Virtualisation et cloud computing

Elyes Gassara

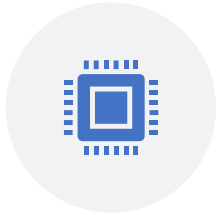
AU. 2021-2022

PLAN

INTRODUCTION GÉNÉRALE

VIRTUALISATION DES SERVEURS

VIRTUALISATION DES RÉSEAUX



VIRTUALISATION DE STOCKAGE

VIRTUALISATION DES POSTES DE TRAVAIL

VIRTUALISATION DES APPLICATIONS

LES CONTENEURS

VIRTUALISATION ET CLOUD COMPUTING

Les conteneurs : Plan

- Objectifs
- DevOps
- Serveur sans Docker
- Serveur avec Docker
- Image Docker, Docker Hub, Docker File, Docker compose, …
- Orchestration et gestion des conteneurs
- Docker Swarm
- Google Kubernetes

Les conteneurs : Orchestration et gestion des conteneurs

Dans le développement moderne, les applications ne sont plus monolithiques. Elles sont au contraire composées de douzaines voire de centaines de composants mis en conteneurs, associés de manière souple, qui doivent fonctionner ensemble pour permettre à telle ou telle application de fonctionner correctement. L'orchestration de conteneur désigne le processus d'organisation du travail des composants individuels et des niveaux d'application.

Tandis que des plateformes telles qu'Apache Mesos, Google Kubernetes et Docker Swarm disposent chacune de méthodologies spécifiques pour la gestion des conteneurs, les moteurs d'orchestration de conteneurs permettent aux utilisateurs de contrôler le démarrage et l'arrêt des conteneurs, de les regrouper en clusters et de coordonner tous les processus composant une application.

Les outils d'orchestration de conteneur permettent aux utilisateurs de guider le déploiement de conteneur et d'automatiser les mises à jour, la surveillance d'état et les procédures de basculement.

Orchestration et gestion des conteneurs

Orchestration et gestion des conteneurs

Malgré les différences qu'on peut trouver entre ces solutions, elles répondent principalement à des besoins d'automatisation de la gestion des cycles de vies des conteneurs :

–**L'approvisionnement et le placement des conteneurs** : l'orchestrateur répartit et déploie les conteneurs sur les machines hôtes selon des besoins spécifiés en termes de mémoire et CPU.

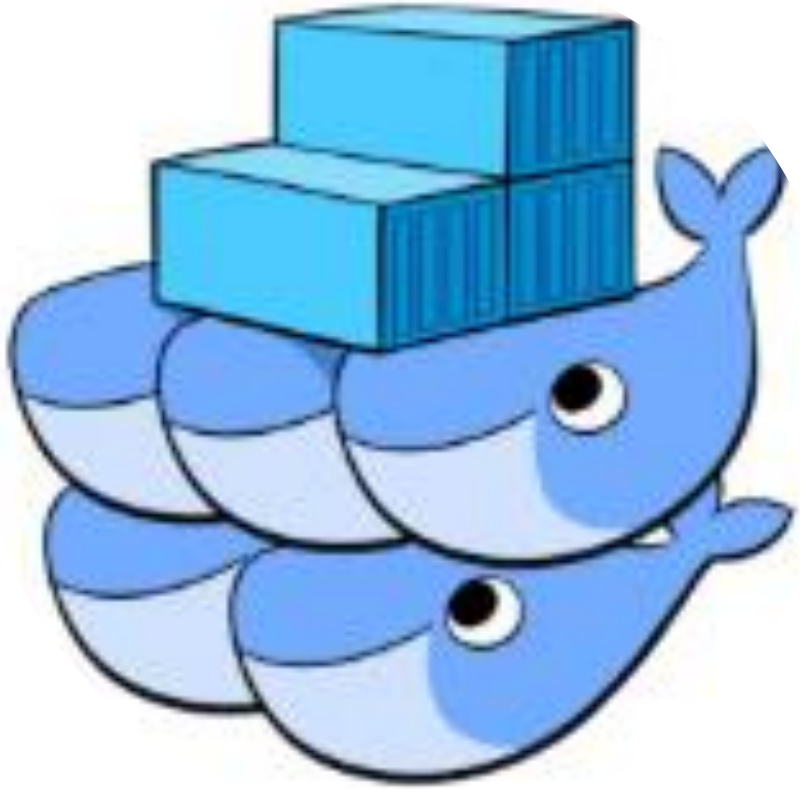
– **Le monitoring** : une vue d'ensemble sur les métriques et les health-checks à la fois des conteneurs et des machines hôtes peut être portée par l'outil d'orchestration.

