Kubernetes, le Windows des conteneurs

23 décembre 2022

claude@lemarson.com
https://www.lemarson.com

Sommaire

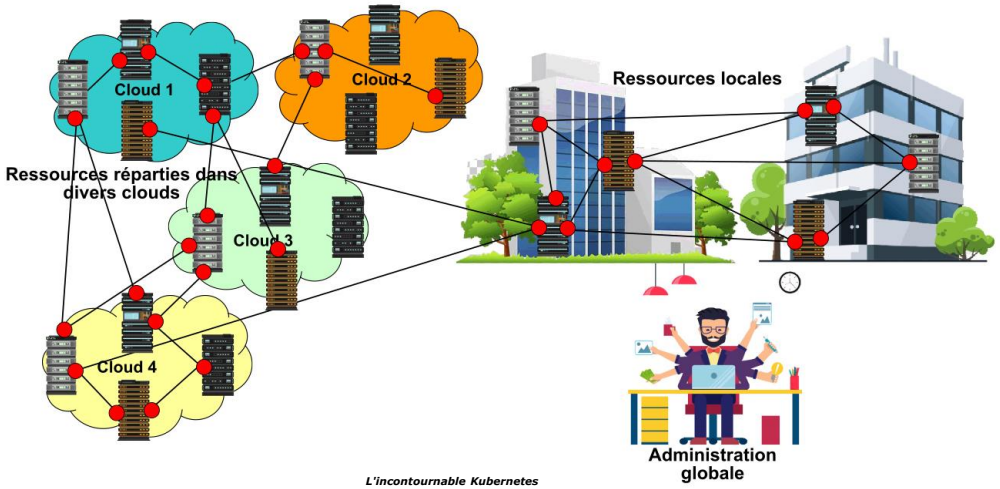
Kubernetes, le Windows des conteneurs

- *Quel est l'enjeu : la gestion des charges réparties.*
- *Les sphères d'influence : CNCF, OCI*
- *L'approche des conteneurs vs machines virtuelles*
- *Le rendez-vous raté de Docker*
- *L'importance et le rôle de Kubernetes*
- *Ce que fait Kubernetes : serveur et client*
- *Ce que n'impose pas Kubernetes*
- *Avantages et inconvénients*
- *Quelques plates-formes de distribution*
- *Quelle stratégie adopter : Cloud et locale*

Le marché global de Kubernetes (GlobeNewswire) passe de 1,74 G$ en 2021 à 5,47 G$ en 2028, avec un CAGR de 17,70 %. 320 M$ en 2022 à 650 M$ en 2029 pour Swarm. L'écart augmente.

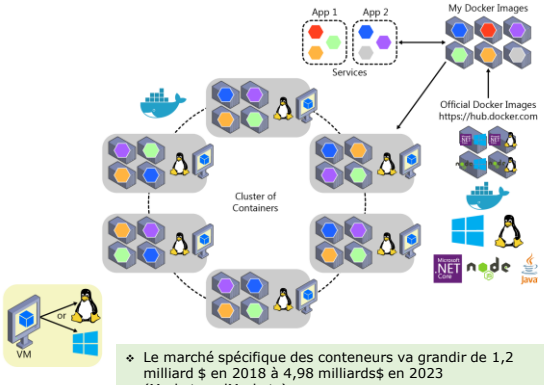
L'enjeu : la gestion des charges réparties

- L'objectif est clair : installer et administrer des charges de travail réparties dans un cluster.
- Les besoins essentiels :
  - Maintien de la continuité de service.
  - Répartition possible entre ressources locales et cloud.
  - Réorganisation en temps réel et de manière transparente de la distribution des ressources.
  - Swarm et Kubernetes effectuent ce travail



