

Minikube

Docker Compose vs. Kubernetes (K8s)

Caractéristique	Docker Compose	Kubernetes (K8s)
Objectif Principal	Définir et exécuter des applications multi-conteneurs.	Orchestration et gestion des applications conteneurisées à grande échelle.
Environnement	Développement local, tests unitaires, petits projets sur un seul hôte .	Production distribuée, clusters de serveurs, environnements cloud .
Mise à l'Échelle/HA	Limité (manuelle ou nécessite des outils externes comme Swarm).	Automatique (auto-scaling, self-healing, haute disponibilité native).

Kubernetes (K8s) : Introduction

Kubernetes est un système open-source conçu pour **automatiser le déploiement, la mise à l'échelle et la gestion** des applications conteneurisées. Il fournit une plateforme robuste pour gérer des charges de travail distribuées sur un cluster de machines.

Composants Fondamentaux de Kubernetes

1. Les Pods

Le **Pod** est la plus petite unité déployable dans Kubernetes.

- **Unité de Colocation** : Un Pod contient un ou plusieurs conteneurs qui partagent les mêmes ressources réseau (adresse IP et ports) et le stockage.
- **Adresse IP Unique** : Chaque Pod reçoit une adresse IP unique au sein du cluster.
- **Nature Éphémère** : Les Pods sont considérés comme **éphémères**. Lorsqu'un Pod meurt ou est remplacé, son ancienne adresse IP est perdue, et un nouveau Pod reçoit une nouvelle adresse. Il est essentiel de ne jamais se fier directement à l'adresse IP d'un Pod.

2. Les Services

Les **Services** résolvent le problème de l'éphémérité des Pods en fournissant un point d'accès stable.

- **Rôle :** Assurer une connectivité stable et un équilibrage de charge pour un groupe de Pods.
- **Fixation de l'Adresse IP :** Le Service se voit attribuer une **adresse IP et un nom DNS stables** qui ne changent pas, même lorsque les Pods sous-jacents sont remplacés.
- **Types de Services pour la connectivité :**
 - **ClusterIP (défaut)** : IP interne, accessible uniquement au sein du cluster.
 - **NodePort** : Expose le service sur un port statique sur chaque Nœud de travail.
 - **LoadBalancer** : Expose le service via l'équilibrEUR de charge d'un fournisseur cloud, fournissant une adresse IP publique stable.

3. Ingress : Le Routeur Externe

Ingress gère l'accès externe aux Services du cluster, principalement pour le trafic **HTTP/HTTPS**.

- **Rôle :** Agir comme un routeur de couche 7 centralisé.
- **Fonctionnalités :**
 - Routage du trafic basé sur le **nom d'hôte** (api.example.com).
 - Routage basé sur le **chemin d'accès** (example.com/users).
 - Gestion de la terminaison SSL/TLS.

Architecture du Cluster

Un cluster Kubernetes est divisé en deux rôles de machines : le **Control Plane** et les **Worker Nodes**.

A. Control Plane (Plan de Contrôle)

C'est le **cerveau** du cluster, chargé de maintenir l'état désiré.

- **Rôle :** Prendre toutes les décisions d'orchestration (planification, mise à l'échelle, etc.) et maintenir l'état du cluster.
- **Composants clés :**

- **API Server** : Point d'entrée pour toutes les communications.
- **etcd** : Base de données qui stocke l'état complet de la configuration du cluster.
- **Scheduler** : Choisit sur quel Worker Node lancer un nouveau Pod.

B. Worker Node (Nœud de Travail)

Ce sont les machines qui **exécutent** les charges de travail conteneurisées.

- **Rôle** : Exécuter les Pods.
- **Composants clés** :
 - **Kubelet** : Agent qui s'assure que les Pods sur ce Nœud sont en cours d'exécution.
 - **Kube-proxy** : Gère les règles réseau sur le Nœud pour permettre la communication avec les Pods et les Services.
 - **Container Runtime** : Logiciel qui exécute les conteneurs (ex: Docker, Containerd).

Minikube

Minikube est un outil de développement et d'apprentissage qui permet de faire tourner un cluster Kubernetes **localement** sur votre machine. Il crée un cluster à **nœud unique** (Control Plane et Worker Node combinés) dans une machine virtuelle pour faciliter le développement et les tests.

