Introduction à Kubernetes

Kubernetes (K8s) est une plateforme open-source pour **orchestrer** des conteneurs, automatisant le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Il résout les défis de gestion de multiples conteneurs dans des environnements complexes, en assurant haute disponibilité, scalabilité et résilience.

- **Rôle principal** : Gérer des conteneurs (ex. : Docker) à grande échelle sur un cluster de machines, en coordonnant leur exécution, réseau et stockage.
- **Composants clés** :
    - **Pods** : Unités de base contenant un ou plusieurs conteneurs partageant réseau et stockage.
    - **Services** : Fournissent une adresse stable et un équilibrage de charge pour accéder aux Pods.
    - **Nodes** : Machines du cluster (Worker Nodes exécutent les Pods ; Master Node gère le cluster).
    - **Deployments** : Gèrent la mise à l'échelle et les mises à jour des Pods.
    - **Ingress** : Gère l'accès HTTP externe (ex. : routage par domaine).
    - **ConfigMaps/Secrets** : Stockent configurations et données sensibles.
- **Avantages** :
    - **Automatisation** : Déploiement, redémarrage après crash, équilibrage de charge.
    - **Scalabilité** : Ajout/suppression dynamique de conteneurs selon la charge.
    - **Portabilité** : Compatible avec cloud, on-premise ou hybride.
- **Outils associés** :
    - **kubectl** : CLI pour interagir avec le cluster.
    - **Minikube** : Cluster local pour tests.
    - **cAdvisor/Prometheus** : Monitoring des performances.

Exemple : Application web MERN (MongoDB, Express.js, React, Node.js)

Une application **MERN** est un cas idéal pour Kubernetes, car elle combine plusieurs services (frontend, backend, base de données) nécessitant orchestration.

Cas d'usage

Imaginons une application MERN : un site e-commerce avec :

- **Frontend** : Interface React pour afficher les produits.
- **Backend** : API Express.js/Node.js pour gérer les commandes.
- **Base de données** : MongoDB pour stocker produits et utilisateurs.

Comment Kubernetes gère cette application

1. **Pods** :
    - Un Pod pour le **frontend React** (conteneur avec l'app buildée, servie via Nginx).

    - Un Pod pour le **backend Express.js** (conteneur Node.js exécutant l'API).
    - Un Pod pour **MongoDB** (conteneur avec l'image officielle MongoDB).
    - Exemple : Chaque Pod peut contenir un conteneur principal et un conteneur de monitoring (ex. : cAdvisor).
2. **Deployments** :
    - Un **Deployment** pour le frontend (ex. : 3 Pods pour gérer le trafic).
    - Un Deployment pour le backend (ex. : 5 Pods pour l'API sous forte charge).
    - Un Deployment pour MongoDB (ex. : 1 Pod avec persistance via un volume).
    - Commande : kubectl create deployment mern-frontend --image=mern-frontend:latest.
3. **Services** :
    - Un **Service ClusterIP** pour le backend, permettant au frontend de communiquer avec l'API (ex. : backend-service:3000).
    - Un **Service NodePort** ou **LoadBalancer** pour exposer le frontend aux utilisateurs (ex. : port 30080).
    - Un Service pour MongoDB, accessible uniquement par le backend.
    - Commande : kubectl expose deployment mern-backend --port=3000.
4. **Ingress** :
    - Un **Ingress** route le trafic HTTP externe (ex. : shop.example.com vers le frontend, shop.example.com/api vers le backend).
    - Exemple : Configuré avec un contrôleur Ingress comme Nginx.
5. **ConfigMaps/Secrets** :
    - Un **ConfigMap** pour stocker les variables comme l'URL de MongoDB (ex. : mongodb://mongo-service:27017).
    - Un **Secret** pour les identifiants de la base (ex. : utilisateur/mot de passe MongoDB).
6. **Nodes et Master Node** :
    - Les Pods sont répartis sur des **Worker Nodes** (ex. : 3 nœuds pour équilibrer la charge).
    - Le **Master Node** gère le cluster (Scheduler place les Pods, API Server traite les commandes kubectl).
7. **Monitoring** :
    - **cAdvisor** (intégré au Kubelet) collecte les métriques (CPU/mémoire) des Pods MERN.
    - **Prometheus** stocke ces métriques, et un **Kubernetes Dashboard** affiche les performances.