Une architecture qui remonte loin

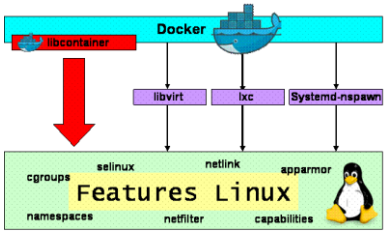
- Plate-forme de diffusion d'applications destinée aux systèmes distribués en clusters.
- Environnement d'exécution complet : une application avec ses dépendances, ses bibliothèques et fichiers binaires, ainsi que les fichiers de configuration nécessaires pour l'exécuter, regroupés dans un seul package, mais **PAS** l'OS.
- Technologie qui a plus de … 40 ans :
  - Commande Unix **chroot** apparue en 1979 : changement de répertoire du système de fichier racine Unix.
  - Pour faire des tests, implémenter des process critiques, sans détériorer le système hôte.
  - C'est le début de la séparation des environnements pour chaque process, qu'il faut recréer localement.
- Les **prisons** FreeBSD en 2000 : plusieurs environnements administrables indépendamment.
- Surtout les "**Solaris Containers**" de Sun en 2004 :
  - Technologie issue de la virtualisation d'OS des x86 et Sparc.
  - Séparation en "zones" : l'OS est virtualisé et isolé des zones.
- **Process containers** en 2006 (Google), devenus cgroups.
- **LXC** (LinuX Containers) en 2008 : implémentation des cgroups et namespaces de Linux, fonctionne sur un seul noyau Linux.
- 2017 : Adoption de Kubernetes par la CNCF (Cloud Native Computing Foundation).

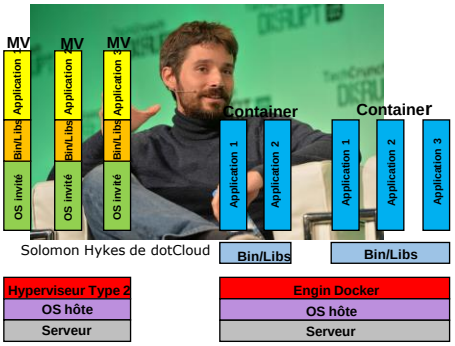


- Le marché spécifique des conteneurs va grandir de 1,2 milliard $ en 2018 à 4,98 milliards$ en 2023 (MarketsandMarkets).
- Autre source : 8,20 milliards $ en 2025 (CAGR de 31,8 % de 2018 à 2025) (Allied Market).
- Virtualisation d'OS : 62,91 milliards$ en 2023 (Technavio), CAGR de 30%.
- La virtualisation des OS serveurs représente un marché plus de 12 fois supérieur à celui des conteneurs.
- Ne pas opposer conteneurs et virtualisation : ils se complètent.

L'architecture des conteneurs

- Le principe du container est de construire une boîte applicative, dans laquelle il y aura l'application et un minimum de « features » système dont elle aura besoin pour s'exécuter.
  - Objectif : diffuser les applications par une simple commande, sans la lourdeur de la virtualisation.
  - Victoire du « couper/coller » container contre « installer/configurer/exécuter » de la virtualisation.
  - L'idée est de construire un repositiory d'images (les build exécutables), à transférer à la demande sur des machines distantes ou dans un Cloud.
- Une architecture conteneur est fondée sur trois grandes couches au-dessus du système d'exploitation : l'image applicative, le container lui-même, c'est-à-dire l'ensemble des services rendus aux applications à l'exécution (le « run time ») et la couche d'administration, telle que Kubernetes ou Swarm.





L'intérêt des conteneurs ?

- Partage des ressources sous-jacentes : OS, drivers, mémoire, CPU… Ce qui peut poser des problèmes de sécurité
- Découplage par rapport aux infrastructures, ce qui les rend portables et plus faciles à administrer.
- La création des images exécutables est plus facile et agile qu'avec une MV.
- Développement, intégration et déploiement continus.
- Capacité d'observabilité et de surveillance des applications.
- La technologie ne perturbe pas les environnements de production : il est facile de construire en continu des instances d'images, pour les tests, la pré-production, avec des builds identiques entre machines locales et Cloud.
- Grande portabilité : fonctionnent avec de nombreuses distributions Linux : Ubuntu, RHEL… en local ou dans le Cloud.
- Très adapté aux architectures de micro-services, avec un faible couplage entre les ressources et les infrastructures sous-jacentes.
- Adéquation avec les méthodes agiles : déploiement en préproduction et production des histoires.