- Démarrage d'un cluster Kubernetes local à l'aide de la commande **minikube start**. Cette étape configure un nœud unique, télécharge la version Kubernetes v1.34.0 et configure l'outil de ligne de commande kubectl pour l'accès local.

```
C:\Users\chayma>minikube start
* minikube v1.37.0 sur Microsoft Windows 11 Pro 10.0.26200.7309 Build 26200.7309
* Choix automatique du pilote docker. Autres choix: virtualbox, ssh
* Utilisation du pilote Docker Desktop avec le privilège root
* Démarrage du nœud "minikube" primary control-plane dans le cluster "minikube"
* Extraction de l'image de base v0.0.48...
* Téléchargement du préchargement de Kubernetes v1.34.0...
  > preloaded-images-k8s-v18-v1...: 337.07 MiB / 337.07 MiB 100.00% 1.79 Mi
  > gcr.io/k8s-minikube/kicbase...: 488.52 MiB / 488.52 MiB 100.00% 1.85 Mi
* Création de docker container (CPU=2, Memory=5000Mo) ...
! Échec de la connexion à https://registry.k8s.io/ depuis l'intérieur du minikube container
* Pour extraire de nouvelles images externes, vous devrez peut-être configurer un proxy : https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Préparation de Kubernetes v1.34.0 sur Docker 28.4.0...
* Configuration de bridge CNI (Container Networking Interface)...
* Vérification des composants Kubernetes...
  - Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
* Modules activés: default-storageclass, storage-provisioner

! C:\WINDOWS\system32\kubectl.exe est la version 1.30.0, qui peut comporter des incompatibilités avec Kubernetes 1.34.0.
- Vous voulez kubectl v1.34.0 ? Essayez 'minikube kubectl -- get pods -A'
* Terminé ! kubectl est maintenant configuré pour utiliser "minikube" cluster et espace de noms "default" par défaut.

C:\Users\chayma>
```

Cette image documente le **démarrage initial du cluster Minikube**, confirmant l'utilisation du pilote Docker et la configuration des ressources pour la version Kubernetes v1.34.0.

```
C:\Users\chayma>minikube version
minikube version: v1.37.0
commit: 65318f4cff9c12cc87ec9eb8f4cdd57b25047f3

C:\Users\chayma>kubectl version --client
Client Version: v1.30.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
```

Ces captures illustrent la vérification des versions de Minikube et kubectl, l'état du nœud de contrôle, et la **création initiale du déploiement hello-nginx**

```
C:\Users\chayma>kubectl get nodes
NAME      STATUS    ROLES      AGE      VERSION
minikube  Ready     control-plane  4m44s   v1.34.0
```

```
C:\Users\chayma>kubectl create deployment hello-nginx --image=nginx
deployment.apps/hello-nginx created
```

```
C:\Users\chayma>kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
hello-nginx-fbcf4796f-4kb8c  1/1     Running   0          35s
```

- Cette séquence de commandes illustre la mise à l'échelle du déploiement Nginx à **quatre répliques** et son **exposition via un Service NodePort** pour la connectivité externe

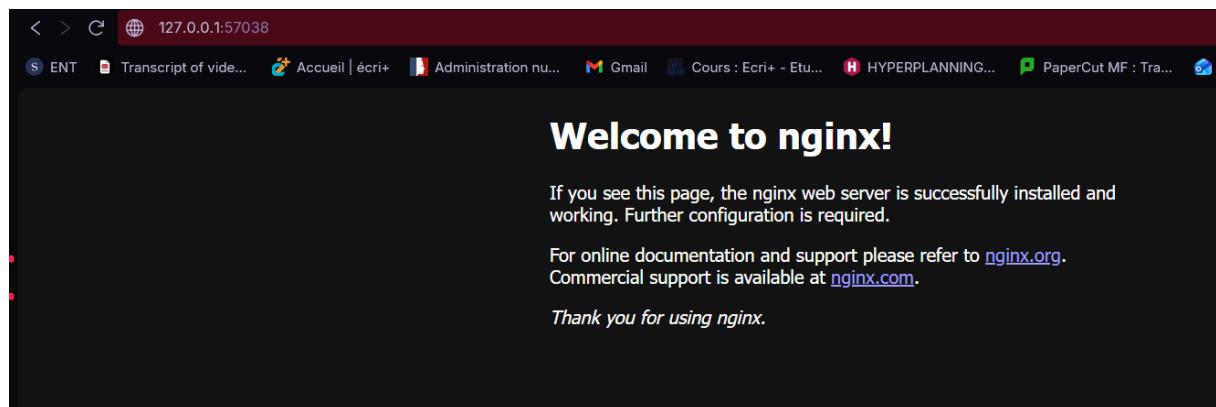
```
C:\Users\chayma>kubectl scale deployment hello-nginx --replicas=4
deployment.apps/hello-nginx scaled

C:\Users\chayma>kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
hello-nginx-fbcf4796f-4kb8c      1/1     Running   0          5m58s
hello-nginx-fbcf4796f-8899s      1/1     Running   0          17s
hello-nginx-fbcf4796f-kh9hw      1/1     Running   0          17s
hello-nginx-fbcf4796f-rqkgx      1/1     Running   0          17s
```

```
C:\Users\chayma>kubectl expose deployment hello-nginx --type=NodePort --port=80
service/hello-nginx exposed
```

