**Projet Kubernetes**

Guide d’installation et déploiement

Cours :

DevSecOps – ESILV

Participantes :

Reizène Ben Hassine

Maureen Drouard

Date :

12/12/2024

Sommaire

[Présentation 3](#_Toc184838705)

[1. Phase de préparation 4](#_Toc184838706)

[a) Définir l’architecture de l’application et les ressources nécessaire 4](#_Toc184838707)

[b) Rédiger les Dockerfiles pour conteneuriser l’application web et les scripts de base de données 4](#_Toc184838708)

[c) Configurer l’environnement Kubernetes et Docker 5](#_Toc184838709)

[2. Phase de développement 7](#_Toc184838710)

[a) Développer l’application web 7](#_Toc184838711)

[b) Créer et tester les images Docker des conteneurs 7](#_Toc184838712)

[3. Phase de Déploiement 9](#_Toc184838713)

[a) Créer les fichiers YAML, pour les pods, Services, ConfigMaps, Secret et PersistentVolumeClaims 9](#_Toc184838714)

[b) Déployer les Pods dans Kubernetes et configuration des services 10](#_Toc184838715)

[c) En cas d’erreurs 10](#_Toc184838716)

[4. Phase de Validation et Test 11](#_Toc184838717)

[a) Vérifier que l’application web est accessible depuis l’extérieur du cluster via l'URL ou l’IP du Service. 11](#_Toc184838718)

[b) Tester les fonctionnalités de l’application web, notamment l’accès aux données. 12](#_Toc184838719)

[c) Effectuer des tests de résilience en redémarrant les Pods pour s'assurer de la persistance des données. 12](#_Toc184838720)

[5. Phase de Documentation et Livraison 13](#_Toc184838721)

## Présentation

Cette documentation consiste à décrire les étapes pour installer et déployer une application web (Node.js) avec une base de données MySQL sur un cluster Kubernetes. Ce projet utilise notamment Docker pour la conteneurisation des composants de l’application et de la base de données.

**Objectifs :**

* Déployer une BDD comme un Pod Kubernetes avec persistance des données
* Déployer une app web conteneurisée en utilisant Docker
* Créer des services pour la communication interne et externe entre les composants de l’application

**Outils utilisés :**

* Docker : pour les conteneurs
* Minikube : pour créer et gérer les cluster Kubernetes
* Kubectl : pour interagir avec les clusters
* VS Code : pour éditer les fichiers de configurations

## Phase de préparation

### Définir l’architecture de l’application et les ressources nécessaire

Le projet consiste à déployer une application web qui se connecte à une base de données. Cette application sera alors conteneurisée grâce à Docker, et Kubernetes qui permettra à orchestrer ces conteneurs.

L’architecture sera basée alors de 2 pods : un qui contient la base de données et l’autre qui contient notre application web.

### Rédiger les Dockerfiles pour conteneuriser l’application web et les scripts de base de données

Pour commencer, un Dockerfile est un fichier de configuration qui permet de définir l’environnement et les étapes pour créer des images Docker, qui contiennent des applications et des services que sont exécuter dans des conteneurs. Pour notre projet, nous devons rédiger 2 Dockerfiles :

* **Le Dockerfile pour l’application web (Node.js) :**

Ce Dockerfile permet de définir les étapes pour installer toutes les dépendances, copier le code et exposer le port utilisé par notre application.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Accompagné de ce Dockerfile, il faut créer un fichier *package. Json*, qui sera le fichier clé du projet *Node.js*. Il contient des informations comme le nom, le service, la version, les dépendances, etc…

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

* **Le Dockerfile pour la base de données (MySQL) :**

Ce Dockerfile permet de mettre en place MySQL et d’inclure le script d’initialisation pour configurer la BDD.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Ce Dockerfile s’accompagne lui aussi d’un fichier *init.sql* :

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

### Configurer l’environnement Kubernetes et Docker

Dans cette étape, nous allons configurer les outils nécessaires pour permettre à Docker et Kubenertes de fonctionner.

Préalablement, installez **Docker Desktop** et **Minikube**.

1. **Configurer Docker** : Pour notre part, nous l’avions déjà installé.
2. **Configurer Minikube** : avec la commande « *minikube start –driver=docker* »

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour vérifier que tout fonctionne correctement, il est possible de faire « *kubectl get nodes* », puis « *minikube status* ».

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. **Configurer *kubectl***:

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Il faut savoir que Minikube configure automatiquement kubectl pour se connecter à un cluster local. En faisant « *kubectl config view* », on peut avoir une vision sur cette configuration.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## Phase de développement

La phase de développement consiste à développer l’application web, créer les images Docker des conteneurs de l’application web et de la BDD, et pour finir, de tester les images Docker pour vérifier si tout fonctionne.

### Développer l’application web

Premièrement, il nous faut une application web en node.js. Notre application web est relativement simple, car sa complexité n’est pas l’élément centrale du projet.

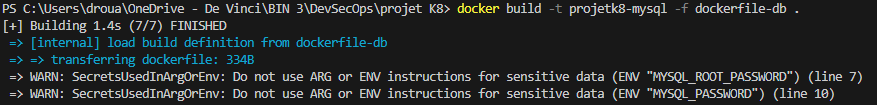
### Créer et tester les images Docker des conteneurs

Maintenant que l’application web est créée, nous devons la conteneuriser ainsi que la base de données MySQL.

