

# Rapport de Projet : Déployer une Image Docker dans Kubernetes

Aicha NASIH  
Salma HERMAK  
Khadija BOUYOUSSEF

## **Sommaire :**

<b><u>SOMMAIRE :</u></b>	<b>1</b>
<b><u>INTRODUCTION</u></b>	<b>2</b>
<b><u>OBJECTIFS DU PROJET</u></b>	<b>2</b>
<b><u>TECHNOLOGIES UTILISÉES</u></b>	<b>2</b>
<b><u>ARBORESCENCE DU PROJET</u></b>	<b>2</b>
<b><u>ÉTAPES DE RÉALISATION :</u></b>	<b>2</b>
<b>DOCKERFILE</b>	<b>2</b>
<b>CONSTRUCTION DE L'IMAGE</b>	<b>3</b>
<b>LE CODE SOURCE DE INDEX.JS</b>	<b>3</b>
<b>CRÉATION DU FICHIER PACKAGE.JSON :</b>	<b>4</b>
<b>PUBLICATION SUR DOCKER HUB</b>	<b>4</b>
<b>FICHIER DEPLOYMENT.YAML</b>	<b>5</b>
<b>FICHIER SERVICE.YAML</b>	<b>6</b>
<b>ACCÈS À L'APPLICATION</b>	<b>7</b>
<b><u>CONCLUSION :</u></b>	<b>8</b>

## Introduction

Dans le cadre de ce projet, nous avons mis en œuvre un processus de conteneurisation d'une application Node.js, suivi de son déploiement dans un cluster Kubernetes. L'objectif est d'illustrer la manière dont les technologies modernes permettent l'automatisation, la portabilité et la scalabilité des applications web.

## Objectifs du projet

- Construire une image Docker à partir d'une application Node.js.
- Publier l'image sur Docker Hub.
- Créer les fichiers de configuration Kubernetes nécessaires.
- Déployer l'application dans un cluster Kubernetes (via Docker Desktop).
- Exposer le service localement et valider son bon fonctionnement.

## Technologies utilisées

Technologie	Version / Détail
Node.js	18-alpine
Docker	Docker Desktop (Windows)
Kubernetes	Intégré à Docker Desktop
Docker Hub	Pour héberger l'image
YAML	Pour les fichiers de config

## Arborescence du projet

```
pgsql
Copy code
ProjetNode.js/
  └── app/
    ├── index.js
    └── Dockerfile
  └── deployment.yaml
  └── service.yaml
```

## Étapes de réalisation :

### Dockerfile

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json .
RUN npm install
COPY . .
```

```
EXPOSE 8000
CMD ["node", "index.js"]
```

## Construction de l'image

### Le code source de index.js

```
const http = require('http');
const hostname = '0.0.0.0';
const port = 8000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/html');
  res.end(`

<html>
  <head>
    <title>Kubernetes Project</title>
    <style>
      body {
        background: linear-gradient(135deg, #1e1e2f, #282a36);
        color: #ffffff;
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
        text-align: center;
        padding-top: 100px;
      }
      h1 {
        font-size: 3em;
        color: #00ffcc;
        text-shadow: 0 0 10px #00ffcc, 0 0 20px #00ffff;
        animation: pulse 2s infinite;
      }
      h2 {
        color: #ff66cc;
        margin-top: 20px;
        font-size: 1.5em;
      }
      p {
        font-size: 1.2em;
        color: #f8f8f2;
      }
    @keyframes pulse {
      0% { transform: scale(1); }
      50% { transform: scale(1.05); }
      100% { transform: scale(1); }
    }
  </style>
</head>
```

```

<body>
  <h1>🚀 Project Successfully Deployed on Kubernetes!</h1>
  <h2>❖ This project is proudly presented by:</h2>
  <p>Khadija Bouyoussef<br>Aicha Nasih<br>Salma Hermak</p>
</body>
</html>
`);
`);