Cette image confirme le succès du déploiement en utilisant la commande **minikube service --url** pour accéder au Service NodePort exposé, ce qui affiche la page d'accueil de **Nginx** dans le navigateur

```
C:\Users\chayma>minikube service hello-nginx --url
http://127.0.0.1:57038
! Comme vous utilisez un pilote Docker sur windows, le terminal doit être ouvert pour l'exécuter.
```



Cette capture montre la commande **kubectl delete** utilisée pour supprimer les ressources de l'application – le Service, puis le Déploiement – ce qui est confirmé par l'absence de Pods restants dans l'espace de noms par défaut.

```
C:\Users\chayma>kubectl delete service hello-nginx
service "hello-nginx" deleted

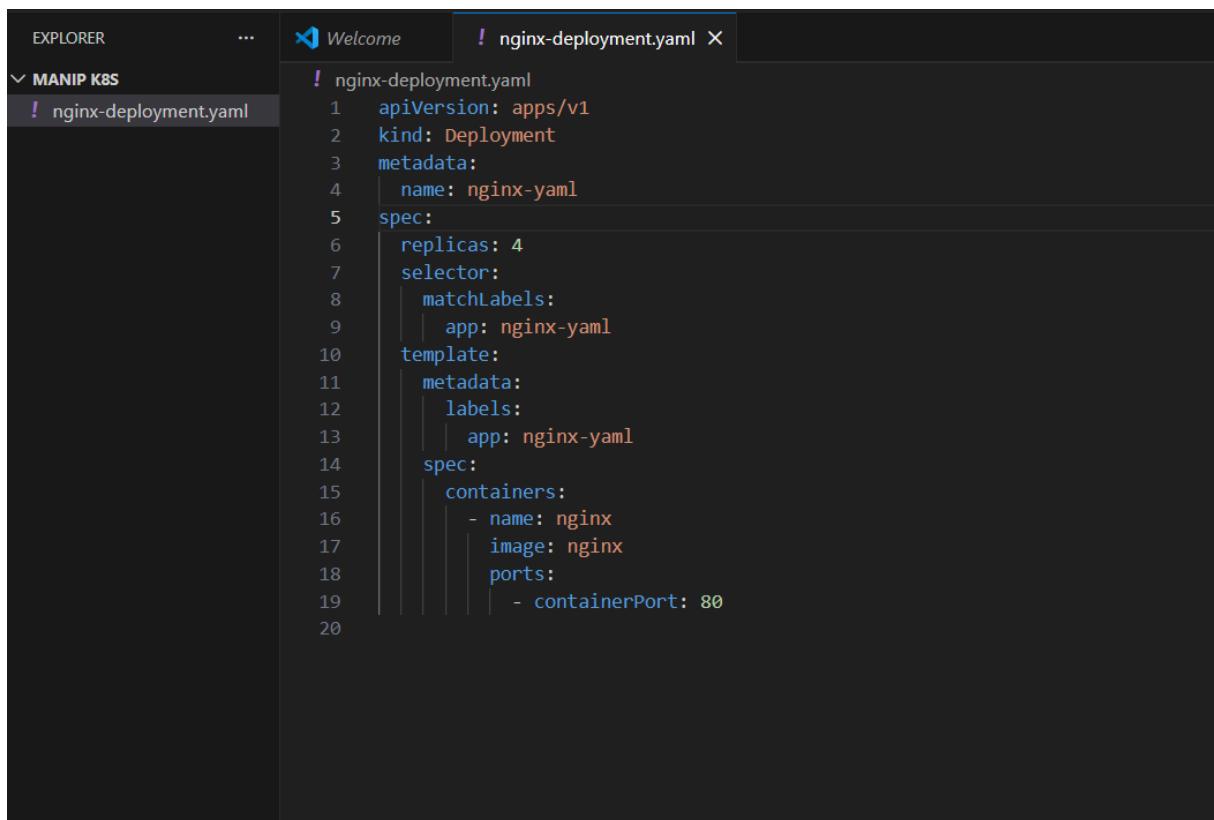
C:\Users\chayma>kubectl delete deployment hello-nginx
deployment.apps "hello-nginx" deleted
```

```
C:\Users\chayma>kubectl delete service hello-nginx
service "hello-nginx" deleted

C:\Users\chayma>kubectl delete deployment hello-nginx
deployment.apps "hello-nginx" deleted

C:\Users\chayma>kubectl get pods
No resources found in default namespace.
```

