Déployer Laravel sur Kubernetes

Guide complet pour déployer une application Laravel avec MySQL sur Kubernetes, incluant PersistentVolume, Secrets Docker Registry et Service NodePort

DevOps Laravel Kubernetes 20 min de lecture Niveau Avancé

Document généré le 13/05/2026 à 11h12 · nouvy.fr/wiki/deployer-laravel-kubernetes

Sommaire

31 section(s) · 20 min de lecture

Ce guide complet vous explique comment déployer une application Laravel avec MySQL sur un cluster Kubernetes, en utilisant des PersistentVolumes pour la persistance des données, des Secrets pour l'authentification Docker Registry, et un Service NodePort pour exposer l'application.

Prérequis

- Un cluster Kubernetes fonctionnel
- `kubectl` configuré et connecté au cluster
- Une image Docker Laravel disponible sur GitHub Container Registry (ghcr.io)
- Accès à un dépôt GitHub avec un Personal Access Token (PAT)

Architecture du déploiement

Le déploiement comprend :

- **MySQL** : Base de données avec stockage persistant
- **Laravel** : Application web déployée en plusieurs réplicas
- **Secrets** : Authentification pour accéder aux images privées
- **Services** : Exposition des applications dans le cluster

1. Configuration MySQL avec stockage persistant

PersistentVolume et PersistentVolumeClaim

Créez un fichier `mysql-storage.yaml` :

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /mnt/data/mysql   # chemin sur ta machine hôte
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Copier

Appliquez la configuration :

```
kubectl apply -f mysql-storage.yaml
```

Copier

Vérifiez que le PV et le PVC sont créés :

```
kubectl get pv
kubectl get pvc
```

Copier

Deployment MySQL

Créez un fichier `mysql-deployment.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:8
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: "secret"
            - name: MYSQL_DATABASE
              value: "laravel"
          ports:
            - containerPort: 3306
          volumeMounts:
            - name: mysql-data
              mountPath: /var/lib/mysql
      volumes:
        - name: mysql-data
          persistentVolumeClaim:
            claimName: mysql-pvc
```

Copier

Appliquez le déploiement :

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

Copier

Service MySQL

Créez un fichier `mysql-service.yaml` :

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
    - port: 3306
      targetPort: 3306
  type: ClusterIP
```

Copier

Appliquez le service :

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

Copier

Le service MySQL sera accessible depuis les autres pods via le nom DNS `mysql` sur le port `3306`.

2. Configuration du Secret Docker Registry

Pour accéder à une image Docker privée sur GitHub Container Registry, vous devez créer un Secret Kubernetes.

Créer le Secret

```
kubectl create secret docker-registry ghcr-secret \
  --docker-server=ghcr.io \
  --docker-username=TON_USERNAME_GITHUB \
  --docker-password=TON_TOKEN_PAT \
  --docker-email=TON_EMAIL
```

Copier

Important :

- Remplacez `TON_USERNAME_GITHUB` par votre nom d'utilisateur GitHub
- Remplacez `TON_TOKEN_PAT` par un Personal Access Token GitHub avec les permissions `read:packages`
- Remplacez `TON_EMAIL` par votre email GitHub

Vérifier le Secret

```
kubectl get secret ghcr-secret
```

Copier

3. Déploiement de l'application Laravel

Deployment Laravel

Créez un fichier `laravel-deployment.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: laravel-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: laravel-app
  template:
    metadata:
      labels:
        app: laravel-app
    spec:
      imagePullSecrets:
        - name: ghcr-secret   # secret pour image privée
      containers:
        - name: laravel-app
          image: ghcr.io/nouvy/laravel-app:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          env:
            - name: DB_CONNECTION
              value: "mysql"
            - name: DB_HOST
              value: "mysql"
            - name: DB_PORT
              value: "3306"
            - name: DB_DATABASE
              value: "laravel"
            - name: DB_USERNAME
              value: "root"
            - name: DB_PASSWORD
              value: "secret"
```

Copier

Appliquez le déploiement :

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

Copier

Vérifier le déploiement

