

Kubernetes

Kubernetes, aussi appelé k8s, est une plateforme Open Source qui automatise l'exploitation des conteneurs Linux. Elle permet d'éliminer de nombreux processus manuels associés au déploiement et à la mise à l'échelle des applications conteneurisées. Les clusters créés par Kubernetes peuvent couvrir des hôtes situés dans des clouds publics, privés ou hybrides. C'est la raison pour laquelle Kubernetes est la plateforme idéale pour héberger les applications cloud-native qui requièrent une mise à l'échelle rapide.

Kubernetes doit pouvoir s'intégrer aux services de mise en réseau, de stockage, de sécurité, de télémétrie, entre autres, pour fournir une infrastructure de conteneurs complète.

Nous allons déployer un wordpress simple mais fonctionnel avec une base de donnée MySQL et des persistent volumes en utilisant Kubernetes. Nous allons réaliser cet exemple sur un cluster local avec Docker desktop. Cet exemple est à but éducatif seulement et n'est pas destiné à être utilisé en production.

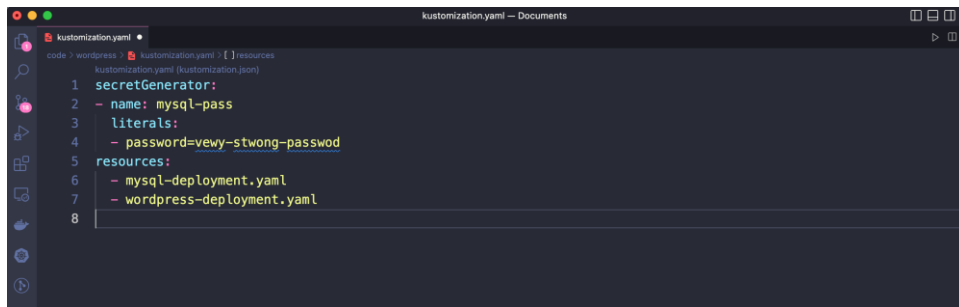
Un cluster Kubernetes est composé d'un Maître (Master node) qui donne des tâches à des Noeuds (worker nodes). Dans un noeud il est possible de déployer des pods dans lequel nous avons un ou plusieurs conteneurs. Tous les conteneurs d'un pod partagent une même adresse IP, un même nom d'hôte et d'autres ressources. Les pods séparent le réseau et le stockage du conteneur sous-jacent, ce qui permet très facilement déplacer les conteneurs au sein d'un cluster.

Pour créer notre wordpress nous allons avoir besoin de deux applications, un wordpress et une base de donnée mysql. Pour fonctionner tous deux auront besoin d'un déploiement (un objet kubernetes qui permet d'indiquer l'image de l'application, le nombre de pods à exécuter et la façon dont ils doivent être mis à jour), un service (qui se chargera d'assurer la connectivité en exposant des ports internes et externes), un secret (objet qui contient des données sensibles comme des mots de passes ou des clés), et enfin un persistent volume claim (qui va nous permettre de monter un Persistent Volume et d'avoir un stockage durable pour nos pods)

Nous allons tout d'abord créer un dossier avec trois fichiers yaml. Les fichiers yaml sont très faciles à lire et à comprendre mais sont également très stricts sur l'indentation. Il faut donc être très rigoureux sur ce point.

```
-rw-r--r-- 1 cmaroy staff 138B Jun 12 14:12 kustomization.yaml
-rw-r--r-- 1 cmaroy staff 1.2K Jun 12 13:56 mysql-deployment.yaml
-rw-r--r-- 1 cmaroy staff 1.2K Jun 12 20:04 wordpress-deployment.yaml
```

Le fichier kustomization.yaml va nous permettre de déployer tous les éléments dans le dossier en une seule commande. Nous allons donc placer les ressources (les fichiers yaml qui seront déployés) ainsi qu'un générateur de mots de passe (secrets) dans ce fichier.

