DEVOPS FOR AIOT
SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING, SINGAPORE POLYTECHNIC

**LABORATORY 7: INTRODUCTION TO KUBERNETES**

## Objectives

By the end of the laboratory, students will be able to
- Deploy their containerized Python application as a Pod in Kubernetes
- Delete the Python Application Pod
- Deploy their containerized Python application as a Deployment in Kubernetes
- Increase the number of replicas in the Deployment
- Create a Service

## Activities
- Introduction to kubectl command line interface
- Deploy a Pod
- Delete a Pod
- Deploy a Deployment
- Observe the desired replica state of a Deployment
- Increase the number of replicas in a Deployment
- Expose the Deployment as a Service

## Review
- A Python Flask Web Application is deployed as a Pod in Kubernetes
- The application is deployed as a Deployment in Kubernetes
- The number of replicas of the Deployment is scaled up

## Equipment:
- Windows OS laptop with Ubuntu Linux VM
- Git / Github
- Docker Hub

# 1    Setup of Docker Hub

Docker Hub account should have been created in Lab 6 with containerised Python Flask Web Application. You can refer to the previous Lab 6 – Introduction to Docker guide for the account creation if you have not done so.

# 2    Containerisation of Python Flask Web Application

The Python Flask Web Application should be containerised and pushed to Docker Hub. You can refer to your previous Lab 6 – Introduction to Docker guide for the steps if you have not done so.

# 3    Accessing Kubernetes

We have Docker daemon running in the Ubuntu Linux VM. It comes along with a Kubernetes environment for us to run our application.

## 3.1.    Kubernetes "kubectl" command line interpreter

Similar to Git and Docker, we will run Kubernetes at the command line using the "kubectl" executable. Kubectl is already installed in the VM.

For more detailed information for all of the commands used in this lab exercise, please refer to the online official Kubernetes "kubectl" manual at the URL below.

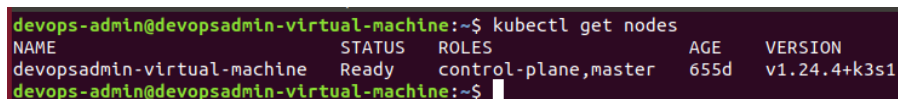https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

## 3.2.    Verify our Kubernetes cluster

We will verify that Kubernetes is running in our local environment :

▪  Open a terminal in your Ubuntu Linux VM by right clicking on the desktop.

▪  Run the following kubectl command below:

```
kubectl get nodes
```

▪  You should see the output like the following showing that docker-desktop is running as a node with the role of control-plane.



```
devops-admin@devopsadmin-virtual-machine:~$ kubectl get nodes
NAME                         STATUS   ROLES                 AGE    VERSION
devopsadmin-virtual-machine  Ready    control-plane,master  655d   v1.24.4+k3s1
devops-admin@devopsadmin-virtual-machine:~$
```
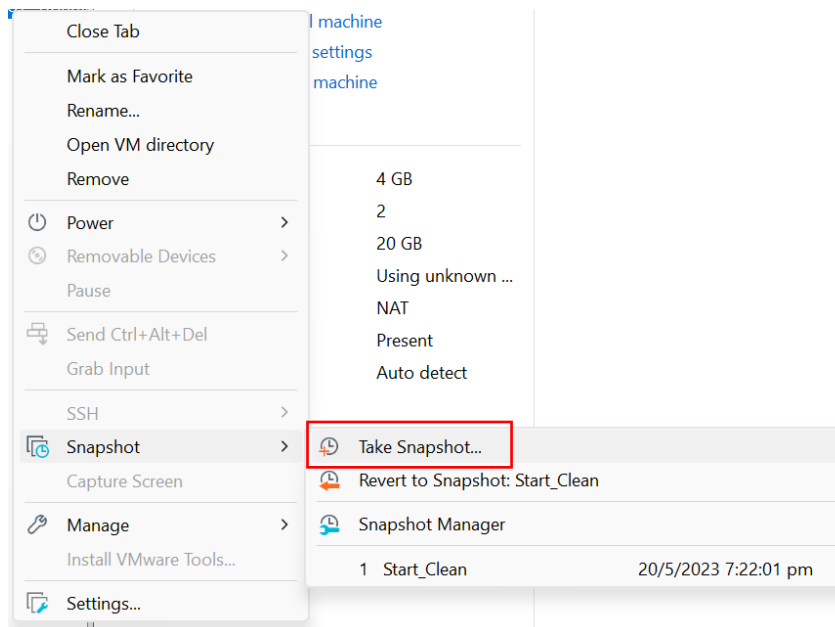
## 4 Snapshot of Ubuntu Linux

It is a good practice to take a snapshot of the current status of your Ubuntu Linux Virtual Machine, in case if you want to practise Lab 7 again.

