Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

Course: SBL: Cloud Computing
Course code: CSL605
Year: TE SEM: VI

Experiment No. 10

AIM:- To study and implement container orchestration using Kubernetes

Name:

Roll Number:

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	5	3	2
Understanding	5	3	2
Journal work and timely submission.	10	8	4

Checked by

Name of Faculty : Ichhanshu Jaiswal

Signature :

Date :

Experiment No. 10

Aim: To study and implement container orchestration using Kubernetes

Theory:

Container orchestration automates the deployment, management, scaling, and networking of containers across the cluster

It is focused on managing the life cycle of containers.

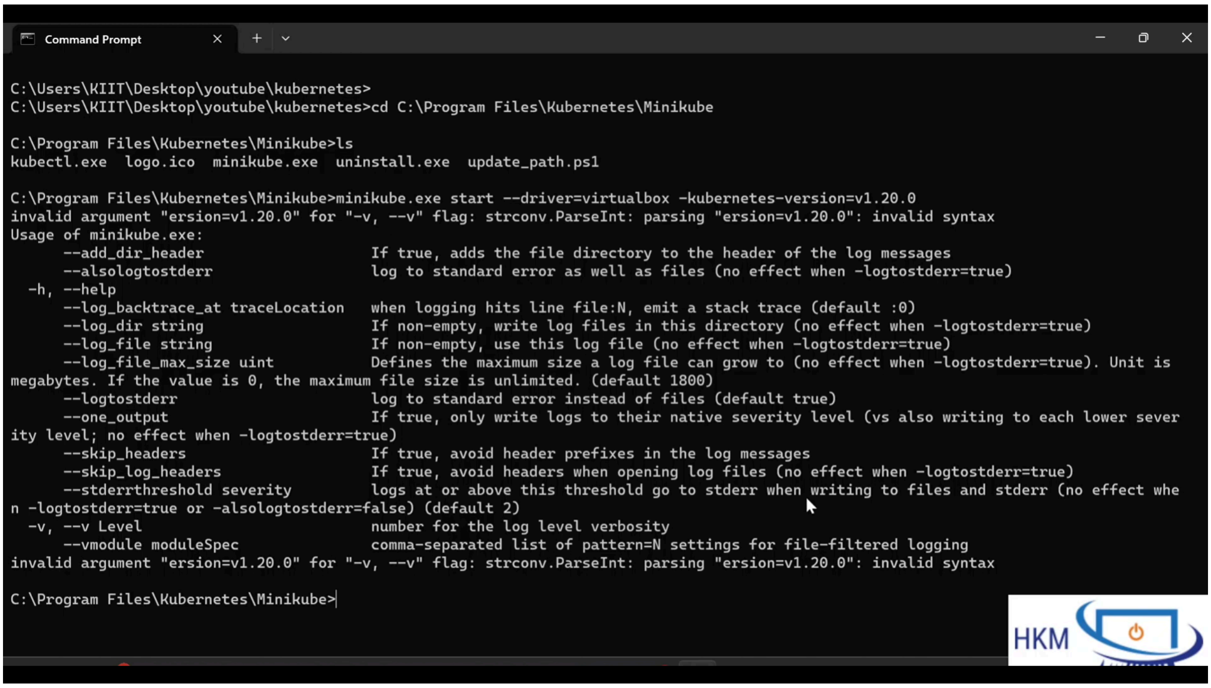
Enterprises that need to deploy and manage hundreds or thousands of Linux® containers and hosts can benefit from container orchestration.

Container orchestration is used to automate the following tasks at scale:

- Configuring and scheduling of containers
- Provisioning and deployment of containers
- Redundancy and availability of containers
- Scaling up or removing containers to spread application load evenly across host infrastructure
- Movement of containers from one host to another if there is a shortage of resources in a host, or if a host dies
- Allocation of resources between containers
- External exposure of services running in a container with the outside world
- Load balancing of service discovery between containers
- Health monitoring of containers and hosts

Kubernetes is an open source orchestration tool developed by Google for managing microservices or containerized applications across a distributed cluster of nodes.
Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.
Kubernetes is a popular container orchestration tool similar to docker swarm, K8 is mostly used to manage the containers, it is also used for blue-green deployments, also use to scale the containers.
Kubernetes provides highly resilient infrastructure with zero downtime deployment capabilities, automatic rollback, scaling, and self-healing of containers (which consists of auto-placement, auto-restart, auto-replication, and scaling of containers on the basis of CPU usage).