Images et conteneurs... Docker s'essouffle

- Ne pas confondre le conteneur et l'image.
- Marché très diversifié : OS, build d'images, runtimes, utilitaires
- Forte érosion de Docker.
- Container Linux (anciennement CoreOS) était un système. d'exploitation léger open source basé sur le noyau Linux et conçu pour fournir une infrastructure aux déploiements en cluster, tout en se concentrant sur l'automatisation, la facilité de déploiement des applications, la sécurité, la fiabilité et l'évolutivité. Arrêté, Fedora CoreOS ou RHEL pour le remplacer.
- **CoreOS rkt** (prononcer rocket), racheté par Red Hat :
  - Très proche conceptuellement de Kubernetes (les rknetes s'installent facilement).
  - Au départ 2 types d'images : Docker et appc.
  - Compatible : idéal pour la portabilité dans les Clouds publics.
  - A l'origine, pas conforme OCI, mais corrigé maintenant (Rklet)..
- **Mesos Containerizer :**
  - Apache, en 2018.
  - Compatible OCI, deux formats d'images Docker et appc.
  - Un usage répandu : constructions Big Data avec Spark et Flink.
  - Une image Mesos ne peut pas fonctionner seule, elle doit s'appuyer sur le framework Mesos.
- **LXC Container** : version modernisée de LXC :
  - 3 modules : lxc le runtime, lxd très important, un démon écrit en Go qui gère les containers et images avec nouvelle UI et gestion des containers et lxfuse, la gestion de fichier.

Containerd.
Runtime simple du CNCF, longtemps celui de DOcker (runC). Gère le cycle de vie des conteneurs : création, exécution, destruction supervision. Intègre de nombreux outils : runC, Kubernetes AWS et Azure. Existe en daemon sous Windows.

LXD
Importe un OS Linux complet dans un conteneur. Gère aussi des MV. Peut traiter des clusters complexes avec MV **et** conteneurs.

Azure Container Registry
Permet d'accéder au registre privé Docker dans Azure. Mode lignes de commandes Docker. Dispositions de sécurité, dont scanner de vulnérabilités. Une sorte de catalogue d'mages administrables par Swarm ou Kubernetes.

buildah
Interface ligne de coimmandes pour conteneurs et images OCI (fabrication de dockerfiles sans le runtime). Pour ajouter du contenu à une image.

podman
Conteneur Linux qui implémente la librairie libpod de gestion des conteneurs. Mise à jour d'image. Peut porter les conteneurs sous Windows et Mac. Conforme OCI (à l'origine projet kpod dans le projet CRI-O).

BuildKit
Build d'images Docker, autonome ou intégré au Build de Docker. Très performant dans la construction.

RUNC
Outil en ligne de commande pour diffuser et exécuter des images conteneurs sous Linux. Fonctions de bas niveau, réservé aux spécialistes. Version autonome distincte de Docker. Runtime léger et portable, proche de Containerd sans le support Windows.

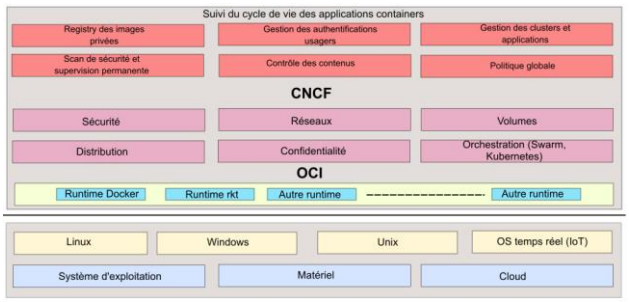
Kaniko
Constructeur d'images conteneurs pour Kubernetes à partir de fichiers Dockerfile. Utilisable avec le "Google Container Builder", sans avoir besoin de mode privilégié root.

rkt
Alternative à Docker. Conteneur adapté au Cloud. Natif pod. S'intègre avec d'autres systèmes. N'est plus en développement chez GitHub.