– **La gestion du failover des conteneurs et la scalabilité** : en cas d'indisponibilité d'une machine hôte par exemple, l'orchestrateur permet de redémarrer le conteneur sur une deuxième machine hôte. Cette scalabilité, en fonction de la solution d'orchestration utilisée, peut être manuelle ou automatique (autoscaling).

– **La gestion des mises à jour et rollbacks des conteneurs** : le principe du rolling update permet à l'orchestrateur d'assurer la mise à jour des conteneurs de manière successive et sans induire d'indisponibilité applicative. Pendant la phase de mise à jour d'un conteneur, les autres conteneurs disponibles sont exécutés.

Docker-Swarm

- Swarm ou Docker-Swarm est un outil conçu pour enrichir Docker Engine, qui est un moteur d'exécution, et permettre à Docker d'offrir un « mode Swarm ». Ce mode donne la possibilité de créer des clusters de machines exécutants des conteneurs Docker, qui fonctionnent ensemble comme une seule machine.
- Vous pouvez ainsi exécuter des commandes Docker sur les machines composant le cluster, comme s'il s'agissait d'une seule et même machine. Le cluster est contrôlé depuis une machine maître, appelée Node Manager (nœud manager).



Docker-Swarm

Docker-Swarm se compose de deux parties distinctes : les nœuds et les services.

Les noeuds

- Les nœuds sont de deux types, les Managers et les Workers.
- Les Manager Nodes s'occupent des tâches de gestion, de supervision et de contrôle du cluster. Les managers assurent donc la stabilité et la performance du Swarm et des services qu'il exécute.
- Les Workers Nodes en revanche n'ont aucun droit de gestion et se chargent uniquement d'exécuter les conteneurs. Chaque Worker a besoin d'au moins un Manager pour fonctionner.
- Par défaut, chaque Manager est aussi un Worker mais il est possible d'empêcher un Manager d'exécuter des tâches des Workers, en ajustant sa configuration par défaut.

Docker-Swarm

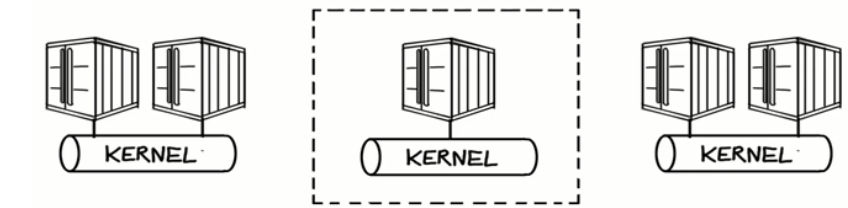
Les services

- Outre les Nodes, Swarm fonctionne grâce aux services, qui sont des descriptions de l'état qu'on souhaite garder pour les nœuds du cluster. Pour fonctionner, un service a besoin d'un conteneur et de commandes à exécuter sur celui-ci.
- Les services exécutés sur Swarm peuvent avoir plusieurs caractéristiques, telles que :
  - Options des services : lors de la création du service, vous pouvez configurer plusieurs paramètres selon les besoins de vos applications (limites mémoire, le nombre des répliques de l'image à exécuter sur Swarm, etc.).
  - État désiré : le déploiement du service permet de définir l'état désiré sur le Swarm. L'état désiré représente le comportement normal ou la configuration idéale de l'application sur Swarm. Par exemple, lorsqu'un problème survient et met à défaut l'état désiré, les « Manager Nodes » interviennent pour corriger le problème en affectant plus de ressources au service.

