INTRODUCTION À KUBERNETES

DÉFINITION

QU'EST-CE QUE KUBERNETES ?

Kubernetes est un système **open-source** conçu pour automatiser la gestion, le déploiement et la mise à l'échelle d'applications **conteneurisées**.

Avantages de Kubernetes	Exemples d'utilisation
Automatisation	CI/CD
Gestion	Orchestration des conteneurs
Mise à l'échelle	Microservices, applications cloud-native

POURQUOI UTILISER KUBERNETES ?

Kubernetes facilite l'**orchestration de conteneurs**, en permettant d'automatiser des tâches complexes, telles que :

- Gestion des dépendances
- Découvrabilité
- Mise à jour
- Tolérance aux pannes

ARCHITECTURE

CONCEPT DE CLUSTER

Un **cluster Kubernetes** est un ensemble de **nœuds** qui travaillent ensemble pour exécuter des **applications conteneurisées**.

Termes	Explications
Cluster	Groupe de nœuds travaillant ensemble
Nœuds	Machines physiques ou virtuelles du cluster
Conteneurisées	Applications empaquetées avec leurs dépendances

COMPOSANTS PRINCIPAUX

NŒUDS

Les nœuds sont des **machines physiques** ou **virtuelles** dans un cluster **Kubernetes**, qui exécutent les **applications conteneurisées**.

Caractéristiques	Description
Machines physiques	Ordinateurs physiques ou serveurs
Machines virtuelles	Instances créées par des hyperviseurs

PODS

Les **pods** sont des groupes de un ou plusieurs **conteneurs** qui fonctionnent ensemble sur un même **nœud**, partageant les mêmes **ressources**, l'**adressage IP** et le **stockage**.

SERVICES

Les services assurent l'**accessibilité** et la **découverte** des pods, en fournissant une **adresse IP stable** et un **nom DNS**.

AVANTAGES ET INCONVÉNIENTS

INSTALLATION ET CONFIGURATION

CHOIX D'UN ENVIRONNEMENT

LOCAL (MINIKUBE)

Minikube est un outil qui permet de créer un **cluster Kubernetes local** à des fins de développement et de test.

CLOUD (GOOGLE CLOUD, AWS, AZURE)

Les principaux fournisseurs de **cloud** proposent des **services managés** pour déployer et gérer des **clusters Kubernetes**.

OUTILS DE GESTION

kubectl

C'est l'outil de ligne de commande pour interagir avec votre **cluster Kubernetes**.

Commande	Description
`kubectl get`	Récupère les ressources
`kubectl create`	Crée des ressources à partir d'un fichier ou d'un stdin
`kubectl apply`	Met à jour les ressources à partir d'un fichier
`kubectl delete`	Supprime les ressources
`kubectl describe`	Affiche des informations détaillées sur les ressources

`kubeadm`

kubeadm est l'outil pour **créer**, **mettre à jour** et **administrer** des clusters Kubernetes.

CONFIGURATION DE BASE

CRÉATION D'UN CLUSTER

1. Installer les outils nécessaires :
    - `kubectl`
    - `kubeadm`
    - etc.
2. Initialiser le cluster avec `kubeadm init`
3. Configurer `kubectl` pour utiliser le fichier de configuration généré

INTRODUCTION À KUBERNETES

DÉPLOIEMENT D'APPLICATIONS

Le déploiement d'applications sur un **cluster Kubernetes** se fait en définissant les **ressources** nécessaires et en utilisant des **fichiers de configuration**.

DÉFINITION DE RESSOURCES

Kubernetes utilise différents types de ressources pour gérer les **applications déployées**.

- **Déploiements**
- **Répliques**
- **ConfigMaps**
- **Secrets**

DÉPLOIEMENTS

Les **déploiements** représentent un ensemble de **réplicas** d'un **pod** géré par **Kubernetes**.

- Gestion de la mise à l'échelle
- Mises à jour incrémentielles
- Rollbacks

RÉPLIQUES

Les **répliques** sont des copies d'un **pod** exécutées sur différents **nœuds** pour assurer la **disponibilité** et la **tolérance aux pannes**.

CONFIGMAPS

Les **ConfigMaps** permettent de stocker des informations de configuration pour les applications déployées.

Les utilisations courantes de ConfigMaps	Exemples
Stocker des variables d'environnement	Clés d'API, adresses de serveurs
Stocker des fichiers de configuration	nginx.conf
Configurer des volumes	Mettre en place des montages NFS

- ConfigMaps peuvent être créées avec *kubectl* ou définies dans un fichier yaml
- Les informations stockées dans ConfigMaps peuvent être utilisées dans l'application

SECRETS

Les **secrets** permettent de stocker des informations **sensibles**, comme des mots de passe ou des clés d'accès, pour les applications déployées.