Les sphères d'influence des conteneurs

CNCF : l'Open Source pour le Cloud

- CNCF (Cloud Native Computing Foundation) créé en 2015 par 18 compagnies pour promouvoir les projets Open Source destinés au Cloud. Les deux premiers ont été Kubernetes de Google et Promotheus, pour le « monitoring ».
- Les deux plus grands acteurs du Cloud, AWS et Microsoft ont rejoint la communauté CNCF en tant que membres « platinium » et CNCF regroupe maintenant tous les acteurs qui comptent dans le monde du Cloud.
- CNCF tourne toujours autour de Kubernetes, qui constitue son projet phare.
- CNCF est considéré comme le bras armé de Google dans le monde des containers et la reconnaissance d'IBM, Microsoft , AWS est avant tout celle de Kubernetes, pas du reste…



Les sphères d'influence des containers

OCI pour les containers

- OCI (Open Container Initiative) est l'autre structure phare. Fondée également en 2015, pour élaborer des spécifications autour des containers. Elle doit beaucoup à Docker, qui en a été à l'origine.
- OCI a succédé à l' « Open Container Project », lui-même lancé sous l'impulsion de Docker.
- Le forum OCI a proposé une norme, la 1.0, avec deux composants clés : les formats d'images et les spécifications du « runtime », pour l'exécution. Il manquait encore beaucoup d'éléments, dont certains essentiels comme la distribution des containers ou la méthode pour solliciter une image, hébergée dans un annuaire.
- L'un des points forts d'OCI, est le consensus qui s'est dessiné autour de son nom, la plupart des acteurs qui comptent étant représentés : Docker, Dell, Cloud Foundry, Core OS, Google, IBM, Huawei, Intel, Mesosphere, Microsoft, Oracle, Red Hat, VM Ware, etc. Souvent des concurrents de Docker.
- Le forum OCI a créé un cadre dans lequel peuvent s'insérer différentes formes de containers, qu'elles viennent de Docker ou d'un autre horizon.

OPEN CONTAINER INITIATIVE

CLOUD NATIVE COMPUTING FOUNDATION

Le rendez-vous raté de Docker

- **Swarm (essaim)**
  - Nodes
  - Services et tâches
  - Load balancers
- Open Source, intégré à Docker.
- Avantages
  - Simple à installer, peu encombrant.
  - Le "load balancing" est natif, mais limité.
  - Swarm est accessible par le jeu de commandes CLI (Command Line Interface).
  - Fonctionne avec l'outil Compose :
    - Pour définir et exécuter des applications multi-containers (utilise des fichiers YAML de configuration des services applicatifs).
    - Divers environnements : production, staging, développement, tests.
    - Commandes start, stop, reconfiguration de services.
  - Dispose de sa propre API.
- Swarm ne s'est pas imposé : même problème que pour Docker.
  - Le système est plus fermé.
  - Nécessite des licences, sauf pour certains clients : petites structures, enseignement…
  - La taille de Docker n'est pas crédible.
  - Solomon Hikes a quitté le navire.
  - Aujourd'hui tout devient Open Source, surtout les architectures.
  - L'avenir est aux communautés d'intérêts : chacun participe et bénéficie des retombées, à hauteur de son engagement.

Docker a réalisé une dernière levée de fonds (serie C) de 105 M$ menée par Bain Capital Ventures. L'éditeur se concentre sur le service aux développeurs après la cession de son activité Entreprise à Mirantis en 2019.

Comment se situe Kubernetes

- Couche d'administration essentielle et plébiscitée...
- Google a cédé son produit à la communauté CNCF (Cloud Native Computing Foundation).
- A l'origine, projet interne (Borg).
- Kubernetes : ensemble de composants (services), qui réalisent des opérations de supervision des containers implantés dans les nœuds d'un cluster, services qui se répartissent en deux familles, serveurs centraux et nœuds du cluster.
- Serveurs centraux : outils pour déployer, maintenir et mettre à l'échelle les applications portées par les containers (adaptation au nombre d'usagers).
- L'entité de base gérée par Kubernetes est le « pod », constituée d'un ou plusieurs containers, qui tous sont situés dans le même nœud du cluster, identifié par son adresse IP.
- L'ensemble des applications d'un pod communiquent par divers moyens, tels qu'un sémaphore ou une mémoire partagée.
- Kubernetes est un cluster constitué d'îlots applicatifs, les pods, qui ont pour finalité d'exécuter un traitement et qui pourra faire appel à d'autres containers.
- Les outils Kubernetes sont eux-mêmes empaquetés dans des containers, que ce soit sur les serveurs ou les nœuds.

