Fode Mangane

Kubernetes

Guide Complet de Kubernetes pour Débutants et Intermédiaires

Présentateur
Fode Mangane

Date
16 Mai 2025

https://fodemangane-portfolio.netlify.app/ | +221 787388292 | fodemangane@gmail.com | Dakar, Senegal

Introduction

Kubernetes (souvent abrégé K8s) est une plateforme open-source d'orchestration de conteneurs qui automatise le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Développé à l'origine par Google, Kubernetes est maintenant maintenu par la Cloud Native Computing Foundation.
Ce guide vous accompagnera pas à pas dans la découverte et la maîtrise de Kubernetes, des concepts fondamentaux aux configurations plus avancées.

Pourquoi Kubernetes?

Avant de plonger dans les détails techniques, il est essentiel de prendre un temps d'analyse pour comprendre les raisons profondes qui ont fait de Kubernetes un pilier incontournable dans le paysage technologique moderne. Face à l'essor des applications distribuées, à la nécessité d'agilité, de flexibilité et d'évolutivité, Kubernetes s'impose comme la solution privilégiée pour automatiser, orchestrer et simplifier la gestion des environnements conteneurisés à grande échelle.

Déploiements automatisés

Rollouts et rollbacks sans temps d'arrêt

Portabilité

Déployez vos applications de manière cohérente sur différentes infrastructures (cloud public, privé, hybride)

Gestion multi-cloud

Possibilité de gérer des clusters répartis sur plusieurs fournisseurs cloud

Mise à l'échelle

Adaptez automatiquement les ressources en fonction de la demande

Concepts Fondamentaux

Avant Kubernetes, il est important de comprendre la différence entre les conteneurs et les machines virtuelles (VM) :

Les conteneurs sont plus légers, démarrent plus rapidement et utilisent moins de ressources que les VM, ce qui les rend idéaux pour les architectures microservices.

Conteneurs vs. Machines Virtuelles

- **Machine Virtuelle**
  Émule un système d'exploitation complet, y compris le noyau
- **Conteneur**
  Partage le noyau du système d'exploitation hôte, mais isole les processus applicatifs

Architecture de Kubernetes

Kubernetes repose sur une architecture distribuée de type maître-nœud (master-node), organisée en deux parties principales : le plan de contrôle, qui gère et pilote l'ensemble du cluster, et les nœuds de travail, qui exécutent les conteneurs et assurent le fonctionnement des applications.

1. Plan de contrôle (Gère le cluster)

API Server

Point d'entrée pour toutes les commandes REST

etcd

Base de données clé-valeur pour stocker l'état du cluster

Scheduler

Attribue des nœuds aux nouveaux pods

Controller Manager

Régule l'état du cluster

2. Nœuds (Nodes) - Exécutent les charges de travail

Kubelet

Agent qui s'assure que les conteneurs fonctionnent dans un pod

Kube-proxy

Gère le réseau et les règles de communication

Container Runtime

Logiciel qui exécute les conteneurs (Docker, containerd, etc.)

Les Objets Kubernetes Essentiels

Pod

Le Pod est l'unité la plus petite et la plus simple dans le modèle d'objets Kubernetes. Il représente un ou plusieurs conteneurs qui partagent des ressources réseau et stockage.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: Fode-Mangane-premier-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.19
    ports:
    - containerPort: 80
```

Deployment

Le Deployment est un objet Kubernetes qui permet de gérer le déploiement, la mise à jour et la gestion des Pods de manière déclarative. Il garantit que le nombre souhaité de Pods est toujours disponible et facilite les mises à jour progressives sans interruption de service.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: Fode-application
spec:
  replicas: 3
  selector:
    matchLabels:
      app: Fode-application
  template:
    metadata:
      labels:
        app: Fode-application
    spec:
      containers:
      - name: Fode-application
        image: mon-image:v1.0
        ports:
        - containerPort: 8080
```