Exemple de déploiement

```
# Déployer le backend
kubectl create deployment mern-backend --image=mern-backend:latest
kubectl scale deployment mern-backend --replicas=5
kubectl expose deployment mern-backend --port=3000

# Déployer le frontend
kubectl create deployment mern-frontend --image=mern-frontend:latest
kubectl expose deployment mern-frontend --type=LoadBalancer --port=80
```

```
# Vérifier
kubectl get pods,services
```

Bénéfices pour l'application MERN

- **Scalabilité** : Si le trafic augmente, Kubernetes ajoute des Pods frontend/backend.
- **Résilience** : En cas de crash d'un Pod MongoDB, Kubernetes le redémarre automatiquement.
- **Mises à jour** : Déploie une nouvelle version de l'API Express.js sans interruption (rolling update).
- **Portabilité** : Teste localement avec Minikube, puis déploie sur AWS/GCP.

Résumé

Kubernetes orchestre des conteneurs pour automatiser déploiement, scalabilité et gestion. Pour une app **MERN**, il gère des Pods (React, Express.js, MongoDB), Services (communication/accès externe), Deployments (mise à l'échelle), et Ingress (routage HTTP), avec un Master Node contrôlant les Worker Nodes. Exemple : Déploie 5 Pods backend Express.js et expose le frontend React via un LoadBalancer, avec cAdvisor pour surveiller les ressources.

Relation entre K8s et DevOps

Kubernetes est une plateforme d'orchestration de conteneurs qui automatise des tâches clés alignées avec DevOps :

- **Déploiement rapide** : K8s déploie des conteneurs (ex. : une app MERN) via des **Deployments**, avec des mises à jour sans interruption (rolling updates).
- **Automatisation** : Gère la scalabilité (ajout de Pods selon la charge), la reprise après panne (redémarrage des Pods), et l'équilibrage de charge (Services).
- **CI/CD** : Intègre avec des outils DevOps comme Jenkins ou GitLab CI pour automatiser les pipelines (build, test, déploiement). Ex. : Déployer une nouvelle version d'une API Express.js après un push Git.
- **Infrastructure as Code (IaC)** : Les ressources K8s (Pods, Services, Ingress) sont définies dans des fichiers YAML, gérés avec Git, alignés avec les pratiques IaC de DevOps.
- **Monitoring/Observabilité** : K8s fournit des métriques via cAdvisor/Prometheus, essentielles pour le suivi des performances, un pilier DevOps.
- **Collaboration** : Centralise la gestion des applications, facilitant le travail des équipes Dev (développement) et Ops (infrastructure).

Exemple avec une app MERN

Dans un pipeline DevOps pour une application MERN :

- **CI** : Un push Git déclenche Jenkins pour construire des images Docker (React, Node.js) et les pousser vers un registre.
- **CD** : Kubernetes déploie ces images comme Pods via kubectl apply -f deployment.yaml, met à jour l'API Node.js sans downtime, et expose le frontend via un Ingress.
- **Monitoring** : Prometheus surveille les Pods, alertant l'équipe DevOps si le backend consomme trop de CPU.

Résumé

Kubernetes soutient **DevOps** en automatisant le déploiement, la scalabilité et la gestion des conteneurs, alignés avec les principes d'automatisation, CI/CD, IaC et observabilité. Pour une app MERN, K8s accélère les déploiements, intègre les pipelines CI/CD, et centralise le monitoring, renforçant la collaboration Dev/Ops.