



# Introduction à la formation Kubernetes

De Docker à l'orchestration de conteneurs

# Docker : rappels historiques, vue d'ensemble

## Évolution des conteneurs

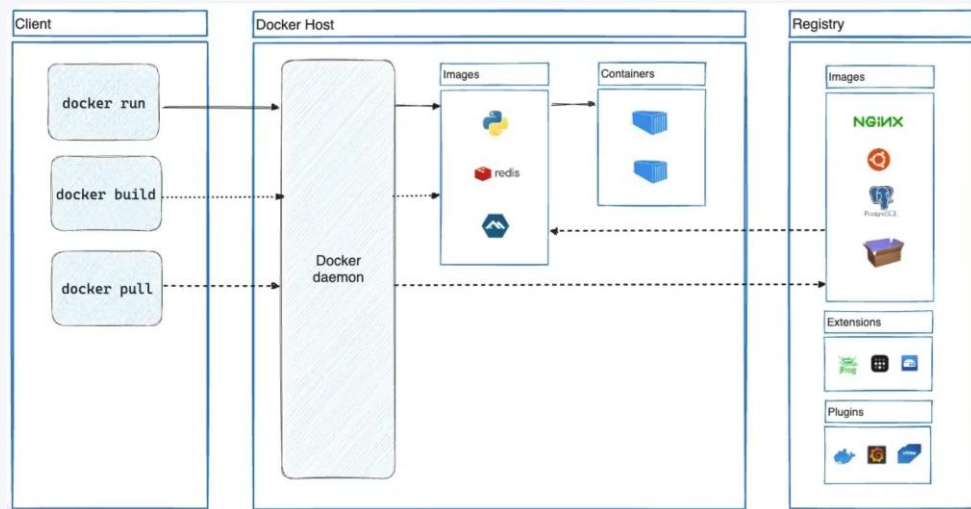
- 1979 : Concept initial avec chroot dans UNIX
- 2000-2008 : FreeBSD Jails, Solaris Containers, LXC
- 2013 : Lancement de Docker, révolutionnant l'utilisation des conteneurs
- 2015+ : Adoption massive par l'industrie, standardisation

## Qu'est-ce qu'un conteneur Docker ?

Un conteneur est une unité logicielle standardisée qui empaquette le code et toutes ses dépendances pour que l'application s'exécute rapidement et de manière fiable d'un environnement à un autre.

### Avantages des conteneurs

- 📦 Portabilité : fonctionne partout de manière identique
- 🛡️ Isolation : sécurité et indépendance des applications
- 🚀 Légèreté : démarrage rapide, utilisation efficace des ressources
- 🔄 Reproductibilité : environnements cohérents et prévisibles



# Conteneurs et orchestration

## Qu'est-ce que l'orchestration ?

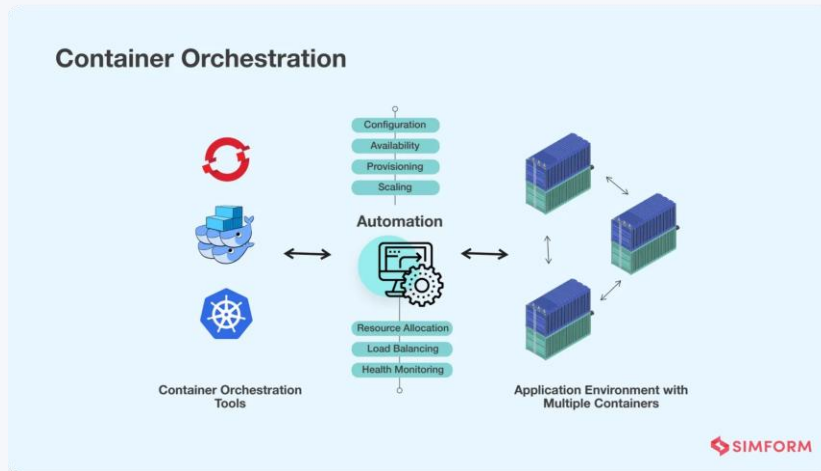
L'orchestration des conteneurs consiste à **automatiser** le déploiement, la gestion, la mise à l'échelle et la mise en réseau des conteneurs tout au long de leur cycle de vie.

