Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2023-24

Class:	TE	**Semester:**	VI
Course Code:	CSL605	**Course Name:**	Skill Based Lab course : Cloud Computing

Name of Student:	Soham Ajit Dahanukar
Roll No. :	13
Experiment No.:	10
Title of the Experiment:	Implement container orchestration using Kubernetes
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty :

Signature :

Date

Experiment No. 10

Aim: To study and implement container orchestration using Kubernetes

Objective: To understand container orchestration using Kubernetes.

Theory:

Container orchestration automates the deployment, management, scaling, and networking of containers. Container orchestration can be used in any environment where you use containers. It can help you to deploy the same application across different environments without needing to redesign it. And microservices in containers make it easier to orchestrate services, including storage, networking, and security. container orchestration to automate and manage tasks such as:

- Provisioning and deployment
- Configuration and scheduling
- Resource allocation
- Container availability
- Scaling or removing containers based on balancing workloads across your infrastructure
- Load balancing and traffic routing
- Monitoring container health
- Configuring applications based on the container in which they will run
- Keeping interactions between containers secure

Kubernetes is an open-source container management (orchestration) tool. Its container management responsibilities include container deployment, scaling & descaling of containers & container load balancing.

Features of Kubernetes

- Automatic Bin packing
- Service Discovery and Load Balancing
- Storage Orchestration
- Self-Healing
- Secrete and configuration management.
- Batch execution
- Horizontal Scaling
- Automatic Rollbacks and Rollouts

Steps:

----Enable Kubernetes---

1. After installing Docker Desktop, you should see a Docker icon in your system tray. Right-click on it, and navigate **Settings** > **Kubernetes**.
2. Check the checkbox labeled **Enable Kubernetes**, and click **Apply & Restart**. Docker Desktop will automatically set up Kubernetes for you. You'll know that Kubernetes has been successfully enabled when you see a green light beside 'Kubernetes *running*' in the **Settings** menu.
3. In order to confirm that Kubernetes is up and running, create a text file called pod.yaml with the following content:

```
apiVersion:
v1 kind: Pod
metadata:
  name:
demo spec:
  containers:
  - name: testpod
    image:
    alpine:latest
    command: ["ping", "8.8.8.8"]
```

This describes a pod with a single container, isolating a simple ping to 8.8.8.8.

4. In PowerShell, navigate to where you created pod.yaml and create your pod:

```
$ kubectl apply -f pod.yaml
```

5. Check that your pod is up and running:

```
$ kubectl get pods
```

You should see something like:

```
NAMEREADY STATUS
RESTARTS AGE
demo 1/1 Running 4
0 s
```

6. Check that you get the logs you'd expect for a ping process:

```
$ kubectl logs demo
```

7. Finally, tear down your test pod:

```
$ kubectl delete -f pod.yaml
```

---Deploy Kubernetes---

Prerequisite

- Download and install Docker Desktop as described in Get Docker.
- Work through containerizing an application in Part 2.
- Make sure that Kubernetes is enabled on your Docker Desktop:
  - **Windows**: Click the Docker icon in the system tray and navigate to **Settings** and make sure there's a green light beside 'Kubernetes'.

Describing apps using Kubernetes YAML

1. Place the following in a file called bb.yaml: apiVersion:

```
apps/v1
kind: Deployment
metadata:
  name:        bb-demo
namespace:     default
spec:
  replicas: 1
  selector:
    matchLabels:
      bb:     web
  template:
  metadata:
     labels:
       bb: web
   spec:
     containers:
     - name: bb-site
       image:    getting-started
       imagePullPolicy:
       Never
---
apiVersion:
v1          kind:
Service
metadata:
  name: bb-entrypoint
  namespace: default
```

```
spec:
  type: NodePort
```

```
  selector:
    bb: web
  ports:
  - port: 3000
    targetPort: 3000
    nodePort: 30001
```

--Deploy and check application—

1. In a terminal, navigate to where you created bb.yaml and deploy your application to Kubernetes:

```
$ kubectl apply -f bb.yaml
```