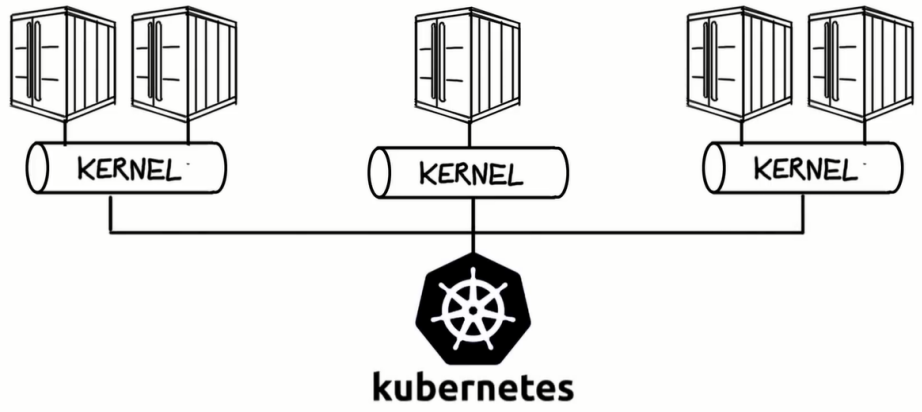
Les conteneurs Google Kubernetes

- Orchestration et gestion des conteneurs sur des clusters de serveurs.

- Kubernetes (communément appelé « K8s ») est un système open source qui vise à fournir une « plate-forme permettant d'automatiser le déploiement, la montée en charge et la mise en œuvre de conteneurs d'application sur des clusters de serveurs »

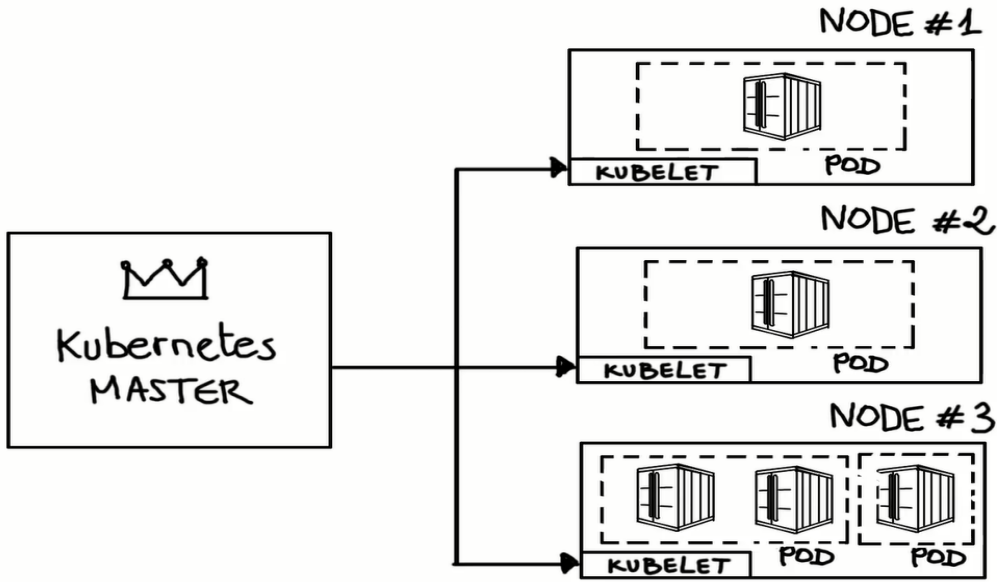


Le conteneur n'a pas conscience de ce qui se passe en dehors de son kernel et donc de la machine hôte.

