

Experiment No. 6

Title: Application deployment using Kubernetes (PaaS)

Batch: B-1**Roll No: 16010422234****Experiment No: 6**

Aim: Nodejs application deployment using Kubernetes (PaaS)

Resources needed: killercoda (online playground)

Theory:

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 Basics Modules

**Procedure:**

- 1) Explore all 1 to 4 modules.
 - 2) 10 kubectl commands on killercoda (online playground).
 - 3) Deploy an app on killercoda (online playground).
 - 4) Explore and Expose an app (preferably image created in docker experiment).
-

Results: (Result of Steps with screenshots)

The screenshot shows a web browser window with the URL <https://killercode.com/playgrounds/scenario/kubernetes>. The page title is "Kubernetes 1.31". On the left, there is a sidebar with a "START" button. The main area displays a terminal window with the following commands and output:

```
controlplane $ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
controlplane  Ready     control-plane  19d   v1.31.0
node01        Ready     <none>      19d   v1.31.0

controlplane $ kubectl get namespaces
NAME          STATUS    AGE
default       Active    19d
kube-node-lease  Active    19d
kube-public    Active    19d
kube-system    Active    19d
local-path-storage  Active    19d

controlplane $ kubectl create deployment myapp --image=nginx
deployment.apps/myapp created

controlplane $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
myapp-7947b7c86d-kp6nx  1/1     Running   0           20s

controlplane $ kubectl create deployment myapp2 --image=chandanagalali/myapp:v1
deployment.apps/myapp2 created

controlplane $ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
myapp         1/1     1             1           20m
myapp2        1/1     1             1           27s

controlplane $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
myapp-7947b7c86d-kp6nx  1/1     Running   0           20m
myapp2-8667f6f75-rh652  1/1     Running   0           40s

controlplane $ kubectl expose deployment myapp2 --type=NodePort --port=8080 --target-port=80 --name=myapp-service2
service/myapp-service2 exposed

controlplane $ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
myapp         1/1     1             1           20m
myapp2        1/1     1             1           2m15s

controlplane $ kubectl get services
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes    ClusterIP   10.96.0.1     <none>         443/TCP           19d
myapp-service2  NodePort    10.99.122.26 <none>         8080:31356/TCP   103s

controlplane $ kubectl get nodes -o wide
NAME          STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
controlplane  Ready     control-plane  19d   v1.31.0   172.30.1.2    <none>         Ubuntu 20.04.5 LTS   5.4.0-131-generic  containerd://1.7.13
node01        Ready     <none>      19d   v1.31.0   172.30.2.2    <none>         Ubuntu 20.04.5 LTS   5.4.0-131-generic  containerd://1.7.13

controlplane $ ^C
controlplane $
```

The screenshot shows a web application titled "Traffic Port Accessor". The subtitle is "Access HTTP services which run in your environment". Below the subtitle, there is a light blue box with the text: "The services need to run on all interfaces (like 0.0.0.0) and not just localhost".

The application displays two hosts, Host 1 and Host 2, each with a "Common Ports" section and a "Custom Ports" section.

Host 1

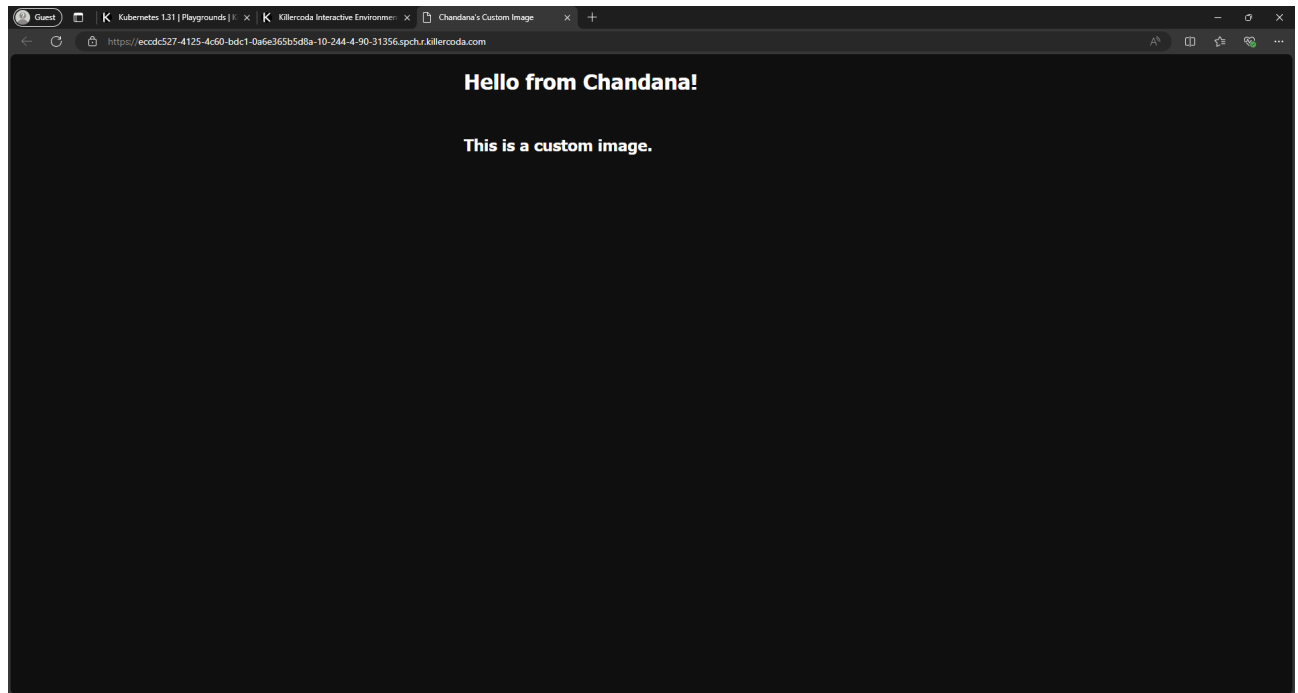
Common Ports: 80, 8080

Custom Ports: 31356, Access

Host 2

Common Ports: 80, 8080

Custom Ports: 1234, Access



Questions:

Q1. Explain micro-services with an example.

Ans: Microservices, also known as the microservices architecture, is a design style where an application is built as a collection of small, autonomous services modeled around a specific business domain. Each service is independently deployable, scalable, and loosely coupled with others. This approach allows teams to develop, test, and deploy features faster and more effectively.

Example:

An e-commerce application can be divided into several microservices:

- User Service: Manages user registration, authentication, and profiles.
- Product Service: Handles product catalog, details, and inventory.
- Order Service: Manages cart operations, order placements, and payment processing.
- Shipping Service: Manages shipment tracking and delivery status.

These services communicate with each other via lightweight APIs (e.g., REST or gRPC) while remaining independent in functionality and deployment.

Outcomes: CO3 – Analyze different cloud architectures and IoT-cloud.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

The experiment demonstrated the deployment and management of a Node.js application using Kubernetes on Killercoda, showcasing key features like container orchestration, declarative configuration, and service exposure, emphasizing Kubernetes as a powerful tool for modern PaaS solutions.

Grade: AA / AB / BB / BC / CC / CD / DD

Signature of faculty in-charge with date

References:

Websites:

- 1) <https://kubernetes.io/docs/tutorials/kubernetes-basics/>
 - 2) <https://killercoda.com/playgrounds/scenario/kubernetes>
 - 3) 5 steps to Deploy docker image to Kubernetes
 - 4) <https://www.youtube.com/watch?v=95zmJnz4iOo>
-

