



# Compte Rendu

## Orchestration de conteneurs avec Kubernetes

Création d'un cluster Kubernetes local pour  
déployer une application conteneurisée

**Présenté par :**

**OUAFIR SAAD**

**NAIT OUAHMAN ABDELLAH**

**OUJAJA MOHAMMED**

**Encadrant :**

**BADR EDDINE CHERKAOUI**

**ESTSB**

**CSTC 2024/2025**

# Table de Contenu

Table de Contenu .....	2
Présentation .....	3
Objectifs du Projet.....	4
Outils Utilisés.....	4
Définitions Générales en K8S.....	6
Configuration de l'environnement de travail.....	7
Les étapes de la réalisation du projet .....	9
Déploiement de l'application sur k8s.....	11
Ressource.....	15

# Présentation

La conteneurisation est une solution qu'elle consiste à emballer une application et tout ce qu'elle a besoin pour fonctionner (comme les os, dépendances, les fichiers du projet ...) dans un "**conteneur**", ce dernier est comme une boîte qui contient l'application et tout ce qu'elle doit avoir pour fonctionner, peu importe où elle est lancée.

Dans un environnement de développement moderne où les applications sont créées et testées, cette méthode est considérée comme une bonne solution pour plusieurs raisons :

- **Portabilité** : c'est à dire que l'application peut fonctionner exactement comme s'elle est exécutée sur l'ordinateur du développeur ou sur un serveur ou dans le cloud, sans rencontrer des problèmes.
- **Scalabilité** : c'est à dire qu'il est simple d'ajouter des ressources supplémentaires. Par exemple, si de nombreux utilisateurs accèdent à notre application simultanément, on peut ajouter d'autres instances facilement pour assurer une qualité de service plus fluide.
- **Efficacité des déploiements** : c'est à dire le déploiement de l'application sera plus rapide et plus facile, car tout est déjà emballé et prêt à être utilisé et on ne serait plus besoin de la configurer à chaque fois.

# Objectifs du Projet

- Configurer un environnement de développement local en utilisant Minikube pour simuler un cluster Kubernetes.
- Conteneuriser une application simple à l'aide de Docker, en créant des images de conteneurs adaptées aux besoins de l'application.
- Déployer l'application sur le cluster Kubernetes local, en utilisant les meilleures pratiques pour la gestion des ressources et la mise à l'échelle.

## Outils Utilisés

### 1. Docker

Docker est un outil qu'il permet de créer, déployer et exécuter des applications sous forme de conteneurs.

Les conteneurs docker encapsulent l'application et toutes ses dépendances, pour garantir que l'application fonctionne sans problèmes dans différents environnements.

Pour notre projet, on va utiliser docker pour :

- Créer des images Docker à partir de notre application.
- Créer et séparer les conteneurs de l'application.
- Tester les conteneurs avant de le déployer avec Kubernetes.

### 2. Minikube

Minikube est un outil qui facilite le déploiement d'un **cluster** Kubernetes local sur une machine de développement.

Il simule un environnement Kubernetes en se basant sur une seule instance VM (machine virtuelle), permettant aux développeurs de tester et de développer des applications conteneurisées sans avoir besoin d'un cluster Kubernetes complet. Dans ce projet, Minikube sera utilisé pour :

- Initialiser un cluster Kubernetes local.
- Simplifier l'interaction avec l'API Kubernetes à partir de la ligne de commande.
- Fournir un environnement isolé pour le développement et les tests.

### 3. Kubernetes

Kubernetes est un système d'orchestration open-source pour gérer des applications conteneurisées à grande échelle.

Kubernetes offre des outils puissants pour le déploiement, la mise à l'échelle et la gestion des applications en garantissant la haute disponibilité.

# Définitions Générales en K8S

**Pod** : un ou plusieurs conteneurs qu'ils doivent être considérée et contrôlée comme une seule application.

**Node** : représente une machine (réelle, virtuelle ou conteneurise) qu'elle contient un ou plusieurs Pods.

**Cluster** : représente un ensemble des Nodes qui sont interconnectés entre eux.

**Deployment** : un objet qui définit l'état souhaité pour une application conteneurisée déployée sur un cluster

**Workload** : représente l'application qu'elle est en exécution.

**Orchestration** : représente l'automatisation des efforts opérationnels pour exécuter le Workload.

**Service** : un groupe de Pods.

# Configuration de l'environnement de travail

Dans cette étape on doit installer et configurer tous les outils qu'on a besoin de, afin de réaliser notre projet.