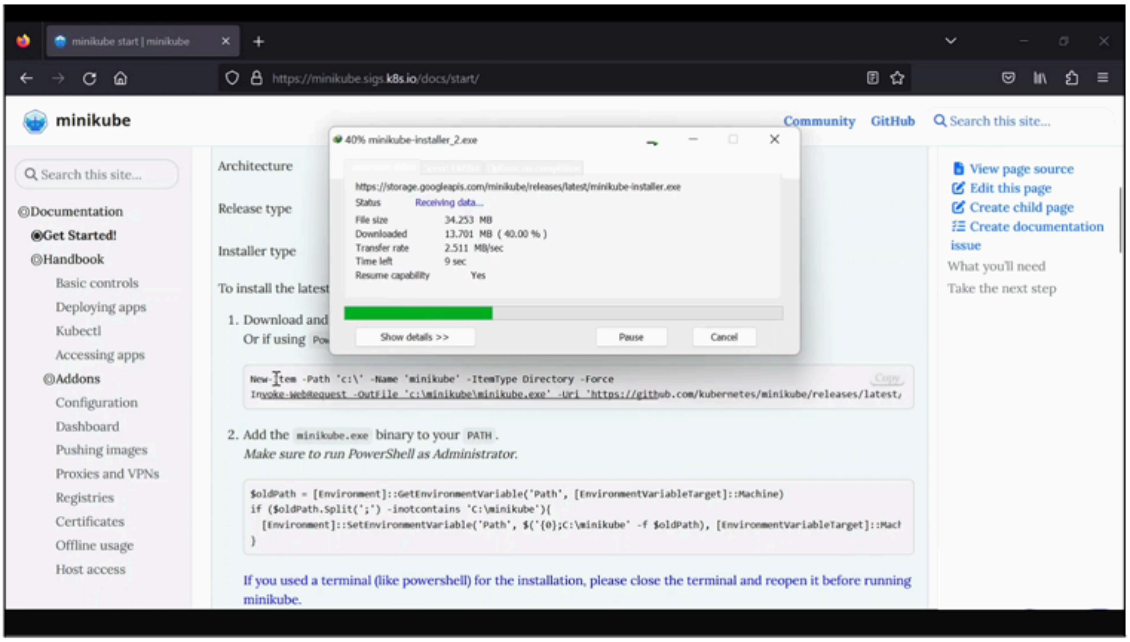
2. Make sure everything worked by listing your deployments:

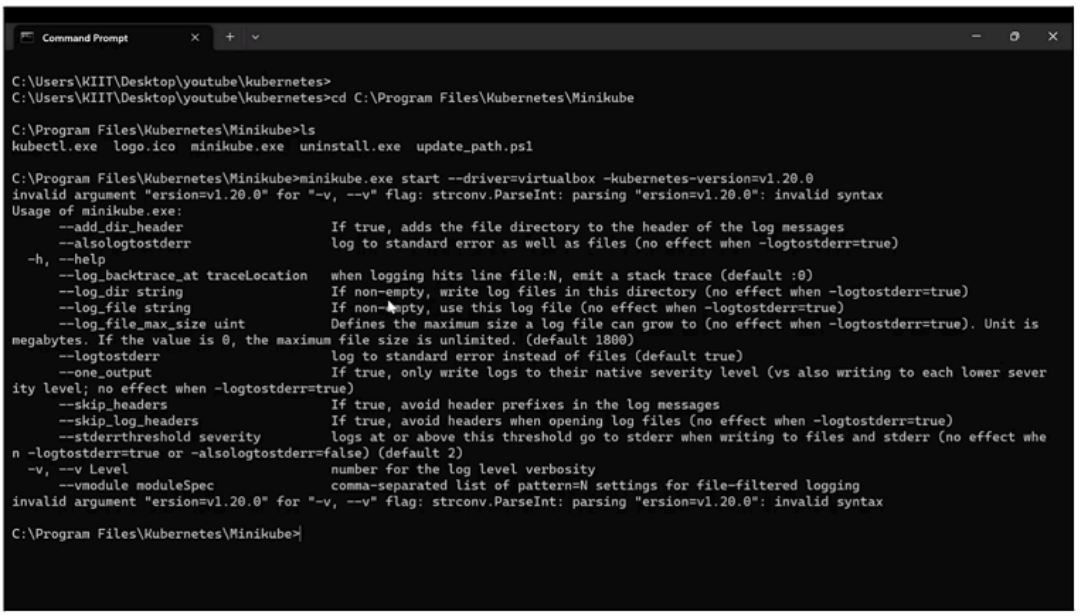
```
$ kubectl get deployments
```

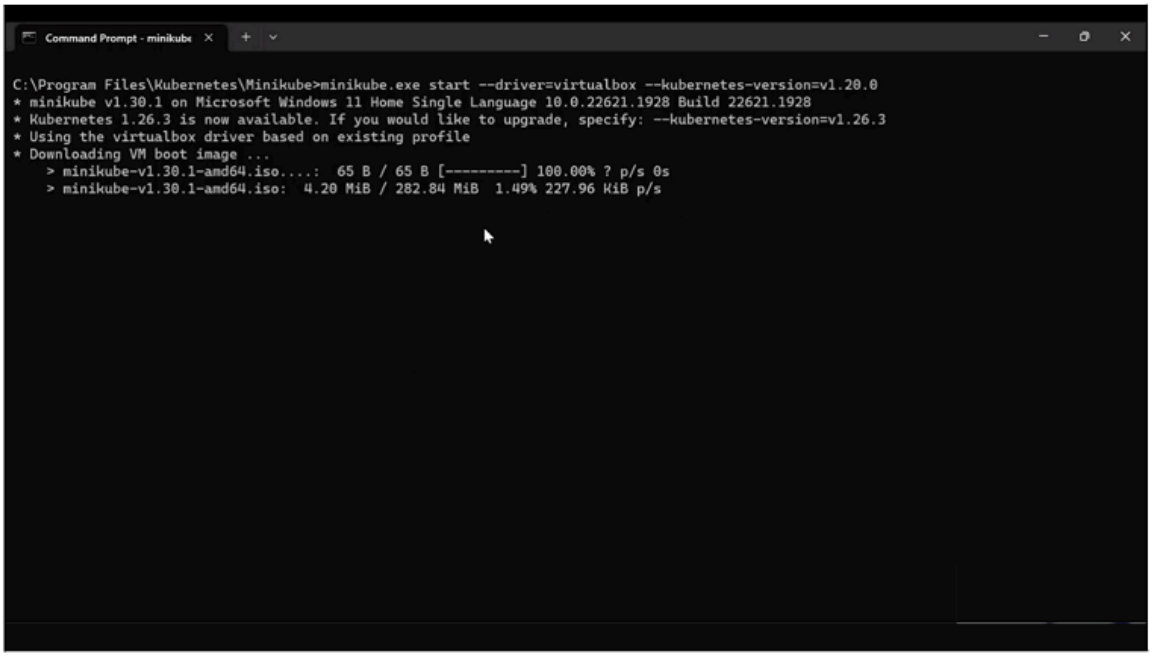
3. Open a browser and visit your Todo app at localhost:30001; you should see your Todo application.
4. Once satisfied, tear down your application:

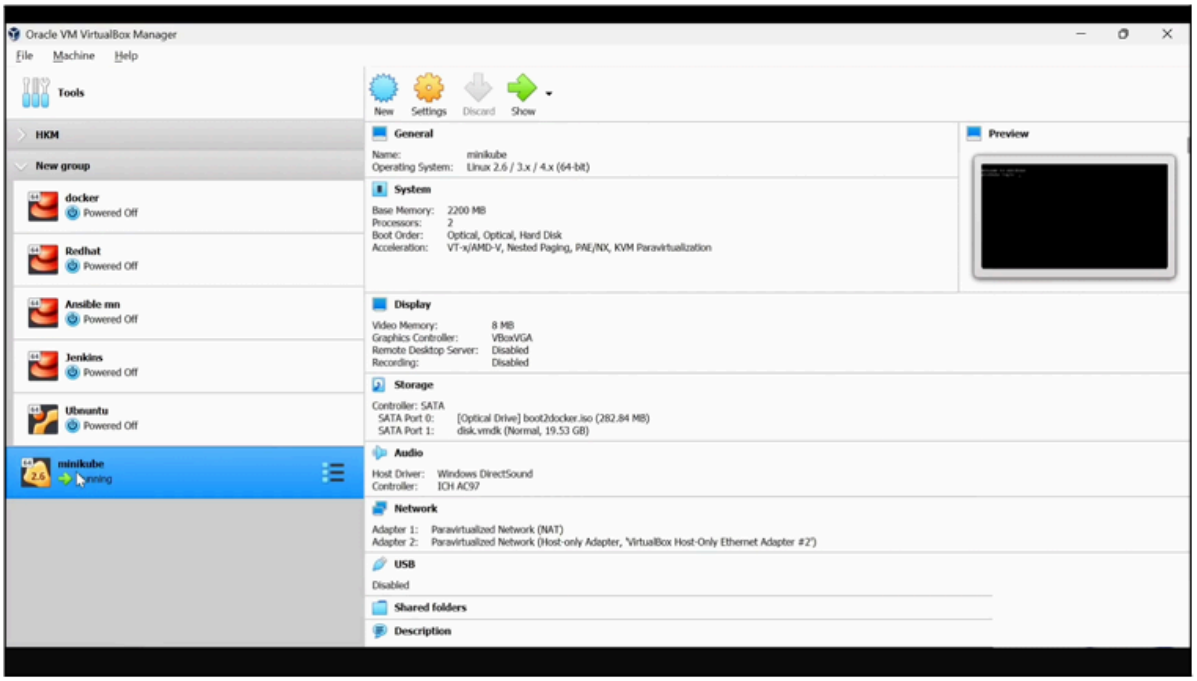
```
$ kubectl delete -f bb.yaml
```

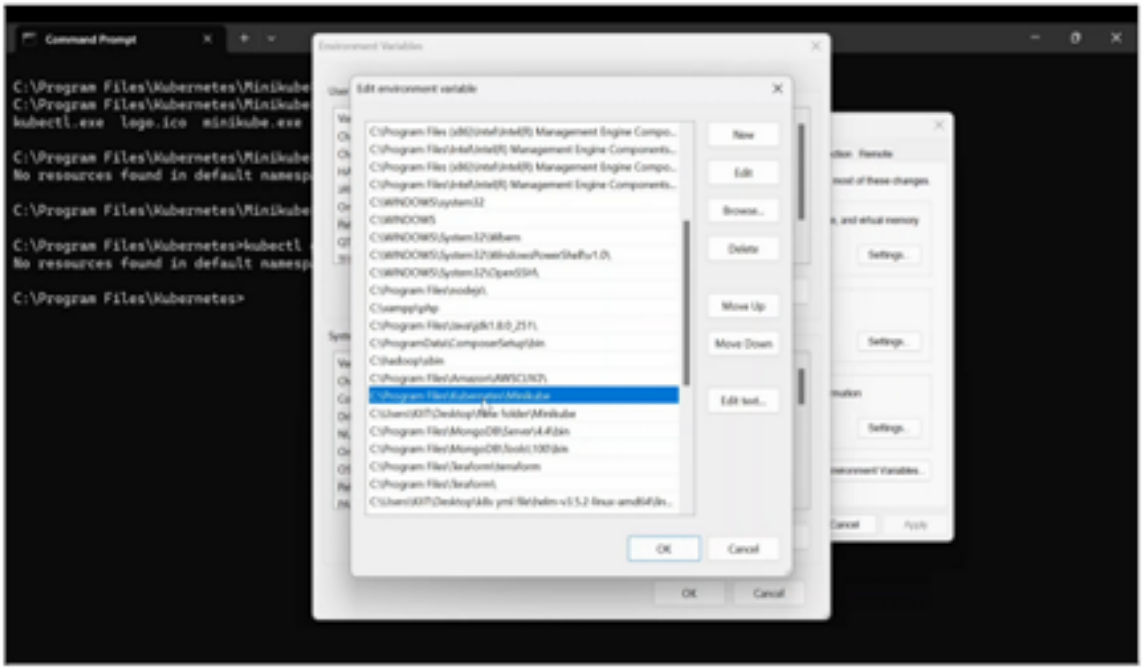
Output/Observation:

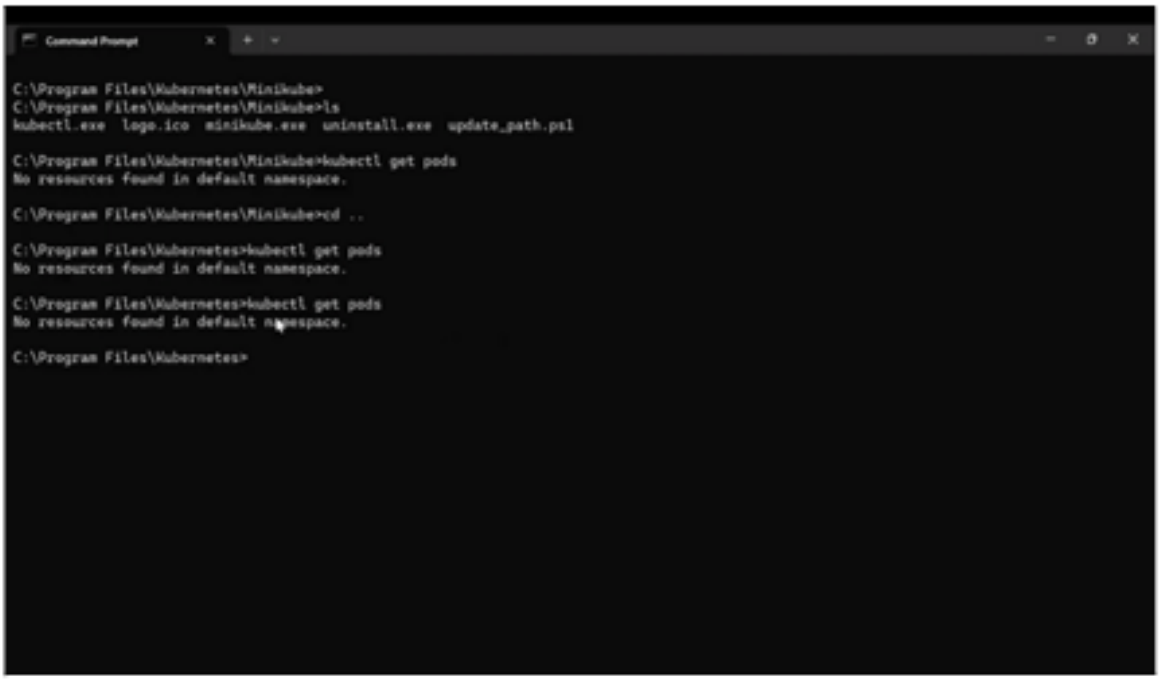





Command Prompt - minikube
C:\Program Files\Kubernetes\Minikube>minikube.exe start --driver=virtualbox --kubernetes-version=v1.20.0
* minikube v1.30.1 on Microsoft Windows 11 Home Single Language 10.0.22621.1928 Build 22621.1928
* Kubernetes 1.26.3 is now available. If you would like to upgrade, specify: --kubernetes-version=v1.26.3
* Using the virtualbox driver based on existing profile
* Downloading VM boot image ...
> minikube-v1.30.1-amd64.iso....: 65 B / 65 B [---------] 100.00% ? p/s 0s
> minikube-v1.30.1-amd64.iso: 4.20 MiB / 282.84 MiB 1.49% 227.96 KiB p/s


Oracle VM VirtualBox Manager
File Machine Help
Tools
New Settings Discard Show
HKM
New group
docker
Powered Off
Redhat
Powered Off
Ansible mn
Powered Off
Jenkins
Powered Off
Ubuntu
Powered Off
minikube
General
Name: minikube
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)
System
Base Memory: 2200 MB
Processors: 2
Boot Order: Optical, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization
Preview
Display
Video Memory: 8 MB
Graphics Controller: VBoxVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: SATA
SATA Port 0: [Optical Drive] boot2docker.iso (282.84 MB)
SATA Port 1: disk.vmdk (Normal, 19.53 GB)
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97
Network
Adapter 1: Paravirtualized Network (NAT)
Adapter 2: Paravirtualized Network (Host-only Adapter, 'VirtualBox Host-Only Ethernet Adapter #2')
USB
Disabled
Shared folders
Description


Command Prompt
C:\Program Files\Kubernetes\Minikube
C:\Program Files\Kubernetes\Minikube
kubectl.exe logo.ico minikube.exe
C:\Program Files\Kubernetes\Minikube
No resources found in default namesp
C:\Program Files\Kubernetes\Minikube
C:\Program Files\Kubernetes>kubectl
No resources found in default namesp
C:\Program Files\Kubernetes>
Edit environment variable
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel


Command Prompt
C:\Program Files\Kubernetes\Minikube>
C:\Program Files\Kubernetes\Minikube>ls
kubectl.exe logo.ico minikube.exe uninstall.exe update_path.ps1
C:\Program Files\Kubernetes\Minikube>kubectl get pods
No resources found in default namespace.
C:\Program Files\Kubernetes\Minikube>cd ..
C:\Program Files\Kubernetes>kubectl get pods
No resources found in default namespace.
C:\Program Files\Kubernetes>kubectl get pods
No resources found in default namespace.
C:\Program Files\Kubernetes>

Conclusion:

Comment on implementation of Containerization using Kubernetes

ANS: Implementing containerization using Kubernetes streamlines application deployment, scaling, and management across clusters. It offers automatic orchestration, scalability, and resilience, enhancing application availability and performance. However, Kubernetes adoption entails complexities in configuration, resource overhead, and operational management, necessitating expertise and careful planning for successful implementation. Despite challenges, Kubernetes provides powerful features for declarative configuration, service discovery, and load balancing, empowering organizations to build and run robust containerized applications at scale.