Service

Le Service est une abstraction dans Kubernetes qui définit un ensemble logique de Pods et permet de gérer leur accès réseau. Il assure une communication stable et permanente vers les Pods, même lorsque ceux-ci sont recréés ou déplacés, en appliquant une politique d'accès adaptée.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: Fode-service
spec:
  selector:
    app: Fode-application
  ports:
  - port: 80
    targetPort: 8080
  type: ClusterIP  # ou aussi le NodePort ou LoadBalancer
```

ConfigMap et Secret

Ces objets permettent de séparer la configuration de l'application de son code :

- ConfigMap : Stocke des données de configuration non sensibles
- Secret : Stocke des données sensibles comme les mots de passe et les tokens

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: Fode-config
data:
  DATABASE_URL: "mongodb://localhost:27017"
  LOG_LEVEL: "info"
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: Fode-secrets
type: Opaque
data:
  username: YWRtaW4=  # "admin" encodé en Base64
  password: MWYyZDFlMmU2N2Rm  # Mot de passe encodé en Base64
```

Mise en Pratique : Premiers Pas avec Kubernetes

- **Installation d'un environnement de développement**

Pour commencer à utiliser Kubernetes localement, plusieurs options s'offrent à vous :

Minikube

Crée un cluster Kubernetes local sur une seule machine

1. Télécharge le binaire Minikube
2. Installe-le
3. Et démarre ton cluster

Kubectl

Kubectl est l'interface en ligne de commande pour interagir avec votre cluster Kubernetes.

```
fode@serveru:~$ kubectl create deployment apache-deply --image=httpd
deployment.apps/apache-deply created
fode@serveru:~$ kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
apache-deply   1/1     1            1           102s
fode@serveru:~$ kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
apache-deply-77dc866b94-c5h6g   1/1     Running   0          3m10s
fode@serveru:~$ kubectl get replicaset
NAME                      DESIRED   CURRENT   READY   AGE
apache-deply-77dc866b94   1         1         1       5m51s
```

Kind (Kubernetes IN Docker)

Exécute Kubernetes dans des conteneurs Docker

K3d

Version légère de K3s (Kubernetes minimaliste) qui fonctionne dans Docker

Grâce à ces différentes solutions d'installation, il est désormais possible de mettre en place rapidement un environnement Kubernetes local pour s'entraîner, tester et déployer ses premières applications en conteneur.

Que ce soit via Minikube, Kind, K3d ou en utilisant Kubectl, chaque outil offre ses propres avantages selon le contexte et les besoins.

Ces premiers pas permettent de se familiariser avec le fonctionnement de Kubernetes avant de passer à des déploiements en environnement de production plus complexes.

Déploiement d'une Application Simple

Après avoir installé notre environnement Kubernetes local, il est temps de passer à la pratique en déployant une première application.

Dans cet exemple, nous allons mettre en place un déploiement d'une application web simple basée sur Nginx et l'exposer via un Service Kubernetes.

Ce cas concret permet de comprendre le cycle de déploiement d'une application conteneurisée sur un cluster et d'appréhender les notions de Deployment, Pods et Service.

Étape 1 : Créer un Deployment

- **Créez un fichier webapp-deployment.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: nginx:1.19
        ports:
        - containerPort: 80
```

- **Puis apliquez le deployment**

kubectl apply -f webapp-deployment.yaml

Étape 2 : Créer un Service

- **Créez un fichier webapp-service.yaml**

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
spec:
  selector:
    app: webapp
  ports:
  - port: 80
    targetPort: 80
  type: NodePort
```

- **Puis appliquez le service**

kubectl apply -f webapp-service.yaml

Étape 3 : Accéder à l'application

Une fois le Service déployé, il devient possible d'accéder à l'application web. Avec Minikube, il suffit d'exécuter :

```
minikube service webapp-service
```

Sinon, on peut obtenir l'adresse et le port d'accès via :

```
kubectl get service webapp-service
```

Concepts Intermédiaires

Maintenant que vous maîtrisez les bases de Kubernetes, passons à des notions plus avancées qui vous permettront d'optimiser vos déploiements et de mieux gérer vos ressources.

