THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

End-to-End Docker and Kubernetes Deployment of a Node.js Application with BusyBox Utilities

Description:

- **Purpose:**
  This document outlines the full lifecycle of containerizing and deploying a simple Node.js application using Docker and Kubernetes. It demonstrates how individual development steps integrate into a DevOps workflow.
- **Application Setup:**
  A lightweight Node.js app was created to simulate a basic HTTP server. The app is designed to return the hostname of the container it is running in, which helps visualize load balancing in Kubernetes later.
- **Dockerization:**
  The application was containerized by writing a Docker file, building a custom Docker image, and running it locally. Docker image and container management commands were used to verify functionality and performance. The app was then tested on localhost to confirm it was working as expected.
- **Image Management:**
  The built Docker image was tagged and pushed to a container image registry. This made the image available for use across different environments and platforms, an essential step for modern CI/CD pipelines.
- **Container Debugging and Shell Access:**
  Running containers were listed, and shell access into containers was demonstrated using interactive execution. This helps understand how to inspect and troubleshoot running applications inside containers.
- **Kubernetes Deployment:**
  After container verification, the image was deployed into a Kubernetes cluster using kubectl. A deployment was created, followed by a service of type NodePort to expose the app externally. This demonstrated the basics of orchestrating containers at scale.

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

1. Run busy box image

```
[thisara@localhost ~]$ sudo docker run busybox echo "hello world"
hello world
[thisara@localhost ~]$
```

2. Create app.js file

```
 GNU nano 2.9.8                                    app.js

const http = require('http');
const os = require('os');
console.log("Kubia server starting ... ");
var handler = function(request, response) {
console. log("Received request from " + request.connection. remoteAddress);
response.writeHead(200);
response.end("You've hit " + os.hostname() + "\n");

var www = http.createServer(handler);
www.listen(8080);
```

3. Check docker status

```
[thisara@localhost ~]$ sudo systemctl status docker
[sudo] password for thisara:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2025-06-23 01:03:21 PDT; 6h ago
     Docs: https://docs.docker.com
 Main PID: 1468 (dockerd)
    Tasks: 17
   Memory: 155.3M
   CGroup: /system.slice/docker.service
           └─1468 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

4. Create Docker file

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.8                                  Dockerfile

FROM node:7
ADD app.js /app.js
ENTRYPOINT ["node", "app.js"]
```

5. Build the docker image

```
[thisara@localhost ~]$ sudo docker build -t kubia .
[+] Building 193.6s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 158B
 => [internal] load metadata for docker.io/library/node:7
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 448B
 => [1/2] FROM docker.io/library/node:7@sha256:af5c2c6ac8bc3fa372ac03
```

6. Show the images

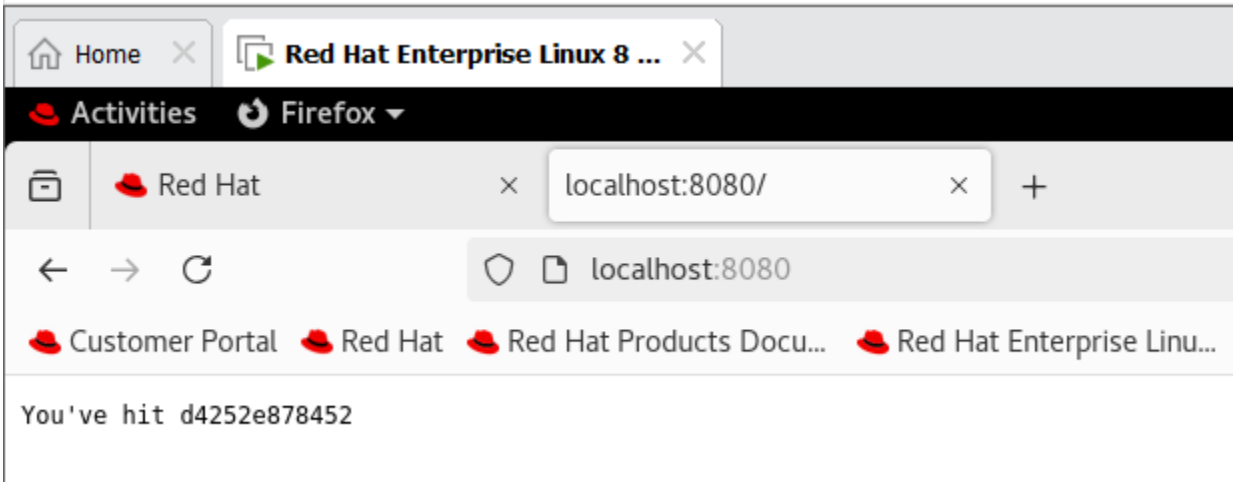
```
[thisara@localhost ~]$ sudo docker images
[sudo] password for thisara:
REPOSITORY                                TAG        IMAGE ID       CREATED         SIZE
kubia                                     latest     b4578cf9aec5   2 minutes ago   660MB
gcr.io/k8s-minikube/kicbase               v0.0.47    795ea6a69ce6   4 weeks ago     1.31GB
registry.k8s.io/kube-apiserver            v1.30.13   e60416bd78b6   5 weeks ago     116MB
registry.k8s.io/kube-scheduler            v1.30.13   12675fc0a409   5 weeks ago     62.5MB
registry.k8s.io/kube-controller-manager   v1.30.13   c32cfb72bacc   5 weeks ago     111MB
```