## Limites des conteneurs individuels

- ⚠ Gestion manuelle complexe à grande échelle
- ⚠ Pas de haute disponibilité ou reprise après panne intégrée
- ⚠ Difficultés pour la mise à l'échelle dynamique
- ⚠ Communication inter-conteneurs complexe à configurer

## Pourquoi orchestrer ?

- ✓ Déploiement automatisé et cohérent à grande échelle
- ✓ Haute disponibilité et résilience des applications
- ✓ Optimisation des ressources et réduction des coûts
- ✓ Gestion simplifiée des environnements complexes



# Fonctionnalités d'orchestration

Les plateformes d'orchestration de conteneurs offrent un ensemble de fonctionnalités essentielles pour gérer efficacement les applications conteneurisées à grande échelle.



## Déploiement et planification

Automatisation du déploiement des conteneurs sur les nœuds disponibles selon des règles définies (affinité, anti-affinité, contraintes de ressources).



## Équilibrage de charge

Distribution automatique du trafic entre les instances de conteneurs pour optimiser les performances et assurer la disponibilité des services.



## Découverte de services

Mécanisme permettant aux applications de localiser et communiquer avec d'autres services sans connaître leur emplacement physique dans le cluster.



## Stockage persistant

Gestion des volumes de données qui survivent au cycle de vie des conteneurs, permettant la persistance des données entre les redémarrages.



## Auto-réparation

Surveillance continue de l'état des conteneurs avec redémarrage automatique en cas de défaillance pour maintenir l'état souhaité du système.



## Mise à l'échelle automatique

Ajustement dynamique du nombre d'instances de conteneurs en fonction de la charge de travail ou de métriques personnalisées.



## Gestion des secrets

Stockage et distribution sécurisés des informations sensibles (mots de passe, clés API, certificats) aux conteneurs qui en ont besoin.



## Surveillance et journalisation

Collecte et agrégation des métriques et des journaux pour le suivi des performances et le diagnostic des problèmes.

# Présentation générale de Kubernetes




## Qu'est-ce que Kubernetes ?

Kubernetes (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.

## Origine et gouvernance

- **2014** : Annoncé par Google, basé sur Borg
- **2015** : Donné à la Cloud Native Computing Foundation
- **Aujourd'hui** : Standard de facto pour l'orchestration

## Principes fondamentaux

-  **Déclaratif**  
Vous déclarez l'état souhaité, Kubernetes s'occupe de le maintenir
-  **Portable**  
Fonctionne sur site, dans le cloud public ou hybride
-  **Extensible**  
Architecture modulaire avec API extensibles



*Le logo de Kubernetes représente un gouvernail (timonier)*

## Cas d'utilisation typiques





Déploiement d'applications microservices  
Environnements cohérents (dev, test, prod)  
Applications cloud-natives à haute disponibilité  
Plateformes en tant que service (PaaS)

# Architecture de Kubernetes




## Vue d'ensemble de l'architecture

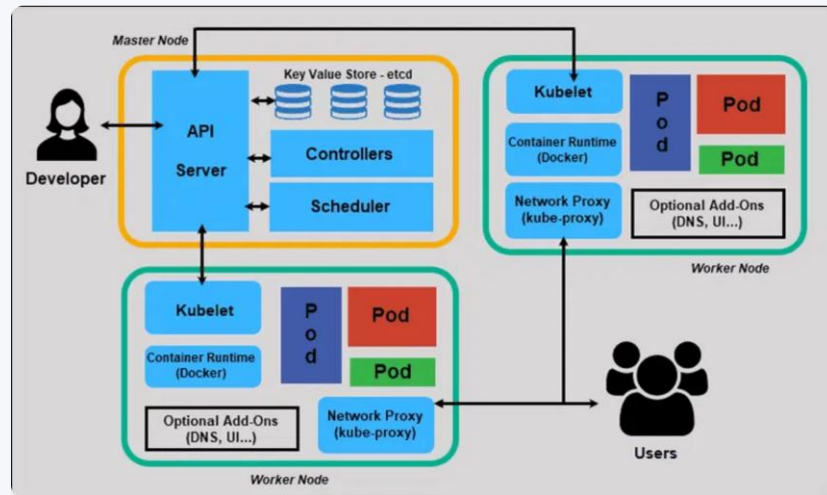
L'architecture de Kubernetes est composée de deux parties principales : le **Plan de Contrôle** (anciennement appelé Master) et les **Nœuds** (Nodes) qui exécutent les applications.