1. Namespaces : Organiser votre cluster

Les namespaces vous permettent de créer des environnements virtuels au sein d'un même cluster physique. C'est un peu comme avoir plusieurs appartements dans un même immeuble - chacun avec son propre espace et ses propres règles.
Imaginez que vous travaillez sur plusieurs projets ou avec différentes équipes. Les namespaces vous permettent de séparer clairement ces contextes

```
fode@serveru:~$ kubectl create namespace mon-namespace
namespace/mon-namespace created
fode@serveru:~$ kubectl get namespaces
NAME              STATUS   AGE
default           Active   17m
kube-node-lease   Active   17m
kube-public       Active   17m
kube-system       Active   17m
mon-namespace     Active   32s
```

2. Gestion des Ressources : Éviter les surprises

Avez-vous déjà eu cette application qui consomme toutes les ressources disponibles ? Dans Kubernetes, vous pouvez définir précisément ce dont chaque conteneur a besoin et ce qu'il peut utiliser au maximum

```yaml
resources:
  requests:
    memory: "64Mi"    # Ce dont l'application a besoin pour démarrer
    cpu: "250m"       # 0.25 CPU (un quart de cœur)
  limits:
    memory: "128Mi"   # Limite maximale de mémoire
    cpu: "500m"       # Limite maximale de CPU (un demi-cœur)
```

Les requests garantissent des ressources minimales tandis que les limits empêchent une application de monopoliser tout le système. C'est comme réserver une table dans un restaurant tout en s'assurant de ne pas dépasser son budget.

3. Stockage Persistant : Garder vos données en sécurité

Dans un monde de conteneurs éphémères, comment conserver vos données importantes ? C'est là qu'interviennent les Persistent Volumes (PV) et les Persistent Volume Claims (PVC).
Le principe est similaire à la location d'un espace de stockage :

1. L'administrateur crée des espaces de stockage disponibles (PV)
2. Votre application demande un espace de stockage (PVC)
3. Kubernetes fait le lien entre les deux

- **L'espace de stockage disponible (créé par l'administrateur)**

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: stockage-donnees
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: standard
  hostPath:
    path: "/data/volume1"
```

- **Votre demande d'espace de stockage**

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fode-base-donnees
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: standard
```

- **Une fois le PVC créé, vous pouvez l'utiliser dans votre pod**

```yaml
volumes:
  - name: donnees-volume
    persistentVolumeClaim:
      claimName: fode-base-donnees
containers:
  - name: postgres
    image: postgres:13
    volumeMounts:
      - mountPath: "/var/lib/postgresql/data"
        name: donnees-volume
```

Ainsi, même si votre pod est supprimé puis recréé, vos données resteront intactes

Mise à l'échelle : S'adapter à la demande

L'un des grands avantages de Kubernetes est sa capacité à s'adapter automatiquement à la charge de travail en ajustant les ressources et en répartissant le trafic de manière optimale. Vous pouvez :

Ajuster manuellement le nombre de répliques selon vos besoins

Passer à 5 instances de votre application

exemple: **kubectl scale deployment mon-app --replicas=5**

Automatiser l'ajustement en fonction de métriques comme l'utilisation CPU :