In the VMware Workstation Pro, right click on the VM that you have created in Lab 6.
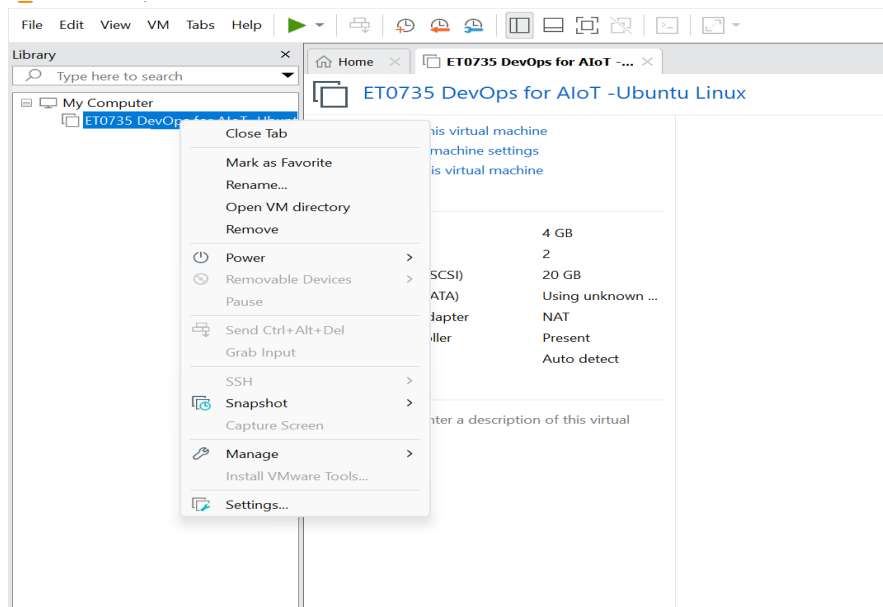


Select "Take Snaphot.." option, and choose a relevant name for the snapshot.



==Anytime you want to get back your VM to the current status, choose the "Revert to Snapshot" option and select the appropriate snapshot.==

DevOps for AIoT

---

**5** **Deploy the Python Flask Web Application from Docker Hub as a Pod in Kubernetes**

We will deploy the Python Flask Web Application which you have pushed earlier to Docker Hub in your local Kubernetes environment.

- In the Terminal, enter the command below which will deploy the Docker Image "flask-app" from Docker Hub to your local Kubernetes:

```
kubectl run my-flask-app --image={your namespace}/flask-
app
```

- You should see the output like the following showing that the Python Flask Web Application named "my-flask-app" has been successfully deployed as a Pod in Kubernetes.

```
(base) esweehweish-a01:~ eswee$ kubectl run my-flask-app --image=eugenesweehs/flask-app
pod/my-flask-app created
```

- To check that the Pod "my-flask-app" has been successfully running, run the command below which lists all the Pods running in Kubernetes.

```
kubectl get pods
```

```
(base) esweehweish-a01:~ eswee$ kubectl get pods
NAME            READY    STATUS     RESTARTS    AGE
my-flask-app    1/1      Running    0           7m49s
```

---

To view the Pod "my-flask-app" in more detail, run the command below:

```
kubectl describe pod my-flask-app
```

```
(base) esweehweish-a01:~ eswee$ kubectl describe pod my-flask-app
Name:          my-flask-app
Namespace:     default
Priority:      0
Node:          docker-desktop/192.168.65.4
Start Time:    Mon, 13 Jun 2022 10:35:06 +0800
Labels:        run=my-flask-app
Annotations:   <none>
Status:        Running
IP:            10.1.0.10
IPs:
  IP:  10.1.0.10
Containers:
  my-flask-app:
    Container ID:   docker://5e80e8791f484ee2fd39a2dd085612a4b1bd28ce
    Image:          eugenesweehs/flask-app
    Image ID:       docker-pullable://eugenesweehs/flask-app@sha256:b
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 13 Jun 2022 10:35:13 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-acc
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
```

## 6   Delete the Python Flask Web Application pod

A single pod runs the risk of a single point of failure in Kubernetes. If the pod fails, users will not be able to access the application.

- We will simulate the failure of the "my-flask-app" pod by deleting the pod. Run the command below:

```
kubectl delete pod my-flask-app
```

```
(base) esweehweish-a01:~ eswee$ kubectl delete pod my-flask-app
pod "my-flask-app" deleted
```

- To check that the Pod "my-flask-app" has been successfully deleted, run the command below:

```
kubectl get pods
```

```
(base) esweehweish-a01:~ eswee$ kubectl get pods
No resources found in default namespace.
```

## 7 Deploy the Python Flask Web Application as a Deployment

A Deployment manages the Pod and ensures the Pod is always running. We will deploy the Python Flask Web Application as a Deployment this time round in Kubernetes.