Kubernetes, prononcez K8s (Il y a 8 lettres entre la première lettre "K" et la dernière "s"...), permet d'automatiser le déploiement, la montée en charge et la mise en œuvre de conteneurs d'applications sur des clusters de serveurs.
On peut aussi abréger en "kube"...

Les fonctionnalités de Kubernetes

- **Découverte de service et équilibrage de charge**

Kubernetes alloue aux pods leur propres adresses IP, des noms DNS uniques et peut équilibrer leur charge.

- **Optimisation automatique ("bin packing")**

Kubernetes planifie automatiquement les containers en fonction de leurs besoins et contraintes.

- **Déploiement de fonctionnalités nouvelles et retours (Rollouts et rollbacks) automatiques**

Kubernetes déploie les mises à jour des applications et les changements de configuration sans interruption.

- **Orchestration du stockage**

Kubernetes génère automatiquement les espaces de stockage dans les containers via des Software-Defined Storage (SDS), quelle que soit les ressources.

- **Gestion du routage**

Kubernetes route le trafic à destination des différents services, quelle que soit la topologie du cluster. En particulier hybride.

- **Auto-médication**

Kubernetes remplace et replanifie automatiquement les containers en défaillance. Il tue et redémarre les containers qui ne répondent pas à des contrôles de santé ("health checks"). Il empêche que le trafic soit routé sur des containers non disponibles.

- **Mise à l'échelle horizontale ("Scaling horizontal")**

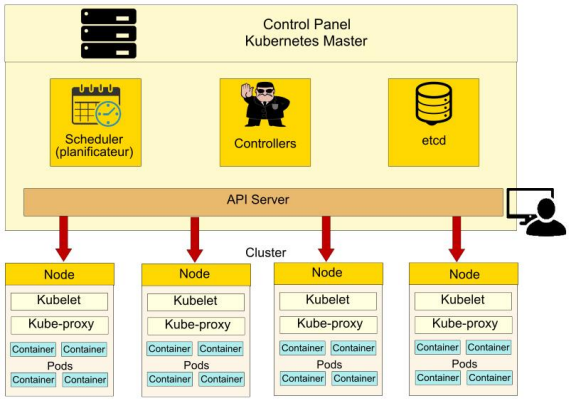
Les applications peuvent être mises à l'échelle manuellement ou automatiquement selon des métriques de type CPU/Mémoire ou personnalisées.

- **Gestion des données sensibles**

Kubernetes gère séparément les données sensibles, telles que les authentifications, les tokens Oauth ou les clés SSH, de manière à ne pas avoir à "rebuilder" les applications, quand il y a des modifications à prendre en compte.

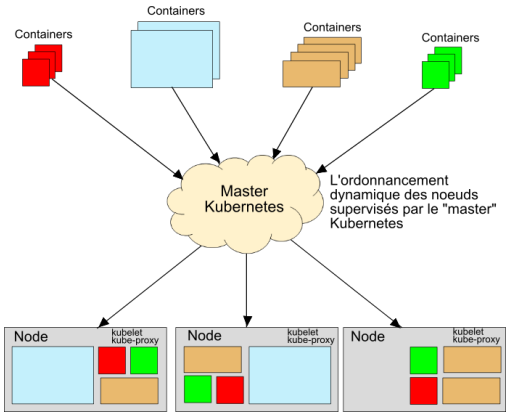
- **Exécution de charges lourdes**

Kubernetes supporte l'exécution de travaux lourds et de longue.