Avantages	Exemples d'utilisation
Sécurité	Stocker des mots de passe
Isolation	Stocker des clés d'accès

- Les secrets sont stockés dans un volume temporaire
- Les secrets ne sont pas exposés dans les logs ou les variables d'environnement

FICHIERS DE CONFIGURATION (YAML)

Les fichiers de configuration en **YAML** sont utilisés pour décrire les **ressources Kubernetes**.

Exemple de fichier YAML pour un déploiement:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mon-application
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mon-application
  template:
    metadata:
      labels:
        app: mon-application
    spec:
      containers:
```

SYNTAXE ET STRUCTURE

Les fichiers de configuration sont écrits en **YAML**, un langage de balisage léger.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
```

CRÉATION ET MODIFICATIONS

Pour créer ou modifier un fichier de configuration **YAML**, il suffit d'éditer le fichier avec un éditeur de texte.

UTILISATION AVEC KUBECTL

Pour appliquer les modifications d'un fichier de configuration **YAML**, on utilise la commande `kubectl apply -f fichier.yaml`.

GESTION DES APPLICATIONS DÉPLOYÉES

Après le déploiement d'une application, **Kubernetes** fournit divers outils pour gérer les applications déployées.

- Gérer et modifier les **configurations** des applications
- Mettre à jour le **code** de l'application
- **Redimensionner** les ressources allouées à l'application
- **Analyser** les performances de l'application

MISE À JOUR

Pour mettre à jour une **application déployée**, on modifie le **fichier de configuration** et on réapplique les changements avec `kubectl apply`.

ROLLBACK

En cas de problème lors d'une mise à jour, on peut revenir à la version précédente en utilisant `kubectl rollback`.

MONITORING ET LOGS

Kubernetes fournit des outils pour **surveiller** les applications déployées et afficher les **logs**, comme `kubectl logs` et `kubectl top`.

Commande	Description
`kubectl logs`	Affiche les logs d'un conteneur
`kubectl top`	Affiche les ressources (CPU, Mémoire) utilisées

SERVICES ET RÉSEAUX

CONCEPTS RÉSEAU

ARCHITECTURE RÉSEAU

Kubernetes offre une architecture réseau **unifiée** pour les applications en conteneurs. Elle permet une communication **transparente** entre les **pods** et les **services**.

ENVIRONNEMENT RÉSEAU VIRTUEL

Kubernetes utilise un **environnement réseau virtuel** pour abstraire le réseau physique sous-jacent.

ISOLATION RÉSEAU

L'**isolation réseau** est utilisée pour séparer les composants et limiter l'accès aux **ressources sensibles**.

Avantages	Exemples d'utilisation
Sécurité	Limitation des accès
Contrôle de flux	Séparation des environnements (dev, prod)

SERVICES

DÉFINITION ET UTILITÉ

Un **service Kubernetes** est une abstraction qui définit un ensemble de **pods** fonctionnant ensemble comme une unité. Il permet d'exposer les applications à l'extérieur ou à d'autres services.

Tableau des types de service :

Type de service	Description
ClusterIP	Expose le service sur une IP interne du cluster. Le service est accessible uniquement à l'intérieur du cluster.
NodePort	Expose le service sur chaque IP de nœud sur un port statique. Le service est accessible à l'extérieur du cluster via l'adresse IP de nœud et le port statique.
LoadBalancer	Expose le service à l'aide d'un équilibreur de charge externe et attribue une IP fixe à celui-ci.
ExternalName	Crée un enregistrement DNS pour le service et renvoie un CNAME avec le nom externe. Ce service ne possède pas d'IP dédiée.

TYPES DE SERVICES

- **ClusterIP**
- **NodePort**
- **LoadBalancer**
- **ExternalName**

Type de service	Description
ClusterIP	Expose le service sur une IP interne au cluster, le rendant accessible uniquement à l'intérieur du cluster.
NodePort	Expose le service sur chaque IP du nœud à un port statique (le NodePort).
LoadBalancer	Expose le service en externe en utilisant l'équilibreur de charge de l'environnement.
ExternalName	Mappe le service à un nom DNS externe défini.