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}`);
});

```

#### Création du fichier package.json :

1. Crée un fichier package.json:

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js\app>npm init -y
Wrote to C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js\app\package.json:

{
  "name": "projet-nodejs",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

2. Ajouter Les dépendances :

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js\app>npm install express
added 66 packages, and audited 67 packages in 19s
14 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

#### Publication sur Docker Hub

1. Connexion à Docker Hub :

```
docker login
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js\app>docker login  
Authenticating with existing credentials... [Username: khadijal11111]  
! Info → To login with a different account, run 'docker logout' followed by 'docker login'  
  
Login Succeeded
```

## 2. Pousser l'image :

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js\app>docker push khadijal11111/projet-nodejs:latest  
The push refers to repository [docker.io/khadijal11111/projet-nodejs]  
71faaf76c478: Pushed  
dd71dde834b5: Pushed  
7d187507aec8: Pushed  
1e5a4c89cee5: Pushed  
ba839c204f6f: Pushed  
55d1c0f3f21c: Pushed  
25ff2da83641: Pushed  
f18232174bc9: Pushed  
1d3b31c9d11d: Pushed  
latest: digest: sha256:9e64aec42036c7a48f80c5e5dba3e7992c0133efb4292c7a8aa499912c885d58 size: 856
```

## 3. Construire l'image :

```
cd ProjetNode.js/app  
docker build -t ton-utilisateur-dockerhub/projet-nodejs:latest .
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js\app>docker build -t khadijal11111/projet-nodejs:latest .  
[+] Building 21.8s (11/11) FINISHED docker:desktop-linux  
=> [internal] load build definition from Dockerfile 0.1s  
=> => transferring dockerfile: 161B 0.0s  
=> [internal] load metadata for docker.io/library/node:18-alpine 9.3s  
=> [auth] library/node:pull token for registry-1.docker.io 0.0s  
=> [internal] load .dockerrcignore 0.0s  
=> => transferring context: 2B 0.0s  
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e 0.1s  
=> => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e 0.1s  
=> [internal] load build context 4.5s  
=> => transferring context: 2.34MB 4.5s  
=> CACHED [2/5] WORKDIR /app 0.0s  
=> [3/5] COPY package*.json ./ 0.0s  
=> [4/5] RUN npm install 0.0s  
=> [5/5] COPY . 0.3s  
=> exporting to image 1.6s  
=> => exporting layers 0.6s  
=> => exporting manifest sha256:89c8fdb4c6c9b545fa85c9d60017293431603d545c19ed53aa911e88f87c209d 0.0s  
=> => exporting config sha256:55b31f60ca2cf6038e7b0ea4b2f31590fb63003ef4d98638dac0d582bf5625c 0.0s  
=> => exporting attestation manifest sha256:ab873bcba410e894822ce5b92b7cef1f97b9802c47e23fb650ea939666fd74a725 0.1s  
=> => exporting manifest list sha256:9e64aec42036c7a48f80c5e5dba3e7992c0133efb4292c7a8aa499912c885d58 0.0s  
=> => naming to docker.io/khadijal11111/projet-nodejs:latest 0.0s  
=> => unpacking to docker.io/khadijal11111/projet-nodejs:latest 0.0s  
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/59p8hmguemjfepmwqncypniz2 0.8s
```

## Fichier deployment.yaml

### 1. Le code source deployment.yaml :

```
apiVersion: apps/v1  
  
kind: Deployment  
  
metadata:  
  
  name: projet-nodejs-deployment  
  
spec:  
  
  replicas: 2
```

```
selector:

matchLabels:

  app: projet-nodejs

template:

metadata:

labels:

  app: projet-nodejs

spec:

containers:

- name: projet-nodejs

  image: khadija111111/projet-nodejs:latest

  ports:

- containerPort: 8000
```

## 2. Appliquer les fichiers de déploiement :

```
kubectl apply -f deployment.yaml
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js>kubectl apply -f deployment.yaml
deployment.apps/projet-nodejs-deployment created
```