Kubernetes côté serveur

- Les composants à installer sont ceux du « control panel », qui pilotent l'activité du cluster, gèrent l'ordonnancement des opérations, la réponse aux évènements, le démarrage d'un nouveau pod, etc. C'est le cœur du système, qu'il faut installer, sachant que l'on pourra configurer plusieurs serveurs maîtres, autant pour la sécurité que pour les performances, Kubernetes pouvant être « load-balanced » entre plusieurs serveurs.
  - **kube-controller-manager** : c'est lui qui exécute les contrôleurs, le node-controller, qui maintient l'activité des nœuds et réagit quand l'un d'eux tombe en panne, le replication-controller qui maintient le nombre correct de pods pour chaque contrôleur, le endpoints-controller qui diffuse les objets vers les nœuds (pods…), service account & token controller, etc. C'est la tour de contrôle de l'activité Kubernetes.
  - **kube-scheduler** : ce service surveille les pods nouvellement créés et les affecte aux nœuds, en se basant sur les ressources nécessaires, les contraintes matérielles et logicielles, la localisation des données, etc.
  - **kube-apiserver** : c'est le front-end de Kubernetes, celui qui expose l'API à l'extérieur.
  - **etcd**, la partie stockage, à la fois pour les données manipulées (backup) et pour le maintien des configurations de chacun des containers. C'est une sorte d'annuaire clés-valeurs.



Nombreuses fonctions d'administration :
- Gestionnaire de packages HELM, pour installer et faire vivre des applications complexes.
- "Feature gates" pour connecter (on) ou déconnecter (off) un nœud.
- "Cluster federation" : pour agréger plusieurs clusters K8s dans un seul cluster logique.
- "Customer scheduler" : pour personnaliser la gestion de certains pods.
- "Sidecar" : pour placer un proxy de conteneur dans un pod.

Kubernetes côté client

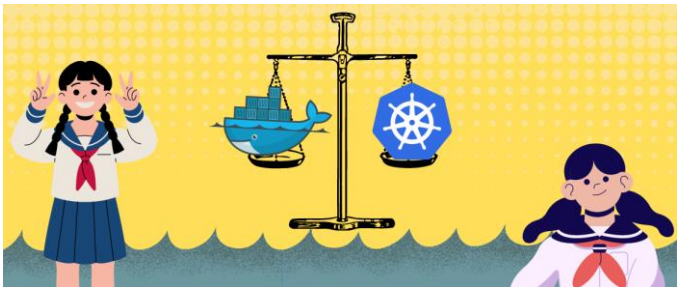
- Côté nœuds, d'autres services doivent être installés, chargés de la supervision des pods applicatifs, à qui ils vont attribuer les ressources nécessaires dans la plate-forme d'exécution locale (le runtime).
- **kubelet** est l'agent qui s'assure que les containers s'exécutent dans un pod et se fonde pour cela sur des PodSpecs. kubelet ne gère que les containers créés par Kubernetes.
- **kube-proxy** : l'abstraction du réseau.
- **container runtime** : une pièce essentielle responsable de l'exécution des containers, qui supporte plusieurs runtimes et formats d'images, containerid, cri-o (container léger), rklet et n'importe quelle implémentation de Kubernetes CRI (Container Runtime Interface).
- containerd a été intronisé au début 2019 dans CNCF.
- Kubernetes exploite aussi des addons, des services et pods, qui implémentent indépendamment certaines fonctions.

Ce que n'impose pas Kubernetes

- L'outil est capable de supporter des applications très diversifiées : avec ou sans état, gros volumes de données, charge de traitement élevée. Si une application peut s'exécuter dans un container, elle est prise en charge par Kubernetes.
- Il n'interfère pas sur les phases de développement, ni sur les stratégies de livraison et de déploiement continus. Il s'adapte à ce qui est en vigueur dans l'entreprise.
- Il ne fournit pas de services applicatifs et ne se substitue pas au middleware : bases de données, frameworks de traitement Big Data (Spark).
- Il ne concurrence pas les "utilitaires système" actifs. S'ils existent, Kubernetes les prend en compte.
- Il n'impose pas de solutions de logging, de monitoring ou d'émission d'alertes. Il fournit par contre l'infrastructure pour lui permettre de recueillir les informations utiles.
- Il n'oblige pas à se servir d'un langage de configuration, tel que JSONnet. Il fournit par contre une API déclarative.
- Kubernetes n'est pas un outil d'orchestration classique, qui ordonnance dans le temps les traitements élémentaires de services. Il fonctionne différemment, sans contrôle centralisé, avec un ensemble de processus répartis qui s'assurent en permanence de l'état ponctuel des constituants.