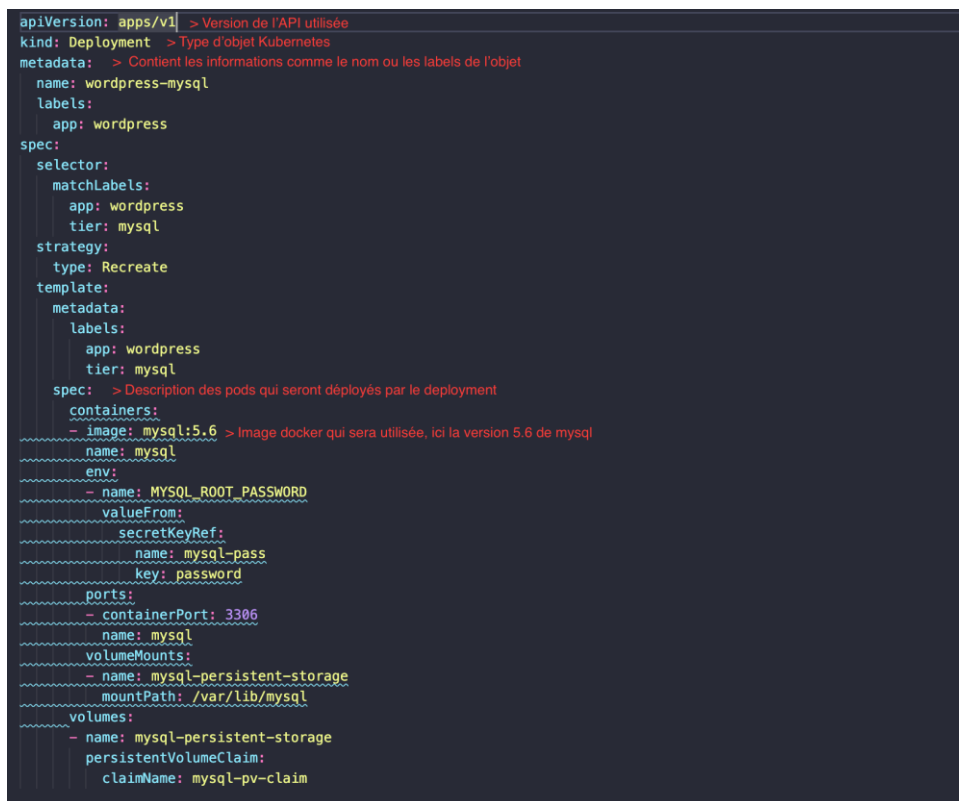


```
kustomization.yaml
1 secretGenerator:
2   - name: mysql-pass
3     literals:
4       - password=vewy-stwong-passwod
5 resources:
6   - mysql-deployment.yaml
7   - wordpress-deployment.yaml
8
```

Il est possible de déployer plusieurs objets Kubernetes dans un seul et même fichier yaml. Pour cela il faut les séparer de ---

Le premier objet que nous allons voir est le déploiement. Voici le déploiement de la base de données mysql. Il n'est pas nécessaire de connaître tous les éléments par coeur mais plutôt de comprendre comment la déclaration se découpe.

Voici le déploiement de la base de données mySQL.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-pass
                  key: password
          ports:
            - containerPort: 3306
          name: mysql
          volumeMounts:
            - name: mysql-persistent-storage
              mountPath: /var/lib/mysql
          volumes:
            - name: mysql-persistent-storage
              persistentVolumeClaim:
                claimName: mysql-pv-claim
```

Voici le service de l'application Wordpress.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
        - image: wordpress:4.8-apache > image wordpress utilisée
          name: wordpress
          env:
            - name: WORDPRESS_DB_HOST
              value: wordpress-mysql
            - name: WORDPRESS_DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-pass
                  key: password
          ports:
            - containerPort: 80 > Port du conteneur qui sera exposé
              name: wordpress
          volumeMounts:
            - name: wordpress-persistent-storage
              mountPath: /var/www/html
          volumes:
            - name: wordpress-persistent-storage
              persistentVolumeClaim:
                claimName: wp-pv-claim

```

Nous allons maintenant rédiger le YAML pour le service de la base de donnée MySQL. Le service permettra d'assurer la connectivité de l'application.

```

1  apiVersion: v1
2  kind: Service > Nous indiquons que l'objet est un service
3  metadata:
4    name: wordpress-mysql
5    labels:
6      app: wordpress
7  spec:
8    ports:
9      - port: 3306
10   selector:
11     app: wordpress
12     tier: mysql
13   clusterIP: None

```

Nous pouvons maintenant créer le service de l'application Wordpress. Nous allons exposer le port HTTP (80)

```

apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None