- Linux ou n'importe quel système d'exploitation
- Installer et configurer Docker Engine
- Installer et configurer k8s
- Installer et configurer Minikube

## Docker Engine

Pour installer docker engine il faut suivre la documentation officielle du Docker, et choisir le OS que vous utilisez (dans notre cas en va utiliser **UBUNTU 20.04.06**)

```
Terminal
# Ajouter la cle GPG du Docker
sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Ajouter le repertoire au fichier sources:
echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

```
# installer les dernières versions du package docker

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin

# verifier que docker etait bien installe à travers de démarrer l'image hello-world

sudo docker run hello-world
```

## Minikube

```
Terminal

#

sudo apt-get update

apt-get update

#

curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb

sudo dpkg -i minikube_latest_amd64.deb
```

## Kubernetes

```
Terminal

# télécharger la dernière version du Kubernetes

curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

# telecharger le fichier checksum

curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256

# valider le checksum (un message de 'kubectl: OK' va être affiche en cas valide)

echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

# installer Kubernetes

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client #pour vérifier la version et assurer que k8s est bien installe
```



# Les étapes de la réalisation du projet

Premièrement en doit créer le projet (code source de l'application), dans ce cas en va adopter une petite application TODO-LIST en utilisant Express JS et Mongo DB.

La 2eme étape sera la création docker file pour configurer l'image et les dépendances de l'application :

```
# syntax=docker/dockerfile:1

ARG NODE_VERSION=19.5.0

FROM node:${NODE_VERSION}-alpine

ENV NODE_ENV production

WORKDIR /usr/src/app

RUN --mount=type=bind,source=package.json,target=package.json \
    --mount=type=bind,source=package-lock.json,target=package-lock.json \
    --mount=type=cache,target=/root/.npm \
    npm ci --include=dev

RUN npm install -g nodemon

COPY . .

RUN chown -R node /usr/src/app

USER node

EXPOSE 3000

CMD npm run dev
```

L'étape suivante c'est de la configuration du docker-compose pour tester les images localement avant publier l'application avec k8s.

```
services:

  todo-app:

    build:

      context: ./app

    depends_on:

      - todo-database

    environment:

      NODE_ENV: production

    ports:

      - 3000:3000

      - 35729:35729

    develop:

      watch:

        - path: ./app/package.json

          action: rebuild

        - path: ./app

          target: /usr/src/app

          action: sync

  todo-database:
```

```
image: mongo:4.4

ports:
- 27017:27017
```

il faut s'authentifier avec les coordonnées de docker hub

```
docker login
```

Builder l'image :

```
docker build -t username20242310/todo-app:latest .
```

Publier docker image sur dockerhub :

```
docker push username20242310/todo-app:latest
```

## Déploiement de l'application sur k8s

Premièrement on doit commencer Minikube sur un cluster k8s :

```
minikube start
```

Après on doit créer 2 fichiers YAML (Manifest) pour configurer le déploiement de l'application et de la base de données

# app-deployment.yaml :

```
apiVersion: apps/v1

kind: Deployment

metadata:

  name: todo-app
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: todo-app
  template:
    metadata:
      labels:
        app: todo-app
    spec:
      containers:
        - name: todo-app
          image: username20242310/todo-app:latest
          ports:
            - containerPort: 3000
          env:
            - name: NODE_ENV
              value: production
            args: ["npm", "start"]
---
apiVersion: v1
kind: Service
metadata:
```

```
name: todo-app

spec:

  type: NodePort

  ports:

    - port: 3000

      targetPort: 3000

      nodePort: 30001

  selector:

    app: todo-app
```

## mongo-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todo-database
spec:
  replicas: 1
  selector:
    matchLabels:
      app: todo-database
  template:
    metadata:
```

```
  labels:
    app: todo-database
spec:
  containers:
    - name: mongo
      image: mongo:4.4
      ports:
        - containerPort: 27017
      volumeMounts:
        - mountPath: /data/db
          name: mongo-storage
  volumes:
    - name: mongo-storage
      emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: todo-database
spec:
  type: ClusterIP
  ports:
    - port: 27017
```

```
selector:  
  
  app: todo-database
```

En commence le déploiement par l'application de la configuration des Déploiements Kubernetes :

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

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

On peut vérifier si l'application est en cours de déploiement :

```
Kubectl get all
```

On peut accéder à l'application par :

```
Minikube service todo-app
```

## Ressource

<https://hub.docker.com/>

<https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2F86-64%2Fstable%2F.exe+download>

<https://www.getambassador.io/blog/deploy-first-application-kubernetes-step-by-step-tutorial>

<https://www.youtube.com/watch?v=jTggu1HiKyY&list=PLX1bWGeBRhDCHijCrMO5F-oHg52rRBpl>