Output:

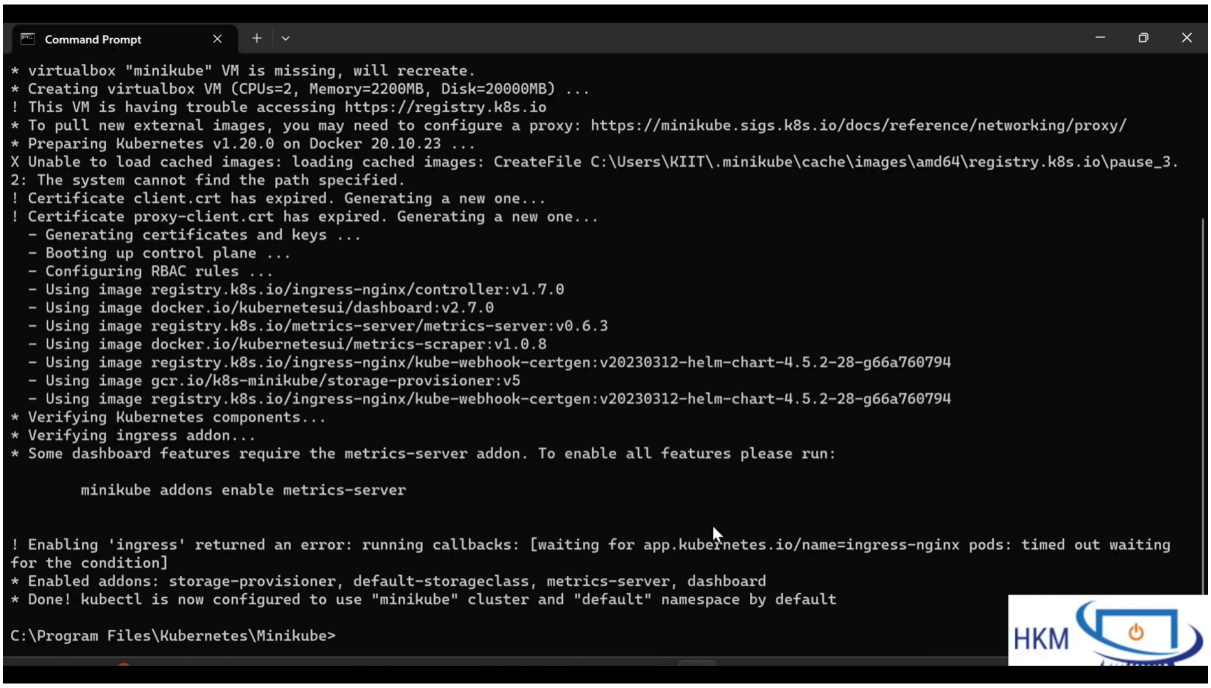


```
C:\Users\KIIT\Desktop\youtube\kubernetes>
C:\Users\KIIT\Desktop\youtube\kubernetes>cd C:\Program Files\Kubernetes\Minikube

C:\Program Files\Kubernetes\Minikube>ls
kubectl.exe  logo.ico  minikube.exe  uninstall.exe  update_path.ps1

C:\Program Files\Kubernetes\Minikube>minikube.exe start --driver=virtualbox -kubernetes-version=v1.20.0
invalid argument "ersion=v1.20.0" for "-v, --v" flag: strconv.ParseInt: parsing "ersion=v1.20.0": invalid syntax
Usage of minikube.exe:
      --add_dir_header                   If true, adds the file directory to the header of the log messages
      --alsologtostderr                  log to standard error as well as files (no effect when -logtostderr=true)
  -h, --help
      --log_backtrace_at traceLocation   when logging hits line file:N, emit a stack trace (default :0)
      --log_dir string                   If non-empty, write log files in this directory (no effect when -logtostderr=true)
      --log_file string                  If non-empty, use this log file (no effect when -logtostderr=true)
      --log_file_max_size uint           Defines the maximum size a log file can grow to (no effect when -logtostderr=true). Unit is
megabytes. If the value is 0, the maximum file size is unlimited. (default 1800)
      --logtostderr                      log to standard error instead of files (default true)
      --one_output                       If true, only write logs to their native severity level (vs also writing to each lower sever
ity level; no effect when -logtostderr=true)
      --skip_headers                     If true, avoid header prefixes in the log messages
      --skip_log_headers                 If true, avoid headers when opening log files (no effect when -logtostderr=true)
      --stderrthreshold severity         logs at or above this threshold go to stderr when writing to files and stderr (no effect whe
n -logtostderr=true or -alsologtostderr=false) (default 2)
  -v, --v Level                          number for the log level verbosity
      --vmodule moduleSpec               comma-separated list of pattern=N settings for file-filtered logging
invalid argument "ersion=v1.20.0" for "-v, --v" flag: strconv.ParseInt: parsing "ersion=v1.20.0": invalid syntax

C:\Program Files\Kubernetes\Minikube>
```