### 1 Composants du Plan de Contrôle

-  **API Server (kube-apiserver)** : Point d'entrée pour toutes les commandes REST
-  **etcd** : Base de données clé-valeur stockant l'état du cluster
-  **Scheduler (kube-scheduler)** : Attribue les pods aux nœuds
-  **Controller Manager** : Régule l'état du cluster

### 2 Composants des Nœuds

-  **Kubelet** : Agent qui s'assure que les conteneurs fonctionnent dans un pod
-  **Kube-proxy** : Gère les règles réseau et le routage
-  **Container Runtime** : Logiciel responsable de l'exécution des conteneurs (Docker, containerd, CRI-O)



# Terminologie : maître, noeuds, pods, services...

## Architecture



### Cluster

Ensemble de machines (nœuds) qui exécutent des applications conteneurisées



### Plan de contrôle

Composants qui contrôlent le cluster (anciennement "Maître")



### Nœud (Node)

Machine physique ou virtuelle qui exécute les charges de travail

## Composants du plan de contrôle



### kube-apiserver

Point d'entrée de l'API Kubernetes pour toutes les opérations



### etcd

Base de données clé-valeur qui stocke les données du cluster



### kube-scheduler

Attribue les pods aux nœuds selon les contraintes

## Objets fondamentaux



### Pod

Plus petite unité déployable, groupe de conteneurs partageant des ressources



### Labels

Paires clé-valeur attachées aux objets pour l'identification



### Service

Abstraction qui définit un ensemble de pods et une politique d'accès

## Contrôleurs et outils



### Contrôleur de réplication

Assure qu'un nombre spécifié de répliques de pods s'exécutent



### Deployment

Gère le déploiement et la mise à jour des applications



### kubectl

Outil en ligne de commande pour interagir avec le cluster

# Positionnement sur le marché



## Kubernetes

- ✓ Solution la plus complète et riche en fonctionnalités
- ✓ Large communauté et écosystème mature
- ✓ Support natif par les principaux fournisseurs cloud



## Docker Swarm

- ✓ Intégration native avec l'API Docker
- ✓ Plus simple à configurer et utiliser
- ! Fonctionnalités limitées par rapport à Kubernetes



## Apache Mesos

- ✓ Approche modulaire pour charges diverses
- ✓ Utilisé par de grandes entreprises (Twitter, Airbnb)
- ! Complexité élevée mais grande flexibilité

Feature	Docker Swarm	Kubernetes	Apache Mesos	CoreOS Fleet
Primary Use Case	Simple container orchestration	Enterprise-grade container management	General workload orchestration	Lightweight service orchestration
Complexity	Low	High	Medium-High	Low
Scalability	Limited	Highly scalable	Very high	Limited
Networking	Swarm Overlay Network	CNI (Calico, Flannel)	Custom (Weave, Flannel)	Systemd-native
Load Balancing	Built-in (Routing Mesh)	Built-in (Ingress, Services)	External (Marathon)	None
Self-Healing	Yes	Yes	Yes	No
Multi-Cloud Support	No	Yes	Yes	No
Storage	Docker Volumes	Persistent Volumes (PV)	External plugins	Systemd Units
Service Discovery	Internal DNS	CoreDNS	External Marathon-LB	Systemd



# Intégration avec les autres plateformes

Kubernetes s'intègre avec un vaste écosystème d'outils et de services, ce qui en fait une plateforme extrêmement flexible et adaptable à différents environnements et besoins.



## Fournisseurs Cloud



AWS (EKS)



Google Cloud (GKE)



Azure (AKS)



DigitalOcean (DOKS)



IBM Cloud Kubernetes



## CI/CD & DevOps



Jenkins



GitLab CI



GitHub Actions



ArgoCD



Flux



## Monitoring & Logging



Prometheus



Grafana



Elasticsearch



Fluentd



Datadog



## Stockage



Ceph



Rook



Longhorn



OpenEBS



## Réseau



Calico



Flannel



Weave Net



Cilium



## Sécurité



Vault (HashiCorp)



Falco



Open Policy Agent



Kyverno

# Conclusion et prochaines étapes

## Points clés à retenir



### Conteneurs

Unités logicielles standardisées qui encapsulent le code et ses dépendances pour une exécution cohérente



### Kubernetes

Plateforme open source d'orchestration qui automatise le déploiement, la mise à l'échelle et la gestion des applications conteneurisées



### Architecture

Composée d'un plan de contrôle et de nœuds, avec des composants spécialisés pour chaque fonction



### Objets fondamentaux

Pods, services, déploiements et autres ressources qui forment le modèle déclaratif de Kubernetes

## Prochaines étapes de la formation



### Travaux pratiques

Installation et configuration d'un cluster Kubernetes, déploiement d'applications simples



### Concepts avancés

Approfondissement des concepts de déploiement, mise à l'échelle, et gestion des ressources



### Intégration cloud

Utilisation de Kubernetes avec les principaux fournisseurs cloud (AWS, GCP, Azure)