# Docker Swarm pour Linux - Orchestration

# Terminologie clé

- **Nœud (Node)** : Un serveur (machine physique ou VM) dans un cluster Docker Swarm.
- **Cluster** : Un ensemble de nœuds regroupés pour exécuter des services en parallèle.
- **Manager** : Le nœud qui contrôle l'état global du Swarm.
- **Worker** : Les nœuds qui exécutent les conteneurs, sous les ordres du manager.
- **Service** : Une tâche à exécuter dans Swarm.
- **Task** : Un conteneur unique, partie d'un service dans Docker Swarm.

# Qu'est-ce que Docker Swarm ?

- **Docker Swarm** est un orchestrateur de conteneurs intégré à Docker, permettant de gérer un cluster de nœuds Docker comme une seule entité.
- Il automatise le déploiement, la gestion, la mise à l'échelle et la disponibilité des applications conteneurisées.
- **Points clés** :
  - Haute disponibilité
  - Équilibrage de charge
  - Mise à l'échelle automatique

# Comment fonctionne Docker Swarm ?

- Chaque nœud exécute Docker et communique avec d'autres nœuds.
- **Manager Nodes** gèrent l'état du cluster et distribuent des tâches aux **Worker Nodes**.
- **Consensus Raft** est utilisé pour garantir une synchronisation distribuée entre les nœuds managers.

# Architecture et Processus de Docker Swarm

- Docker Swarm est une solution d'orchestration de conteneurs qui permet de gérer des clusters de nœuds Docker. L'architecture de Docker Swarm est conçue pour assurer la haute disponibilité, la scalabilité et la tolérance aux pannes. Voici les principaux composants et processus impliqués dans Docker Swarm.
- Architecture de Docker Swarm
- L'architecture de Docker Swarm repose sur deux types principaux de nœuds :
- Nœud Manager (Manager Node)
  - Rôle : Le nœud manager gère le cluster et son état global.
  - Fonctionnalités :
    - Gère la planification des tâches (tasks) et distribue les workloads aux nœuds workers.
    - Maintient l'état souhaité du Swarm (par exemple, s'assurer qu'il y a toujours 3 réplicas d'un service en cours d'exécution).
    - Implémente un algorithme de consensus appelé Raft pour garantir qu'au moins la majorité des managers (un quorum) sont d'accord sur les décisions du cluster.
    - Peut aussi exécuter des conteneurs comme un nœud worker, mais ce n'est pas une pratique recommandée en production.

# Architecture et Processus de Docker Swarm

- Rôle : Le nœud worker exécute les tâches qui lui sont assignées par le manager
- Fonctionnalités :
  - Ne prend pas part aux décisions de gestion du cluster.
  - Exécute simplement les conteneurs en fonction des instructions du nœud manager.
  - Peut être ajouté ou retiré dynamiquement du Swarm sans affecter l'état global du cluster.

# Architecture et Processus de Docker Swarm

- Services et Tasks
  - Service : Un service est la définition d'une application conteneurisée, incluant des spécifications comme l'image Docker à utiliser, le nombre de réplicas, et les ports exposés. C'est une abstraction au-dessus des conteneurs.
  - Task : Une tâche représente une instance de conteneur appartenant à un service. Chaque réplique d'un service est une tâche.
- Overlay Network (Réseau Overlay)
  - Docker Swarm crée des réseaux overlay pour permettre une communication sécurisée entre les conteneurs sur différents nœuds, même s'ils sont situés sur des machines différentes.
  - Tous les nœuds dans un cluster Swarm peuvent communiquer entre eux via ces réseaux.
- Load Balancing (Équilibrage de charge)
  - Swarm intègre un équilibrage de charge par défaut : si un service a plusieurs réplicas, les requêtes entrantes seront distribuées de manière égale entre eux.
  - Docker crée un Virtual IP (VIP) pour chaque service, ce qui permet de distribuer les requêtes entrantes vers n'importe quelle réplique du service.

# Processus de Docker Swarm

- Voici comment Docker Swarm fonctionne pour gérer les conteneurs dans un cluster.
- Initialisation du Swarm
  - Le Swarm est initialisé avec la commande suivante :
    - docker swarm init
  - Cela transforme le nœud actuel en nœud manager. Il génère également un jeton pour ajouter d'autres nœuds au cluster.
- Ajout de Nœuds au Cluster
  - D'autres nœuds peuvent rejoindre le cluster comme workers ou managers à l'aide du jeton fourni par le nœud manager.
  - Pour ajouter un nœud worker :
    - docker swarm join --token <worker_token> <manager_ip>:23772.
- Planification et Répartition des Tâches
  - Le nœud manager distribue les tasks aux nœuds workers, selon les spécifications du service (nombre de réplicas, par exemple).
  - Swarm garantit que l'état souhaité est maintenu (c'est-à-dire que le nombre de réplicas demandé pour un service est toujours disponible).