## Fichier service.yaml

### 1. Le code source service.yaml :

```
apiVersion: v1

kind: Service

metadata:

  name: projet-nodejs-service

spec:

  type: NodePort

  selector:
```

```
app: projet-nodejs
```

```
ports:
```

```
- protocol: TCP
```

```
port: 80
```

```
targetPort: 8000
```

## 2. Appliquer les fichiers de déploiement :

```
kubectl apply -f service.yaml
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js>kubectl apply -f service.yaml
service/projet-nodejs-service created
```

## Accès à l'application

### 1. Vérification que les pods sont bien lancés :

```
kubectl get pods
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js>kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
projet-nodejs-deployment-554f679b8d-lx8wf   1/1     Running   0          20m
projet-nodejs-deployment-678cbfcc8d-qlhgk   0/1     CrashLoopBackOff   6 (7m5s ago)   13m
projet-nodejs-deployment-678cbfcc8d-wf58w   0/1     CrashLoopBackOff   7 (2m1s ago)   13m
```

### 2. Vérifie le service :

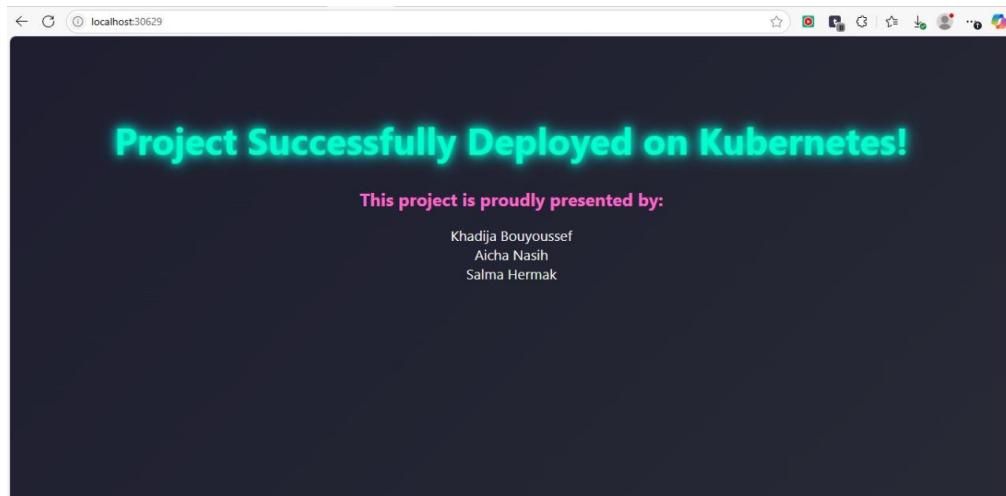
```
kubectl get svc
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js>kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes     ClusterIP  10.96.0.1    <none>        443/TCP   11d
projet-nodejs-service   NodePort   10.97.76.161  <none>        80:30629/TCP   16m
```

### 3. Vérifie les logs :

```
kubectl logs projet-nodejs-deployment-554f679b8d-lx8wf
```

```
C:\Users\user\Documents\SchoolProjects\kubernetes\ProjetNode.js>kubectl logs projet-nodejs-deployment-554f679b8d-lx8wf
Serveur démarré sur http://localhost:8000
```



## Conclusion :

Ce projet nous a offert une opportunité concrète de mettre en œuvre toutes les étapes clés du déploiement d'une application web dans un environnement Kubernetes. De la construction et la publication d'une image Docker personnalisée à la configuration des ressources Kubernetes, en passant par le déploiement et l'exposition de l'application, nous avons acquis une expérience pratique précieuse.

Nous avons renforcé notre compréhension de Docker comme outil de conteneurisation efficace, ainsi que de Kubernetes en tant que plateforme puissante pour l'orchestration et la gestion des applications conteneurisées. Ce travail nous a permis d'appréhender les enjeux de l'automatisation, de la scalabilité et de l'efficacité dans les pratiques modernes de développement et d'exploitation.

Il s'agit d'une expérience formatrice qui nous prépare aux réalités du monde professionnel dans le domaine du DevOps et du déploiement d'applications cloud-native.