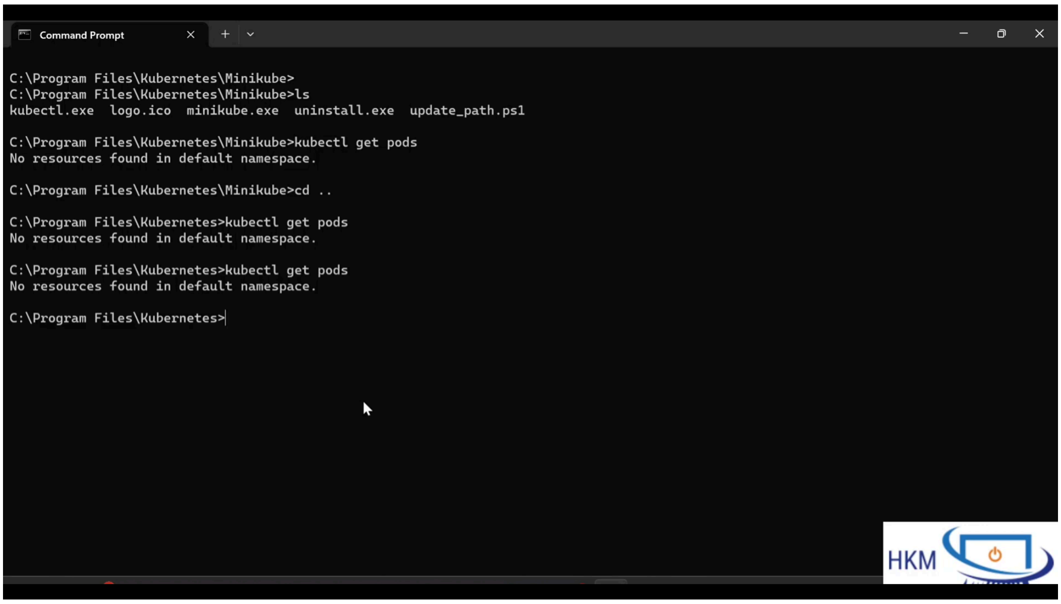


```
* virtualbox "minikube" VM is missing, will recreate.
* Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
! This VM is having trouble accessing https://registry.k8s.io
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.20.0 on Docker 20.10.23 ...
X Unable to load cached images: loading cached images: CreateFile C:\Users\KIIT\.minikube\cache\images\amd64\registry.k8s.io\pause_3.
2: The system cannot find the path specified.
! Certificate client.crt has expired. Generating a new one...
! Certificate proxy-client.crt has expired. Generating a new one...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
  - Using image registry.k8s.io/ingress-nginx/controller:v1.7.0
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
  - Using image registry.k8s.io/metrics-server/metrics-server:v0.6.3
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230312-helm-chart-4.5.2-28-g66a760794
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230312-helm-chart-4.5.2-28-g66a760794
* Verifying Kubernetes components...
* Verifying ingress addon...
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


! Enabling 'ingress' returned an error: running callbacks: [waiting for app.kubernetes.io/name=ingress-nginx pods: timed out waiting
for the condition]
* Enabled addons: storage-provisioner, default-storageclass, metrics-server, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Program Files\Kubernetes\Minikube>
```



```
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 the advantages of Kubernetes

Ans.

Kubernetes offers several advantages that make it a preferred choice for container orchestration in modern software development and deployment environments.

1. Scalability: Kubernetes enables automatic scaling of applications based on resource usage or predefined metrics. This ensures that your application can handle varying loads efficiently without manual intervention.
2. High Availability: With features like automatic restarts, self-healing, and rolling updates, Kubernetes enhances the availability of applications. It ensures that your services are always up and running, minimizing downtime and improving reliability.
3. Portability: Kubernetes provides a consistent environment across different infrastructure providers, making it easier to deploy and manage applications in diverse environments such as on-premises data centers, public clouds, or hybrid setups.
4. Resource Efficiency: By optimizing resource allocation and utilization through features like container density and bin packing, Kubernetes helps in maximizing resource efficiency, thereby reducing infrastructure costs.
5. Service Discovery and Load Balancing: Kubernetes automates service discovery and load balancing, making it easier to connect and route traffic to different parts of your application without manual intervention.

Overall, Kubernetes streamlines the deployment, scaling, and management of containerized applications, offering numerous benefits in terms of scalability, reliability, portability, efficiency, and security.