# Processus de Docker Swarm

- Réplication et Tolérance aux Pannes
  - Si un nœud worker échoue ou se déconnecte, le nœud manager recrée automatiquement les tâches du nœud défaillant sur d'autres nœuds disponibles.
  - Cela garantit une tolérance aux pannes en maintenant la disponibilité des services même en cas de défaillance de certains nœuds.
- Mise à jour des Services (Rolling Update)
  - Docker Swarm permet la mise à jour progressive (rolling update) des services. Par exemple, si vous mettez à jour un service, les réplicas sont mis à jour un par un sans interrompre l'application.
    - docker service update --image <new_image> <service_name>
- Mise à l'Échelle Automatique
  - Vous pouvez facilement faire évoluer un service (ajouter ou retirer des réplicas) avec une commande unique :
    - docker service scale <service_name>=<number_of_replicas>
  - Swarm gère automatiquement la répartition des conteneurs sur les différents nœuds du cluster.

# Résumé

- Docker Machine : provisionne les hôtes et installe Docker Engine.
- Docker Compose : déploie des applications multi-conteneurs en créant les conteneurs requis.
- Docker Swarm : regroupe plusieurs hôtes Docker sous un seul hôte. Il peut également s'intégrer à n'importe quel outil fonctionnant avec un seul hôte Docker.

# Avantages de Docker Swarm

- **Facilité de configuration** : intégré dans Docker, pas besoin de logiciel tiers.
- **Tolérance aux pannes** : si un nœud échoue, les tâches sont redistribuées aux autres.
- **Équilibrage de charge** : les requêtes sont distribuées uniformément entre les services.
- **Mise à l'échelle facile** : ajouter des nœuds ou des réplicas de services selon la demande.

# Load balancing.

- Dans Docker Swarm, l'équilibrage de charge se réfère à la répartition automatique du trafic réseau entre les différents conteneurs d'un service.
- Lorsque vous déployez un service avec plusieurs réplicas, Docker Swarm distribue les requêtes entrantes de manière équilibrée entre ces réplicas pour garantir que la charge de travail est répartie uniformément.
- Cela améliore la disponibilité, la performance et la tolérance aux pannes de l'application.

# Configuration d'un Swarm

- Initialiser un Swarm :
  - docker swarm init
  - (Cela transforme le nœud actuel en manager)
- Ajouter des Worker Nodes :
  - Commande (générée par le manager après docker swarm init) :
  - docker swarm join --token <token> <manager-ip>:2377
  - Les nœuds workers exécutent cette commande pour rejoindre le Swarm.

# Commandes Docker Swarm de base

- Voir l’état du Swarm :
  - docker node ls
- Créer un service (exemple avec Nginx) :
  - docker service create --name web --replicas 3 -p 80:80 nginx
- Lister les services actifs :
  - docker service ls
- Redimensionner un service :
  - docker service scale web=5

# Gestion des Services

- Créer un service répliqué :
- Crée plusieurs copies d'un conteneur pour une meilleure disponibilité.
- Exemple :
  - docker service create --name vote --replicas 3 -p 5000:80 vote-app
- Mise à jour d'un service :
- Met à jour un service existant (par ex., changement de version).
- Exemple :
  - docker service update --image nginx:latest web
- Supprimer un service :
  - docker service rm web

# Équilibrage de charge dans Docker Swarm

- Les requêtes entrantes sont automatiquement réparties entre les réplicas de service via un mécanisme d'équilibrage de charge.
- Tous les nœuds workers et managers peuvent gérer les requêtes réseau pour les services exposés.

# Tolérance aux pannes et récupération

- Si un nœud worker échoue, les tâches exécutées dessus sont redistribuées automatiquement aux nœuds restants.
- Le **mode manager redondant** permet d'éviter un point unique de défaillance.
  - Recommandé d'avoir un nombre impair de nœuds managers (3, 5, etc.) pour garantir la tolérance aux pannes.

# Mise à l'échelle d'un Swarm

- Vous pouvez facilement ajouter plus de nœuds au Swarm ou augmenter le nombre de réplicas d'un service :
  - docker service scale <service_name>=<nombre_de_réplicas>
- Swarm ajustera dynamiquement les ressources et équilibrera la charge.

# Exemple pratique

- Déploiement d’une application
- Créez un fichier docker-compose.yml pour définir les services :

```
version: '3’
services:
    web:
        image: nginx
        ports:
            - "80:80"
        deploy:
            replicas: 3
```

- Déployez l’application dans le Swarm :
  - docker stack deploy --compose-file docker-compose.yml my_stack

# Conclusion

- Docker Swarm est un outil puissant et intuitif pour l'orchestration de conteneurs, offrant une solution intégrée et facile à configurer pour gérer des clusters Docker.
- Grâce à sa simplicité, il est particulièrement adapté aux applications de petite à moyenne envergure ou aux environnements où une orchestration rapide et efficace est requise, sans complexité excessive.
- Swarm se distingue par une courbe d'apprentissage douce, ce qui le rend accessible aux développeurs et équipes souhaitant rapidement adopter la mise à l'échelle automatique, la tolérance aux pannes et l'équilibrage de charge.
- C’est une excellente option pour les entreprises qui utilisent Docker en production et souhaitent éviter la surcharge administrative d’outils plus complexes.

# Conclusion

- Cependant, pour les environnements nécessitant une haute scalabilité, une flexibilité avancée ou des intégrations complexes, Kubernetes reste une alternative plus robuste et largement adoptée, notamment dans les infrastructures cloud.
- Docker Swarm, avec sa simplicité et son intégration native à Docker, représente toutefois une solution solide pour de nombreux cas d'utilisation dans le monde des conteneurs.