- Cette capture montre la définition déclarative du déploiement Nginx dans un fichier **YAML** et son application via la commande **kubectl apply**



The screenshot shows a code editor interface with a dark theme. On the left, there's an 'EXPLORER' sidebar with a tree view showing a folder named 'MANIP K8S' containing a file named 'nginx-deployment.yaml'. The main area is titled 'Welcome' and contains the content of the 'nginx-deployment.yaml' file. The file is a YAML configuration for a Deployment:

```
! nginx-deployment.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-yaml
5 spec:
6   replicas: 4
7   selector:
8     matchLabels:
9       app: nginx-yaml
10    template:
11      metadata:
12        labels:
13          app: nginx-yaml
14      spec:
15        containers:
16          - name: nginx
17            image: nginx
18            ports:
19              - containerPort: 80
20
```

```
PS C:\Users\chayma\Desktop\ing3\procedure de test\manip k8s> kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-yaml created
PS C:\Users\chayma\Desktop\ing3\procedure de test\manip k8s> []
```

- Cette capture démontre l'éphémérité des Pods en listant les quatre répliques, puis en **supprimant manuellement un Pod** pour montrer qu'il n'est plus trouvé.

```
PS C:\Users\chayma\Desktop\ing3\procedure de test\manip k8s> kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
nginx-yaml-5476995d8f-2wzxp  1/1     Running   0          19s
nginx-yaml-5476995d8f-ft95q  1/1     Running   0          19s
nginx-yaml-5476995d8f-jzjn5  1/1     Running   0          19s
nginx-yaml-5476995d8f-pvzfw  1/1     Running   0          19s
PS C:\Users\chayma\Desktop\ing3\procedure de test\manip k8s> █
```

```
C:\Users\chayma>kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
nginx-yaml-5476995d8f-2wzxp  1/1     Running   1 (3m37s ago)  4m29s
nginx-yaml-5476995d8f-ft95q  1/1     Running   1 (3m37s ago)  4m29s
nginx-yaml-5476995d8f-jzjn5  1/1     Running   1 (3m37s ago)  4m29s
nginx-yaml-5476995d8f-pvzfw  1/1     Running   1 (3m37s ago)  4m29s

C:\Users\chayma>kubectl delete pod nginx-yaml-5476995d8f-2wzxp
pod "nginx-yaml-5476995d8f-2wzxp" deleted

C:\Users\chayma>kubectl get pod nginx-yaml-5476995d8f-2wzxp
Error from server (NotFound): pods "nginx-yaml-5476995d8f-2wzxp" not found
```

La commande **minikube stop** est utilisée pour arrêter le nœud et mettre hors tension le profil du cluster, ce qui conclut la session de travail

```
NO resources found in default namespace.

C:\Users\chayma>minikube stop
* Nœud d'arrêt "minikube" ...
* Mise hors tension du profil "minikube" via SSH...
* 1 nœud arrêté.

C:\Users\chayma>|
```