- In the Terminal, enter the command below which will deploy the Docker Image "flask-app" from Docker Hub to your local Kubernetes:

```
kubectl create deployment my-flask-app --image={your namespace}/flask-app
```

- You should see the output like the following showing that the Python Flask Web Application named "my-flask-app" has been successfully deployed as a Deployment in Kubernetes.

```
(base) esweehweish-a01:~ eswee$ kubectl create deployment my-flask-app --image=eugenesweehs/flask-app
deployment.apps/my-flask-app created
```

- To check that the Deployment "my-flask-app" has been successfully running, run the command below which lists all the Deployments running in Kubernetes:

```
kubectl get deployments
```

```
(base) esweehweish-a01:~ eswee$ kubectl get deployments
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
my-flask-app    1/1     1            1           6m54s
```

- Run the command below which lists all the Pods running in Kubernetes:

```
kubectl get pods
```

```
(base) esweehweish-a01:~ eswee$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
my-flask-app-96f9987c5-7bh5q  1/1     Running   0          7m41s
```

- Notice the name of the Pod is appended with some random hash code behind the name of the deployment "my-flask-app". This indicates that the Pod is being managed by a Deployment.

## 8    Delete the Python Flask Web Application pod

We will simulate the failure of the pod again by deleting the pod.

- Run the command below (note that the name of your pod differs from what you see in the screenshot as the appended hash code is random):

```
kubectl delete pod {name of my-flask-app pod}
```

```
(base) esweehweish-a01:~ eswee$ kubectl delete pod my-flask-app-96f9987c5-7bh5q
pod "my-flask-app-96f9987c5-7bh5q" deleted
```

kubectl delete pod my-flask-app-7d79ffb5d7-zlrc9

- Run the command below to see the status of the deployment:

```
kubectl get deployments
```

```
(base) esweehweish-a01:~ eswee$ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
my-flask-app    0/1     1            0           36m
```

- You may see the READY status as 0/1 as shown in the screenshot above or READY status as 1/1 if the deployment has already created a new pod.

- Run the command below which lists all the Pods running in Kubernetes:

```
kubectl get pods
```

```
(base) esweehweish-a01:~ eswee$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
my-flask-app-96f9987c5-4z2m7  1/1     Running   0          24m
```

- Notice that the hash code appended as part of the name of the pod has changed. This is a new pod that is created by the deployment when it detects that the previous pod was deleted.

## 9    Describe the Python Flask Web Application Deployment

We will look in more detail on the deployment configuration.

- Run the command below to view the deployment:

```
kubectl describe deployment my-flask-app
```

```
(base) esweehweish-a01:~ eswee$ kubectl describe deployment my-flask-app
Name:                   my-flask-app
Namespace:              default
CreationTimestamp:      Mon, 13 Jun 2022 11:06:43 +0800
Labels:                 app=my-flask-app
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=my-flask-app
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=my-flask-app
  Containers:
   flask-app:
    Image:          eugenesweehs/flask-app
```

- Notice the "Replicas" field has been set to "1 desired". This setting tells Kubernetes to ensure there is always 1 Pod of my-flask-app to be running at all times

## 10  Scale the Python Flask Web Application with more Replicas

A Replica is a copy of the Pod that is running our flask-app. Our current Deployment has 1 Replica running by default. We can increase the number of Replicas to ensure our application can scale to handle the increasing load from a spike in the number of users accessing.

- We will scale our flask-app by increasing to 5 Replicas. Run the command below to increase our replicas to 5:
  devops-admin@devopsadmin-virtual-machine:~$ kubectl scale deployment my-flask-app --replicas=5

```
kubectl scale deployment my-flask-app --replicas=5
```

```
(base) esweehweish-a01:~ eswee$ kubectl scale deployment my-flask-app --replicas=5
deployment.apps/my-flask-app scaled
```

- To check the updated deployment, run the command below:

```
kubectl get deployments
```

```
(base) esweehweish-a01:~ eswee$ kubectl get deployments
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
my-flask-app    5/5     5            5           66m
```

- Notice that the READY status is now 5/5.
- Run the command below which lists all the Pods running in Kubernetes:

```
kubectl get pods
```

```
(base) esweehweish-a01:~ eswee$ kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
my-flask-app-96f9987c5-4tfhn    1/1     Running   0          4m3s
my-flask-app-96f9987c5-cz7qq    1/1     Running   0          4m3s
my-flask-app-96f9987c5-g7ft4    1/1     Running   0          4m3s
my-flask-app-96f9987c5-gcs9m    1/1     Running   0          31m
my-flask-app-96f9987c5-lzs54    1/1     Running   0          4m3s
```

- We have 5 pods running our flask-app that are managed by the deployment.
- Run the command below to view the deployment:

```
kubectl describe deployment my-flask-app
```

- Notice the "Replicas" field has been set to "5 desired". This setting tells Kubernetes to ensure there are always 5 Pods of my-flask-app to be running.

## 11  Expose the Python Flask Web Application Deployment as a Service

Pods are unreliable and are replaced with new IP addresses when new Pods are created by replica sets. A Service provides a stable network abstraction point with a reliable IP address which load-balances across the Pods.

- We will expose our flask-app deployment as a Service with the type NodePort by running the command below:

```
kubectl expose deployment my-flask-app --type=NodePort --port=5000
```

kubectl expose deployment my-flask-app --type=NodePort --port=5000



kubectl get svc

Open your web browser and go to the assigned ip address for your service. In this case it is http://10.109.71.70:5000   use firefox in VM



To delete the service:

```
kubectl delete services my-flask-app
```