7. Run the docker image

```
[thisara@localhost ~]$ sudo docker run --name kubia-container -p 8080:8080 -d kubia
87ba71e55c0ea0b26f81150d93fba5da8bdc2de2151d49a01d4856937e9bfc1b
[thisara@localhost ~]$
[thisara@localhost ~]$
```

8. Access the app using localhost

```
[thisara@localhost ~]$ sudo docker run --name kubia-container -p 8080:8080 -d kubia
d4252e878452c6d4f966e598da36c3c19e9f0d99f65284cbcfae2a9e12643ef4
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$ curl localhost:8080
You've hit d4252e878452
[thisara@localhost ~]$
```



9. List running containers

```
[thisara@localhost ~]$ sudo docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS
                                   NAMES
d4252e878452   kubia                  "node /app.js"           3 minutes ago   Up 3 minutes   0.0.0
.0:8080->8080/tcp, :::8080->8080/tcp   kubia-container
c55ac7aedc02   12675fc0a409           "kube-scheduler --au…"   7 hours ago     Up 7 hours
                                   k8s_kube-scheduler_kube-scheduler-localhost.localdomain_kube-syste
m_f2b510c0e1c8ab16519ae8aa551d1a25_2
b3c53b8984d0   c32cfb72bacc           "kube-controller-man…"   7 hours ago     Up 7 hours
                                   k8s_kube-controller-manager_kube-controller-manager-localhost.loca
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

10. Run the shell inside the execution command

```
[thisara@localhost ~]$ sudo docker exec -it kubia-container bash
root@d4252e878452:/#
root@d4252e878452:/# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3 813608 25920 ?        Ssl  15:15   0:00 node /app.js
root          12  0.6  0.0  20252  3116 pts/0    Ss   15:24   0:00 bash
root          18  0.0  0.0  17508  2120 pts/0    R+   15:24   0:00 ps aux
root@d4252e878452:/# S
```

11. Push the image to image registry

```
[thisara@localhost ~]$ sudo docker push thisarakandage/kubia
Using default tag: latest
The push refers to repository [docker.io/thisarakandage/kubia]
a4cacb2a1e07: Pushed
ab90d83fa34a: Mounted from library/node
8ee318e54723: Mounted from library/node
e6695624484e: Mounted from library/node
da59b99bbd3b: Mounted from library/node
```

12. Run the docker image

```
[thisara@localhost ~]$ sudo docker run -p 3000:3000 -d thisarakandage/kubia
543c97036d1d2e401308479c103b881c497a118d24430a8401ff2b39c5827f2f
[thisara@localhost ~]$
[thisara@localhost ~]$
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

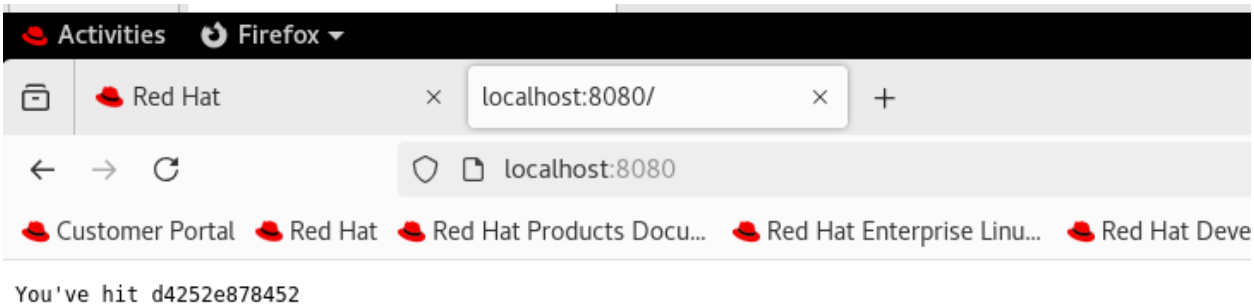
E-mail LinkedIn GitHub Website

13. Deploy it in Kubernetes cluster

```
[thisara@localhost ~]$ minikube start
😄  minikube v1.36.0 on Redhat 8.10
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.47 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.58.2:8443
CoreDNS is running at https://192.168.58.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   6d    v1.33.1
[thisara@localhost ~]$
```

```
[thisara@localhost ~]$ kubectl create deployment kubia --image=thisarakandage/kubia
deployment.apps/kubia created
[thisara@localhost ~]$ kubectl expose deployment kubia --type=NodePort --port=8080
service/kubia exposed
[thisara@localhost ~]$
```




```
You've hit d4252e878452
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Summary

- Learned how to write and prepare a basic Node.js application for containerization.
- Gained hands-on experience with Docker:
    - Created a Dockerfile.
    - Built a custom Docker image.
    - Ran and tested the image locally using port forwarding.
    - Verified container status and accessed container logs.
- Practiced pushing Docker images to a remote image registry (e.g., Docker Hub) for reuse in other environments.
- Explored container management:
    - Listed running and exited containers.
    - Used interactive shell access inside containers for debugging.
- Deployed the application to a Kubernetes cluster:
    - Created a Kubernetes deployment using kubectl.
    - Exposed the deployment using a NodePort service.
    - Verified service availability and cluster status.
- Understood key DevOps concepts including containerization, orchestration, and registry-based deployment.
- Improved practical knowledge of the software deployment lifecycle using Docker and Kubernetes in a local environment.