE TO KUBERNETES CLUSTER

Step 1: Git clone the repository for chapter-6.

Step 2: Kubernetes engine is enabled on docker desktop.



Step 3: Terminal results for commands az --version and kubectl version.



```
PS D:\Myanya\G17722\Task5\$ az --version
azure-cli
 2.6.0 * 

core
 2.6.0 *
telemetry
 1.1.8

Dependencies:
  msal
    1.36.0
  azure-identity
    2.3.1

Python location: 'C:\Program Files\Microsoft SDKs\Azure\CLI2\python.exe'
Extensions directory: 'C:\Users\Myanya\Azure\cliextensions'

Python (Windows) 3.11.8 (tags/v3.11.8:ddbfef8, Feb 6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)]
Legal docs and information: aka.ms/AzureCliLegal

You have 2 update(s) available, consider updating your CLI installation with 'az upgrade'
(PS D:\Myanya\G17722\Task5\$) ▶

PS D:\Myanya\G17722\Task5\$ hubctl version
Client Version: v1.36.2
Kustomize Version: v5.8.4-0.20230901165947-dce8bf90c3
Spiffe Version: v0.10.0
(PS D:\Myanya\G17722\Task5\$) ▶
```

Step 4: The outputs for following commands is exhibited-
1 docker build

2. `kubectl config current-context`,

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a project structure with 'chapter-6', 'github', 'example-1', 'scripts', 'src', 'tests', 'dockerspinner', and 'Docksterle-prod'. The 'scripts' folder contains 'package-lock.json', 'README.md', 'example-2', 'gitignore', 'LICENCE', and 'README.ed'. The terminal window shows the command 'kubectl config current-context' being run, with the output 'cluster' displayed.

3. `kubectl apply`,

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows the same project structure as the previous screenshot. The terminal window shows the command 'kubectl apply -f scripts/deploy.yaml' being run, with the output 'deployment.apps/video-streaming created' displayed.

4. `kubectl get pods`,

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows the project structure. The terminal window shows the command 'kubectl get pods' being run, with the output 'NAME READY STATUS RESTARTS AGE' and 'video-streaming 1/1 Running 0 82s' displayed.

5. `kubectl get deployments`,

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows the project structure. The terminal window shows the command 'kubectl get deployments' being run, with the output 'NAME READY UP-TO-DATE AVAILABLE AGE' and 'video-streaming 1/1 1/1 1/1 82s' displayed.

6. `kubectl get services and`

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows the project structure. The terminal window shows the command 'kubectl get services' being run, with the output 'NAME CLUSTER-IP EXTERNAL-IP PORT(S) AGE' and 'kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 18d' displayed.

7. kubectl delete



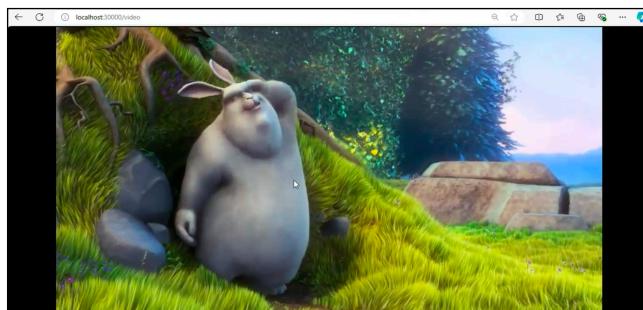
The screenshot shows a terminal window with several tabs open. The active tab is titled "deployment" and contains the following command and its output:

```
PS C:\Users\ASUS\Downloads\k8s\l1\chapter-6\example-1> kubectl delete -f scripts/deploy.yaml
```

The output shows the creation of a deployment named "video-streaming" with one replica, selecting the "Video" namespace, and using the "nginx-ingress" template.

Below the terminal, there are several other tabs and windows visible, including "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The status bar at the bottom indicates "In 5 min" and shows icons for file operations like "New", "Open", "Save", and "Close".

Step 5: Browser result of streamed video, represents successful running of application.



Short Answers-

1. The ‘deploy.yaml’ file for example-1 of chapter-6 clearly indicates that it has 2 sections- deployment and service section. The table, will elaborate on meaning of each line of this file-

Deployment Section-

- apiVersion: apps/v1==> This defines the version of the Kubernetes API being used. Apps/v1 is commonly used for managing deployments
 - kind: Deployment ==> Specifies that this is a Kubernetes deployment. A deployment is responsible for managing and ensuring that the desired number of application replicas are running.
 - metadata: name: video-streaming ==> metadata defines metadata for the deployment such as the name. In this case the deployment is named video-streaming.
 - spec: replicas:1 ==> spec defines the desired state of the deployment. Here, replicas:1 means Kubernetes will maintain one replica or instance of the video-streaming application.
 - selector: matchLabels: app:video-streaming ==> The selector specifies how Kubernetes identifies the pods managed by this deployment. It matches with any pods with the label app:video-streaming, ensuring only those pods are controlled by this deployment.
 - template: metadata: labels: app:video-streaming ==> templates defines the template used to create Pods for the deployment. The labels within the template assign the app:video-streaming label to these pod, allowing them to be selected by the deployment
 - spec: containers: name: video-streaming ==> This spec block defines the specifications for the containers inside each pod. Here, the single container is video-streaming.
 - image: video-streaming:1 ==> The image specifies the container image that Kubernetes will pull to run the application. In this case, it is an image named video-streaming, with version 1.
 - imagePullPolicy: Never ==> It tells Kubernetes to use the local copy of the container image and not to try pulling the image from a registry.
 - env: name:PORT value: 4000 ==> env defines the environment variables passed to the container. Here, a variable need 'PORT' is set to 4000. This is likely used to configure the application to run on the port 4000 within the container.

Service Section-

- apiVersion: v1==> This defines the version of the Kubernetes API for the service resource. V1 is used for the most basic Kubernetes objects like services.
 - kind: Service ==> Specifies that this is a Kubernetes Service resource. A service provides a stable network endpoint to access one or more pods.
 - metadata: name: video-streaming ==> It defines metadata for the service, such as the name. Here, service is named video-streaming.
 - spec: selector: app:video-streaming==> spec defines the service's configuration. The selector specifies that the service should route traffic to pods with the label app:video-streaming (Matching the deployment's pods).
 - type: NodePort ==> It exposes the service on a static port on each Node in the Kubernetes cluster. This allows external access to the service via the node's IP address and the node port.
 - ports: protocol:TCP port:80 targetPort:4000 nodePort:30000 ==> protocol:TCP specifies that the service uses the TCP protocol. Port:80 is the port that the service will expose externally (on the node). targetPort:4000 is the internal port on the container where traffic will be directed (matching the port environment variable). nodePort:30000 is the static port on the node where external traffic can access the service. In this case the service will be accessible at <http://<node-ip>:30000>.

2. The latest Kubernetes version used in example 1 is v1.29.2.