Avantages et inconvénients

- K8s est destiné aux grosses configurations.
- Swarm reste crédible pour de petites structures.
- Très adapté à la gestion de grosses charges de travail distribuées et complexes : c'est sa principale caractéristique.
- Il s'appuie sur une large communauté Open Source, qui bénéficie de l'appui de Google.
- La communauté lui garantit un support de qualité et le recours potentiel à des conseils sur des solutions spécifiques.
- Très gros avantage : il est proposé en standard par les principaux fournisseurs de Cloud, Google Cloud Platform, Microsoft Azure, IBM Cloud, AWS, Alibaba, Baidu, OpenStack (depuis Ocata 2017).
- Sa capacité d'extension en mode horizontal est un gros atout.
- Nativement, il dispose de fonctionnalités très étendues, mais il peut aussi s'appuyer sur une large panoplie de produits tiers, que les usagers peuvent intégrer.



Mais tout n'est pas parfait

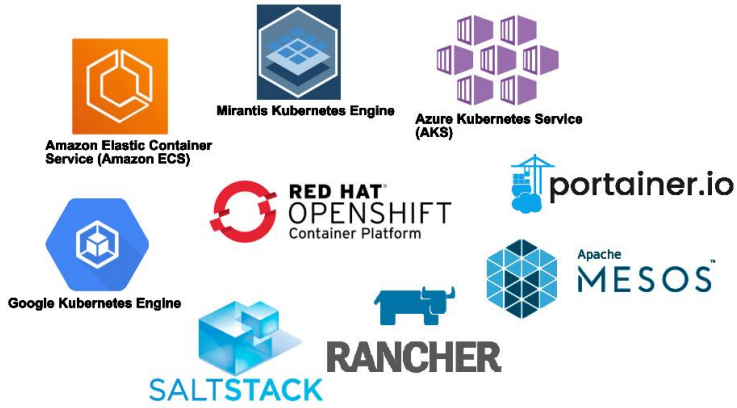
- L'apprentissage de Kubernetes n'est pas "un long fleuve tranquille". C'est un monde en soi et nécessite du temps et de l'expérience.
- Le processus d'installation est complexe, difficilement maîtrisable par les débutants.
- Kubernetes est Open Source, avec une forte communauté qui intervient parfois de manière "erratique". Il peut y avoir des dérives non contrôlées…
- Pour les environnements restreints, Kubernetes est trop lourd et compliqué à mettre en œuvre.

Les plates-formes de distribution Kubernetes

Kubernetes a-t-il encore des concurrents

(pas vraiment)

- Il existe un marché concurrentiel réduit autour de l'administration des conteneurs et Kubernetes n'est pas seul.
- 3 types de solutions :
  - Des boîtes à outils qui traitent une partie des fonctions d'administration.
  - Des plates-formes universelles réellement concurrentes de Kubernetes.
  - Des implémentations dédiées : pour Azure, Google, etc.



Quelle stratégie adopter

- Le choix ne se pose plus vraiment en dehors des petites configurations autonomes.
- La concurrence existe, mais reste concentrée sur des plates-formes spécifiques : Google, Azure.
- Mirantis doit se forger une crédibilité auprès des entreprises.
- Fonctionnellement, K8s est le plus complet.
- La version 1.26 de décembre 2022 en témoigne :
  - Assouplissement de la politique multi protocoles de "load balancing"
  - Améliorations de 2 composants phare de kubelet : le CPUManager (gestion des CPU pour les pods) et le DeviceManager (nœuds spécifiques)
  - Nouveau registry.k8s.io qui remplace k8s.gcr.io : ouverture à d'autres fournisseurs que Google (Amazon…).
  - Contrôle des tâches sans persistance des pods (avant il fallait les maintenir).
- Question : quel risque prenons nous avec une politique monomarque K8s :
  - Kubernetes ne se substitue pas aux images conteneurs : il prend ce qu'on lui propose.
  - L'image n'est pas l'élément le plus important : fichier exécutable qui va se diversifier, même si l'hétérogénéité n'est pas recommandée.
  - L'admin est essentielle, sans laquelle rien ne peut fonctionner : il est dangereux de dépendre de Google, même à travers CNCF.
  - Le retrait de Google de CNCF est une attitude de façade.
- Même problème qu'avec l'IBM des grandes années : on sait que l'on est pris en otages, mais ça fonctionne…