TYPES DE SERVICES

CLUSTERIP

Le type de service **ClusterIP** expose l'**application** à l'intérieur du **cluster** via une adresse IP interne.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  type: ClusterIP
```

NODEPORT

Le type de service **NodePort** expose l'application à l'extérieur du cluster via un port spécifié sur chaque nœud.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
      nodePort: 30080
  type: NodePort
```

LOADBALANCER

Le type de service **LoadBalancer** expose l'application à l'extérieur du cluster via un **équilibreur de charge**.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  type: LoadBalancer
```

EXTERNALNAME

Le type de service **ExternalName** permet de mapper un service à un nom DNS externe.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ExternalName
  externalName: my.external.example.com
```

INGRESS ET CONTRÔLEURS D'ENTRÉE

GESTION DU TRAFIC

Les ressources **Ingress** permettent de gérer l'accès aux **services** à partir de l'extérieur du **cluster**.

Ingress	Gère l'accès externe
Cluster	Collection de nœuds Kubernetes
Services	Pods regroupés pour un usage

SSL/TLS TERMINAISON

Les **contrôleurs d'entrée** supportent la **terminaison SSL/TLS** pour sécuriser la communication avec les services.

PROTECTION AVEC DES RÈGLES

Il est possible de configurer des **règles** pour contrôler l'accès aux **services** et assurer leur **sécurité**.

- Les règles de contrôle d'accès permettent de déterminer qui peut accéder aux ressources.
- Les règles de sécurité peuvent être basées sur des rôles ou des autorisations.

GESTION DU STOCKAGE

VOLUMES

CONCEPTS DE BASE

Kubernetes utilise des objets appelés **Volumes** pour gérer les données des **conteneurs** dans un **Pod**.

TYPES DE VOLUMES

Kubernetes propose différents types de volumes pour répondre aux besoins spécifiques de **stockage**.

- **EmptyDir**
- **HostPath**
- **PersistantVolumeClaim (PVC)**

EMPTYDIR

DÉFINITION

Un volume **EmptyDir** est temporaire et dure seulement pendant la durée de vie du **Pod**.

```
volumes:
- name: cache-volume
  emptyDir: {}
```

UTILISATION

Convient pour stocker des **données temporaires**, des **caches** ou des **fichiers intermédiaires**.

- **Données temporaires** : Les données générées par l'application qui n'ont pas besoin d'être persistantes et peuvent être supprimées à tout moment
- **Caches** : Sauvegarde temporaire des données pour réduire le temps de réponse des requêtes
- **Fichiers intermédiaires** : Fichiers générés lors du traitement des données, qui sont utilisés pour effectuer d'autres opérations

HOSTPATH

DÉFINITION

Un volume **HostPath** permet d'exposer un fichier ou un répertoire du système de fichiers de l'hôte (noeud) à un **Pod**.

```
volumes:
- name: logs-volume
  hostPath:
    path: /var/log/app
    type: DirectoryOrCreate
```

UTILISATION

Kubernetes : Convient pour stocker des fichiers **partagés** entre les **Pods** exécutés sur le même **hôte**.

PERSISTENTVOLUMECLAIM (PVC)

DÉFINITION

Un **PVC** est une demande de **stockage persistant**, indépendant du cycle de vie du **Pod**.

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: data-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

UTILISATION

Convient pour stocker des **données persistantes**, telles que des **bases de données** ou des **fichiers partagés** entre plusieurs Pods.

CLUSTERS ÉTENDUS ET FÉDÉRATION

CONCEPTS DE BASE

UTILITÉ ET CAS D'UTILISATION

- Permet de **regrouper** plusieurs clusters **Kubernetes**
- **Répartition** des charges de travail
- Tolérance aux pannes (**HA**)

AVANTAGES ET INCONVÉNIENTS

Avantages:

- **Scalabilité horizontale**
- Isolation des ressources
- Disponibilité accrue

Inconvénients:

- Complexité de gestion
- Latence possible

FÉDÉRATION DE CLUSTERS

Une fédération de clusters **Kubernetes** permet de gérer plusieurs clusters comme s'ils étaient un seul cluster.

Avantages	Exemples d'utilisation
Répartition de charge	Applications globales
Tolérance aux pannes	Environnements critiques
Isolation des ressources	Projets séparés

CONFIGURATION

- **Deployment** de l'API de **fédération** et du contrôleur **fédéré**
- Registre des **clusters** dans la **fédération**

EXEMPLE D'ARCHITECTURE

Cluster 1	Cluster 2
Nœud 1	Nœud 1
Nœud 2	Nœud 2
...	...

AGRÉGATION DES RESSOURCES

- Les **ressources** sont réparties entre les **clusters**
- Les **répliques** peuvent être exécutées sur plusieurs clusters