Task 3: Expose with LoadBalancer

Objective: Expose the Node.js application to external traffic by creating a LoadBalancer-type service.

Steps Taken

1. Created a Service manifest (service.yaml) of type LoadBalancer to expose port **80** mapped to container port **3000**.
2. Applied the Service manifest using **`kubectl apply -f service.yaml`**
3. **<u>On local device:</u>**

   Minikube automatically routes loadbalancer on local devices, which makes the external IP appear pending forever. The external IP can be retrieved using **`minikube service nodejs-service --url`**, which opens a temporary local proxy to expose the service. Alternatively, **`minikube tunnel`** can be used to simulate a real cloud LoadBalancer by assigning a routable external IP.
4. Confirmed LoadBalancer service routing using **`curl <external-ip>`**

Screenshots

- **Screenshot 1:** kubectl apply -f service.yaml

```
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s\manifests> kubectl apply -f service.yaml
service/nodejs-service created
```

- **Screenshot 2:** kubectl get svc nodejs-service –watch

```
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s\manifests> kubectl get svc nodejs-service --watch
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
nodejs-service   LoadBalancer   10.108.218.129   <pending>     80:30347/TCP   2s
```

- The External-IP will be pending forever if minikube tunnel isn't enabled

- **Screenshot 3 (Option A):** minikube service nodejs-serivce --url

Get the external IP

```
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s\manifests> docker context use default
default
Current context is now "default"
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s\manifests> minikube service nodejs-service --url
http://127.0.0.1:51688
❗  Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

Screenshot 4 (Option B):
a - `minikube tunnel`

```
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s\manifests> minikube tunnel
✅  Tunnel successfully started

📌  NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

❗  Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: https://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-on-windows-requires-root-permission
🏃  Starting tunnel for service nodejs-service.
```

b- `kubectl get svc nodejs-service`

```
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s> kubectl get svc nodejs-service
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
nodejs-service   LoadBalancer   10.108.218.129   127.0.0.1     80:30347/TCP   20m
```

Screenshot 5: curl http://127.0.01
Verifying the external IP

```
PS C:\Users\Laila\Desktop\docker-tasks\week 2\nodeJS--as-k8s> curl http://127.0.0.1


StatusCode        : 200
StatusDescription : OK
Content           : Hello World from Node.js on Kubernetes!

RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Keep-Alive: timeout=5
                    Content-Length: 40
                    Content-Type: text/plain
                    Date: Sat, 02 Aug 2025 06:35:20 GMT

                    Hello World from Node.js on Kubernetes!

Forms             : {}
Headers           : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Content-Length, 40], [Content-Type,
                    text/plain]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 40
```

☑ Outcome

The application was successfully exposed to external traffic using a LoadBalancer service. It was accessible via the provided external URL.