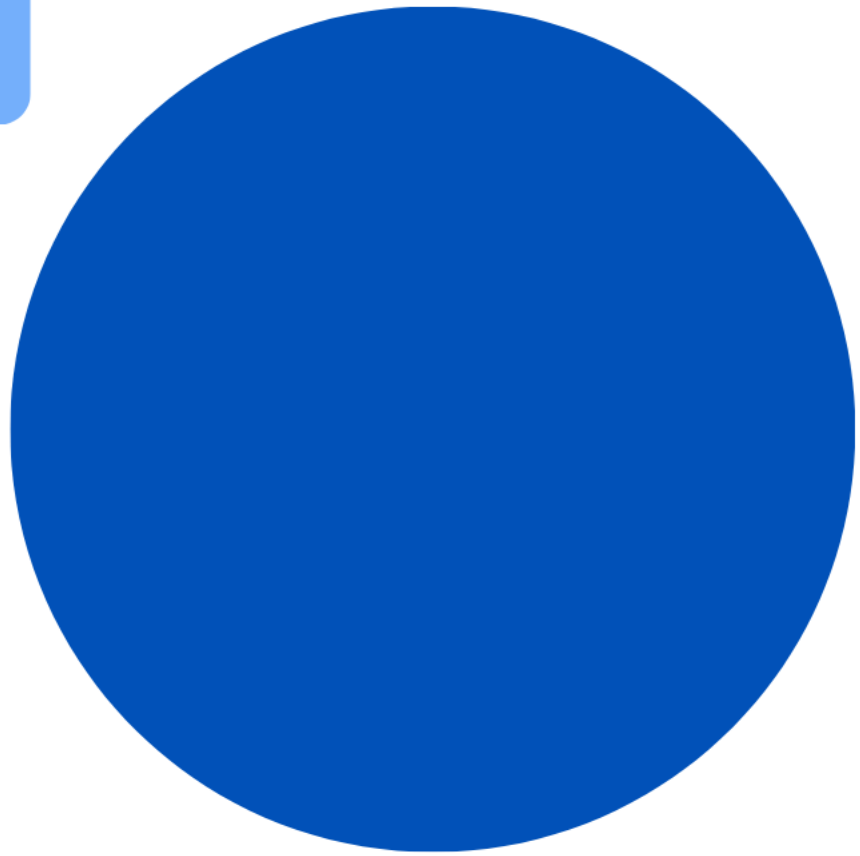
Créer un autoscaler qui maintiendra entre 2 et 10 répliques# selon que l'utilisation CPU dépasse 70%

exemple: **kubectl autoscale deployment mon-app --min=2 --max=10 --cpu-percent=70**

C'est comme avoir une équipe qui s'agrandit automatiquement pendant les heures de pointe et se réduit pendant les périodes calmes - une optimisation parfaite des ressources et des coûts.

Concepts Plus Avancés

Vous avez maintenant acquis les bases essentielles pour démarrer avec Kubernetes. Nous avons couvert les fondamentaux et les concepts intermédiaires qui vous permettront de déployer vos premières applications conteneurisées dans un environnement Kubernetes.
Mais ce n'est que le début de votre voyage dans l'univers de Kubernetes ! Dans mes prochaines publications, nous explorerons ensemble des sujets plus avancés tels que :



- **Les sondes de santé (Health Checks)**

  pour garantir la fiabilité de vos applications

- **Les stratégies de déploiement avancées (Blue/Green, Canary, etc.)**

- **Helm**

  le gestionnaire de paquets qui simplifie le déploiement d'applications complexes

- **Le troubleshooting et débogage**

  pour résoudre les problèmes courants

- **La sécurité dans Kubernetes**

  avec RBAC, Network Policies et bien plus

Si vous avez trouvé ce guide utile, je vous invite à me suivre sur LinkedIn pour ne pas manquer les prochains articles de cette série sur Kubernetes. J'y partagerai également des astuces, des cas d'usage réels et des solutions aux défis que vous pourriez rencontrer dans votre parcours d'adoption de Kubernetes.

Conclusion

Kubernetes révolutionne la façon dont nous déployons et gérons les applications modernes. Ce guide vous a transmis les bases essentielles pour démarrer votre exploration. Je vous encourage à pratiquer, expérimenter et approfondir la maîtrise de cette technologie incontournable. N'hésitez pas à découvrir mon projet Docker, publié sur mon compte LinkedIn, qui pourrait vous inspirer pour vos propres déploiements. Si ce contenu vous a plu, pensez à liker, vous abonner et partager autour de vous pour en faire profiter votre réseau. Restez connectés : la suite arrive très bientôt avec des cas pratiques sur Kubernetes, accompagnés de nouvelles ressources et astuces sur Kubernetes et Docker !

Merci

Pour votre attention à cette présentation.

Remerciements

Je tiens à remercier Monsieur **Massamba Lo**, notre enseignant, pour la qualité de son enseignement et les connaissances qu'il nous a transmises sur Kubernetes. Sa pédagogie a grandement contribué à la réalisation de cette présentation.

https://fodemangane-portfolio.netlify.app/ +221 787388292 fodemangane@gmail.com Dakar, Senegal