```
kubectl get deployments
kubectl get pods -l app=laravel-app
```

Copier

Vous devriez voir 3 pods Laravel en cours d'exécution.

4. Exposition de l'application avec NodePort

Service NodePort

Créez un fichier `laravel-service.yaml` :

```
apiVersion: v1
kind: Service
metadata:
  name: laravel-service
spec:
  selector:
    app: laravel-app
  ports:
    - protocol: TCP
      port: 8080      # port du Service dans le cluster
      targetPort: 80  # port Apache dans le container
      nodePort: 30081 # port exposé sur ton host
  type: NodePort
```

Copier

Appliquez le service :

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

Copier

Vérifier le service

```
kubectl get service laravel-service
```

Copier

Vous devriez obtenir un résultat similaire à :

```
NAME              TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
laravel-service   NodePort   10.96.0.123   <none>        8080:30081/TCP   2m
```

Copier

5. Accéder à l'application

Une fois le service NodePort créé, vous pouvez accéder à votre application Laravel via :

```
http://<IP-NODE>:30081
```

Copier

Pour connaître l'adresse IP d'un nœud :

```
kubectl get nodes -o wide
```

Copier

6. Commandes utiles pour le dépannage

Vérifier les logs des pods Laravel

```
kubectl logs -l app=laravel-app --tail=50
```

Copier

Vérifier les logs d'un pod spécifique

```
kubectl logs <nom-du-pod>
```

Copier

Accéder à un pod Laravel

```
kubectl exec -it <nom-du-pod> -- bash
```

Copier

Vérifier la connexion MySQL depuis un pod Laravel

```
kubectl exec -it <nom-du-pod> -- mysql -h mysql -u root -p
```

Copier

Redémarrer le déploiement

```
kubectl rollout restart deployment/laravel-app
```

Copier

Vérifier le statut du déploiement

```
kubectl rollout status deployment/laravel-app
```

Copier

Mettre à jour l'image Docker

Pour forcer le téléchargement d'une nouvelle image :

```
kubectl set image deployment/laravel-app laravel-app=ghcr.io/nouvy/laravel-app:latest
kubectl rollout restart deployment/laravel-app
```

Copier

7. Points importants

Stockage persistant

- Le `PersistentVolume` utilise `hostPath` pour le développement/test
- En production, utilisez des solutions de stockage distribuées (NFS, EBS, etc.)
- Le `PersistentVolumeClaim` réserve 5Gi d'espace pour MySQL

Sécurité

- **Ne jamais** commiter les mots de passe en clair dans les manifests
- Utilisez des `Secrets` Kubernetes pour les informations sensibles
- En production, utilisez des outils comme `Sealed Secrets` ou `External Secrets Operator`

Scalabilité

- Le déploiement Laravel est configuré avec 3 réplicas pour la haute disponibilité
- Ajustez le nombre de réplicas selon vos besoins :

```
kubectl scale deployment laravel-app --replicas=5
```

Copier

Variables d'environnement

- Les variables d'environnement sont définies directement dans le Deployment
- Pour la production, utilisez des `ConfigMaps` ou des `Secrets` pour gérer la configuration

8. Structure complète des fichiers

Organisez vos fichiers YAML comme suit :

```
kubernetes/
├── mysql-storage.yaml
├── mysql-deployment.yaml
├── mysql-service.yaml
├── laravel-deployment.yaml
└── laravel-service.yaml
```

Copier

Appliquez tous les fichiers en une seule commande :

```
kubectl apply -f kubernetes/
```

Copier

Conclusion

Vous avez maintenant déployé une application Laravel complète sur Kubernetes avec :

- □ MySQL avec stockage persistant
- □ Application Laravel avec plusieurs réplicas
- □ Authentification Docker Registry sécurisée
- □ Exposition de l'application via NodePort

Cette configuration vous permet de déployer et de gérer facilement votre application Laravel sur Kubernetes, avec la possibilité de la mettre à l'échelle selon vos besoins.