```

Pour fonctionner notre application aura besoin d'un stockage persistant. Nous allons donc ajouter un persistent Volume Claim. Nous allons indiquer un mode d'accès et la quantité de ressources que nous souhaitons pour notre base de donnée MySQL.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi

```

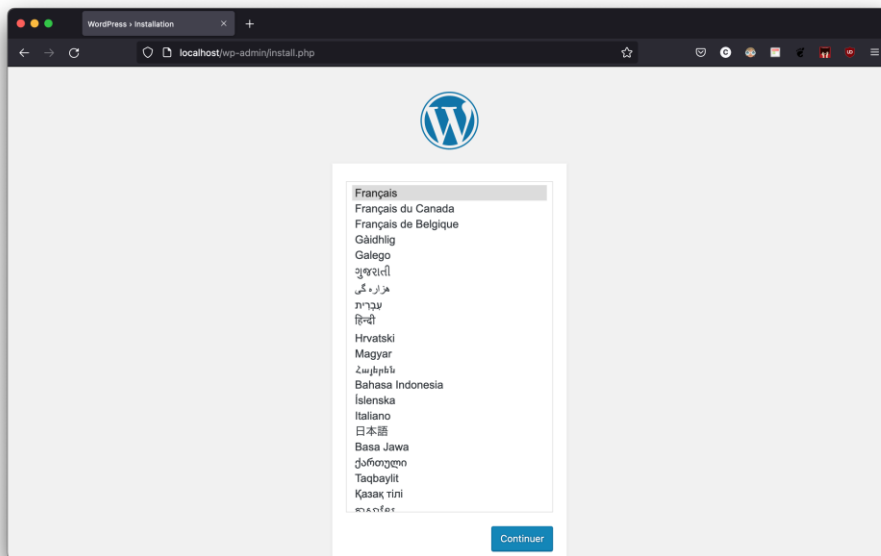
Nous allons réitérer le même processus pour notre application Wordpress.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

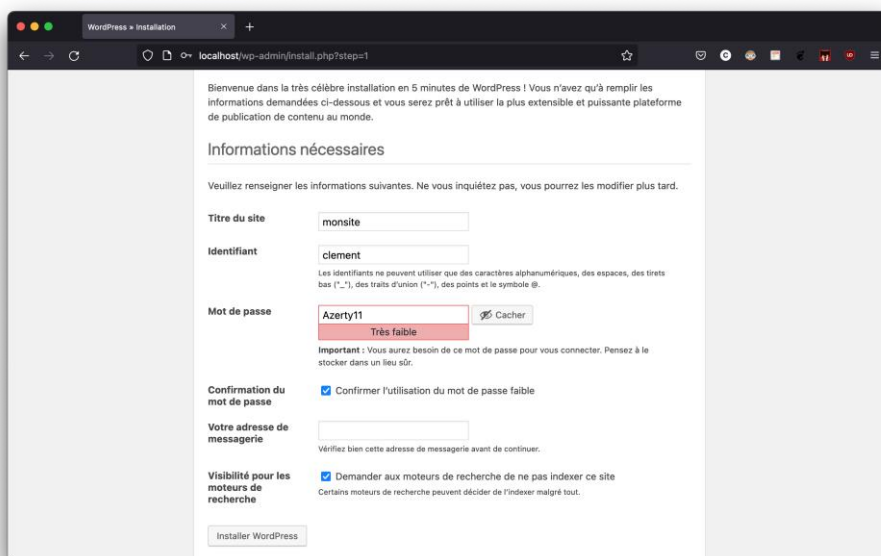
Nous avons tous les éléments dont nous avons besoin. Nous pouvons maintenant lancer notre application Kubernetes en une seule commande grâce au fichier kustomization et au paramètre `kubectl apply -k`.