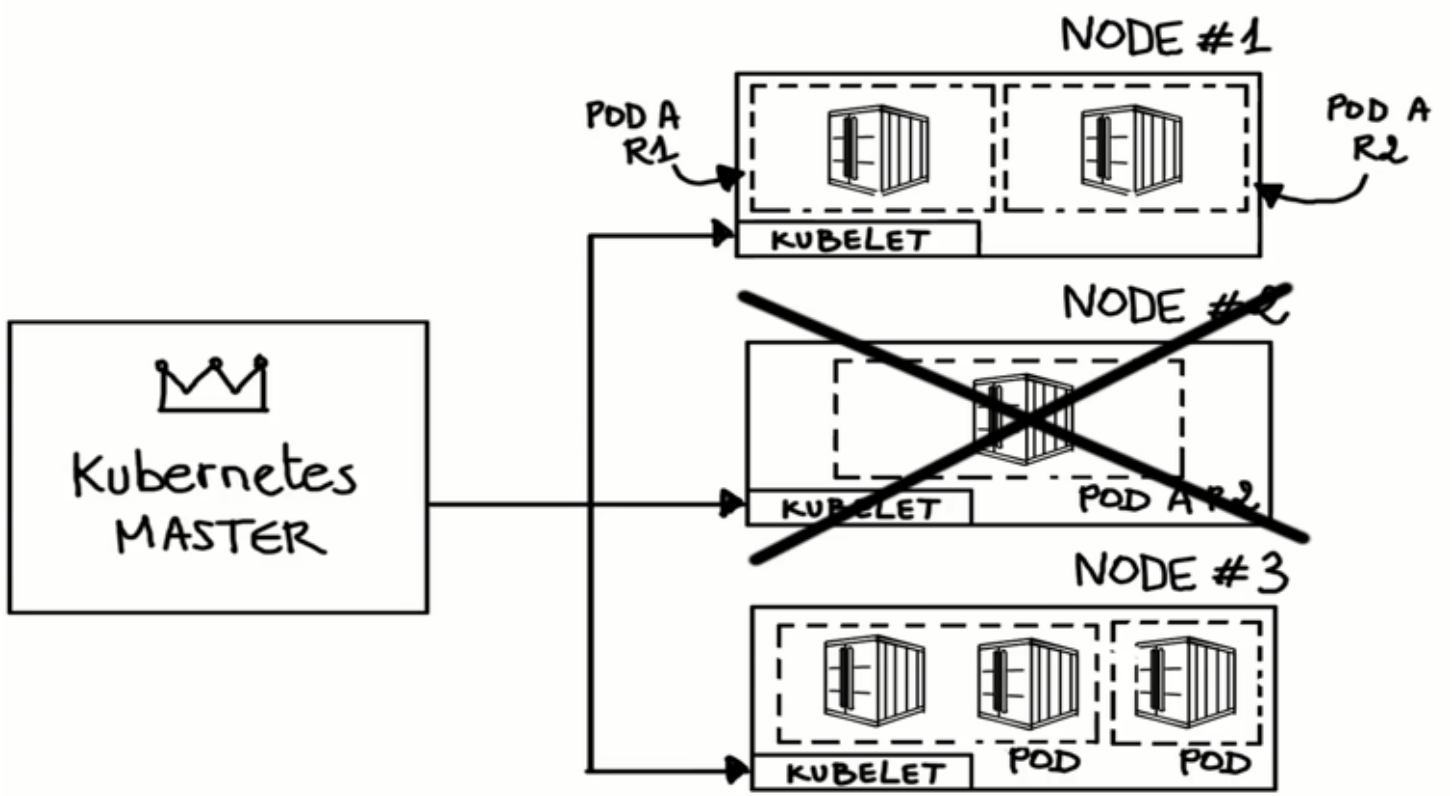


Architecture de Kubernetes

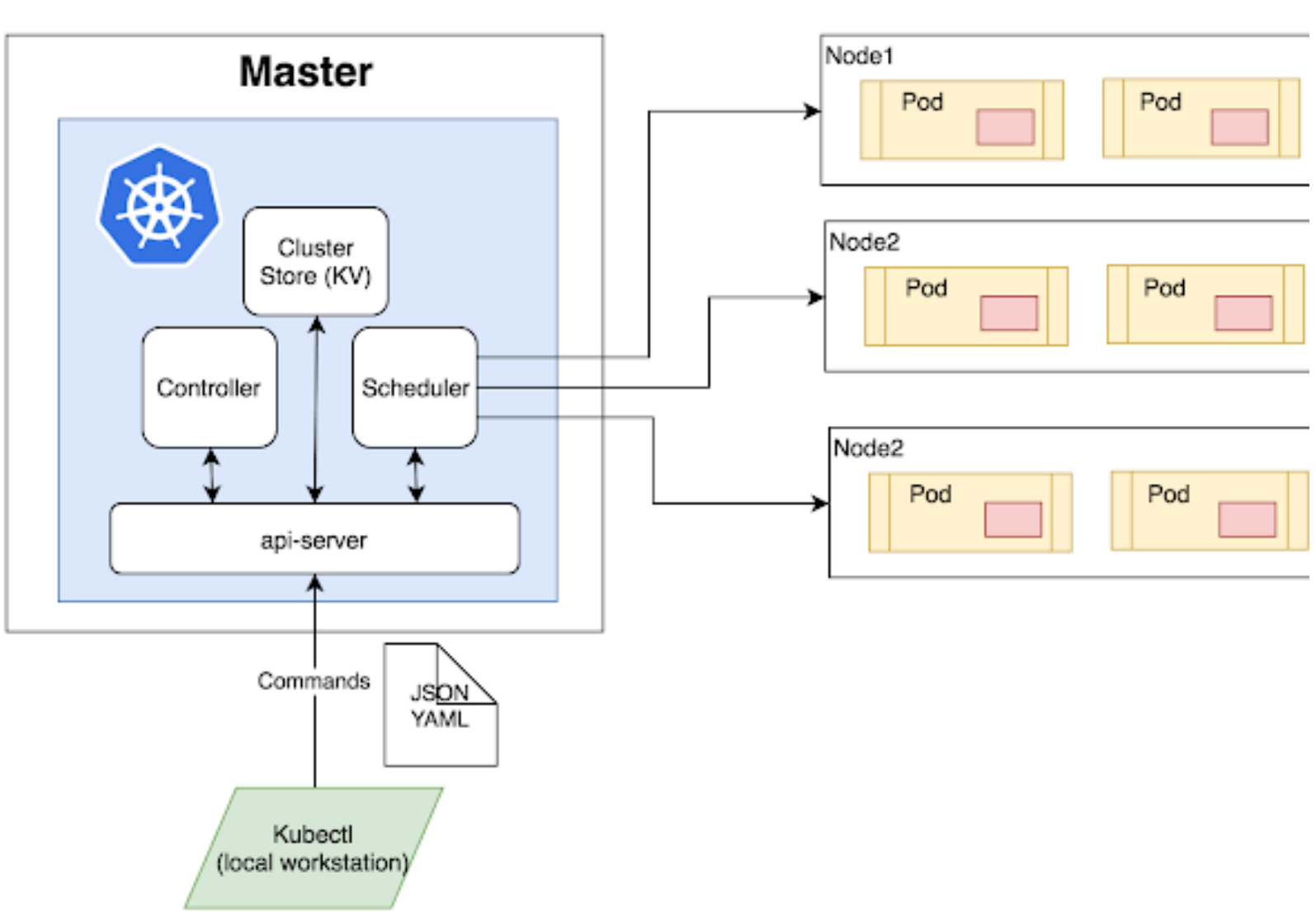
- Le master gère l'utilisation des ressources sur chaque nœud afin de s'assurer que la charge de travail n'est pas en excès par rapport aux ressources disponibles.
- Un Pod représente un processus en cours d'exécution dans votre cluster. Un Pod encapsule un conteneur applicatif (ou, dans certains cas, plusieurs conteneurs), des ressources de stockage, une IP réseau unique, et des options qui contrôlent comment le ou les conteneurs doivent s'exécuter.
- Kubelet est responsable de l'état d'exécution de chaque nœud. Il prend en charge le démarrage, l'arrêt, et la maintenance des conteneurs d'applications (organisés en pods) dirigé par le plan de contrôle.



Architecture de Kubernetes



- Si un nœud tombe en panne, Kubernetes exécute ce POD dans un autre nœud disponible.
- Kubelet surveille l'état d'un pod et s'il n'est pas dans l'état voulu, le pod sera redéployé sur le même node. Le statut du node est relayé à intervalle de quelques secondes via des messages d'état vers le maître.
- Dès que le maître détecte un défaut sur un node, le Replication Controller voit ce changement d'état et lance les pods sur d'autres hôtes en bonne santé.



Architecture de Kubernetes