Pour ce faire, nous allons utiliser les Dockerfile déjà créés qui vont permettre de copier les fichiers du projet et de les exposer sur le port 3000. Avec la commande « *docker build -t <nom\_image> -f <nom\_dockerfile>* », nous allons créer les images.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement



Pour vérifier que les images ont correctement été créées :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Actuellement, il est possible de se connecter au conteneur qui contient la base de données Projetk8 grâce à la commande :

« *docker exec -it projetk8-mysql mysql -u root -p* ».

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Comme nous pouvons le voir, la base de données projetk8 a été créée mais ne possède pas encore de table.

De plus, le site web s’affiche correctement à l’adresse <http://localhost:4000>. Cependant, l’application n’est pas encore connectée à la base de données car la configuration sera réalisée lors de la phase de déploiement.

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Pour finir la phase de déploiement, nous allons permettre la communication entre les 2 conteneurs. Pour cela, nous allons créer un réseau docker.



Pour la dernière étape de cette phase, il suffit de mettre ces images en ligne sur le docker hub. Pour cela, il faut vous connecter à votre docker hub avec la commande « *docker login* ». Ensuite, vous effecturez la commande « *docher tag <nom\_du\_projet>*

*<identifiant/nom\_image>* ». Pour finir, il restera à push ce que vous venez de créer.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

## Phase de Déploiement

La phase de déploiement consiste à orchestrer les conteneurs de notre projet grâce à Kubernetes.

### Créer les fichiers YAML, pour les pods, Services, ConfigMaps, Secret et PersistentVolumeClaims

Nous allons créer 4 fichiers YAML pour Kubernetes :

1. Sql-deployment.yaml : ce fichier configure la base de données MySQL et inclut le script init.sql
2. Web-deployment.yaml : ce fichier déploie l'application Node.js et configure les variables d'environnement nécessaires.
3. Sql-service.yaml : Ce fichier expose la base de données uniquement en interne dans le cluster.
4. Web-service.yaml : Ce fichier expose l'application Node.js pour un accès externe.

Ici, nous avons rassembler dans 2 dockerfiles les services et les déploiements. Par exemple, le Dockerfile *sql-deployment.yaml* contient le déploiement et le service de MySQL.

De plus, pour rendre notre projet plus sécurisé et rendre les données persistantes, nous avons créé 2 autres fichiers yaml :

1. Mysql-init-configmap.yaml: ce fichier contient l’initialisation de la base de données.
2. My-sql-pvc.yaml : ce fichier permet la persistance des données.
3. Secret.yaml : ce fichier contient les mots de passe hashé

### Déployer les Pods dans Kubernetes et configuration des services

Pour déployer les pods dans Kubernetes, il faut appliquer chaque fichier grâce à la commande « *kubectl apply -f <file\_name>* ».

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Par la suite, pour lancer le service, faites « *minikube service <nom de votre service> »*.

Pour notre part :

Une image contenant texte, logiciel, Logiciel multimédia, Police

Description générée automatiquement

### En cas d’erreurs

Si un déploiement ne démarre pas, n’hésitez pas à rollout le pods en question avec la commande : « *kubectl rollout restart deployment <nom de votre deployment> ».*

Si un pods ne démarre pas, retentez la commande « *kubectl apply -f <nom du pods>*».

Si vous voulez les supprimer, faites « *kubectl delete pod <nom du pod>* ».

## Phase de Validation et Test

Cette phase consiste à vérifier que tous les composants fonctionnent et interagissent ensemble.

### Vérifier que l’application web est accessible depuis l’extérieur du cluster via l'URL ou l’IP du Service.

Sur cette capture, MySQL fonctionne et possède une base de données appelée *projetk8*. Dedans, il existe une table appelée USER.

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

La commande pour démarrer le service :

Une image contenant texte, logiciel, Logiciel multimédia, Police

Description générée automatiquement

Le site fonctionne correctement.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

### Tester les fonctionnalités de l’application web, notamment l’accès aux données.

Comme on peut le voir, nous retrouvons les données envoyées.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Pour avoir plus de vision sur Kubernetes, faites « *minikube dashboard*».

Une image contenant diagramme, capture d’écran, conception

Description générée automatiquement

Une image contenant texte, Police, reçu, capture d’écran

Description générée automatiquement

### Effectuer des tests de résilience en redémarrant les Pods pour s'assurer de la persistance des données.

Pour le test de résilience, redémarrez minikube avec la commande « *minikube start* ». Par la suite, faites « *kubectl rollout restart deployment node-web-app* ». Attendre le redémarrage des pods et lancer le service.

## Phase de Documentation et Livraison

Pour retrouver tous nos livrables, suivre ce lien :

<https://github.com/maureeend/Projet_kubernetes>

Vous y retrouverez :

1. 5 fichiers yaml :

* Mysql-init-configmap.yaml
* Mysql-pvc.yaml
* Sql-deployment.yaml
* Web-deployment.yaml
* App-pcv.yaml
* Secret.yaml

1. L’application web :
   1. Config
      1. Database.js

* Public
  + Index.html
  + Search.html
  + Styles.css
* Server.js

1. Les Dockerfiles :

* Dockerfile-web
* Dockerfile-db
* Init.sql