```
→ wordpress kubectl apply -k ./
secret/mysql-pass-52hk5gm5th created
service/wordpress created
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
→ wordpress |
```

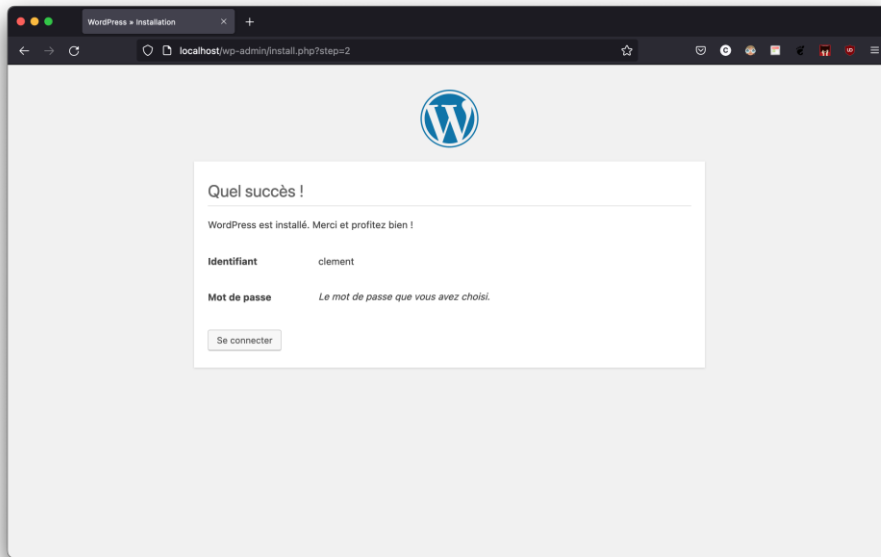
Nous allons maintenant vérifier le bon fonctionnement de notre application en ouvrant une page web et en nous connectant sur localhost, 127.0.0.1 ou l'ip local de notre machine.



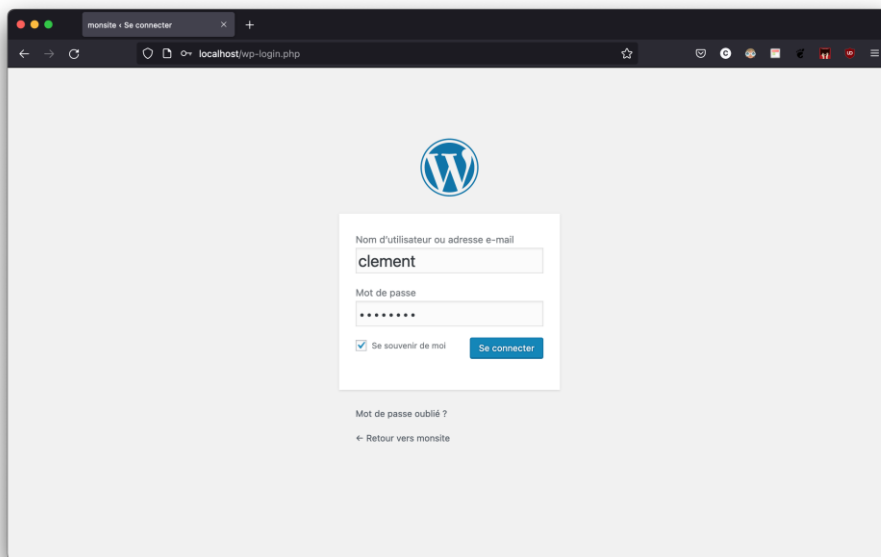
Nous sommes bien arrivés sur la page d'installation wordpress, nous pouvons cliquer sur continuer. Et procéder à l'installation de notre wordpress.

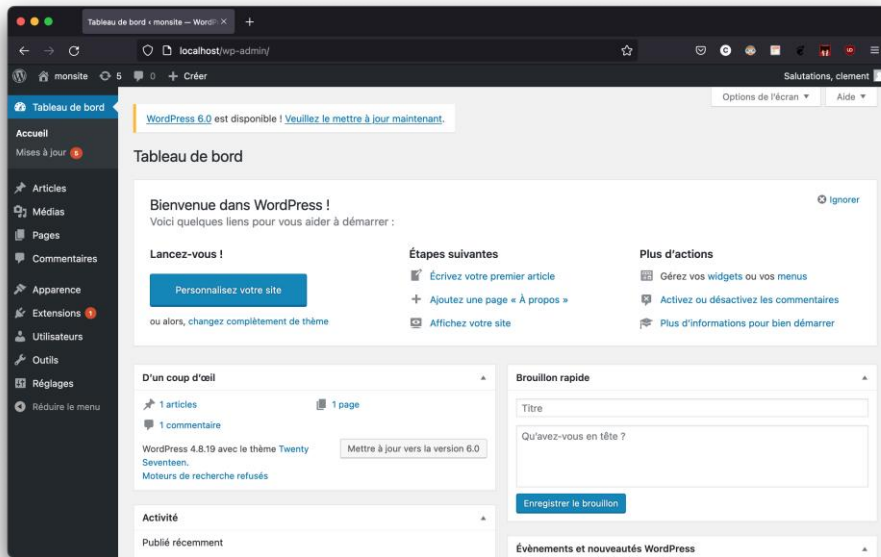


Après avoir rempli les informations, nous pouvons cliquer sur “Installer WordPress”



WordPress est installé, nous pouvons nous connecter à l'aide de l'identifiant et du mot de passe choisi.





Notre WordPress est fonctionnel. Nous pouvons supprimer toute notre installation en une seule commande **kubectl delete -k ./** grâce à notre fichier kustomization.

```
→ wordpress kubectl delete -k ./

secret "mysql-pass-52hk5gm5th" deleted
service "wordpress" deleted
service "wordpress-mysql" deleted
persistentvolumeclaim "mysql-pv-claim" deleted
persistentvolumeclaim "wp-pv-claim" deleted
deployment.apps "wordpress" deleted
deployment.apps "wordpress-mysql" deleted
→ wordpress kubectl get pods
No resources found in default namespace.
→ wordpress |
```

Tous nos objets ont bien été supprimés.