| * Orchestration : (9)   + Définition   + Principe de fonctionnement   + Avantages et inconvénients   + Outils     - Etude comparative     - Kubernetes       * Définition       * Principe de fonctionnement       * Modules       * Avantages et inconvénients |
| --- |

DRAFT:

* What is Orchestration ?
* What are Microservices?
* Orchestration vs Choreography ?
* What is Kubernetes?
* Immutable VS mutable infrastructure
* Déclarative configuration
* Online self-healing systems

Orchestration: 9pg

**Introduction**:

* containers respond to the concerns of the development team mainly(SIL), but the operations team(SIQ) has other important concerns outside the containers themselves : deployment, scalability, networking, insights.

containers solve many dev process related issues like dependencies and compatibility… but now we need to think about how we’re going to deploy those containers? How are we going to scale the services deployed on those containers when we want to (selon le besoin), using available resources (scheduling, partage de travail, and handling failures)? and we need to handle communication between the containers, with the outside world ? We also want to supervise our system of containers.

(mentioned in this video:<https://www.youtube.com/watch?v=kBF6Bvth0zw>)

* the need for managing, maintaining and scaling containers ?
* the different approaches to managing containers ?
* Mention that orchestration tools are one of the most widely methods for managing containers

La conteneurisation est née d'un besoin, et répondent aux préoccupation des équipes de développement, mais de l'utilisation croissantes de ces derniers et de la multiplication des conteneurs par services (notion de micro-services) , d'autres problèmes ont vu le jour : la gestions des différents conteneurs, entre déploiement, évolutivité, mise en réseaux...

Effectivement, les conteneurs résolvent d'eux-mêmes de nombreux problèmes liés au processus de développement, tels que les dépendances et la compatibilité. De ce fait, la réflexion se porte maintenant sur la manière dont les conteneurs seront déployés. Comment faire évoluer les services déployés selon le besoin, comment s'assurer de la bonne utilisation des ressources fournies, et enfin, comment donner l'accès aux services hébergés.

**Définition**:

* On cherche une définition exacte avec référence. “ex: Orchestration is **the automated configuration, management, and coordination of computer systems, applications, and services**” (https://www.redhat.com/en/topics/automation/what-is-orchestration )
* (we add our little explanation)Container orchestration automates the deployment, management, scaling, and networking of containers

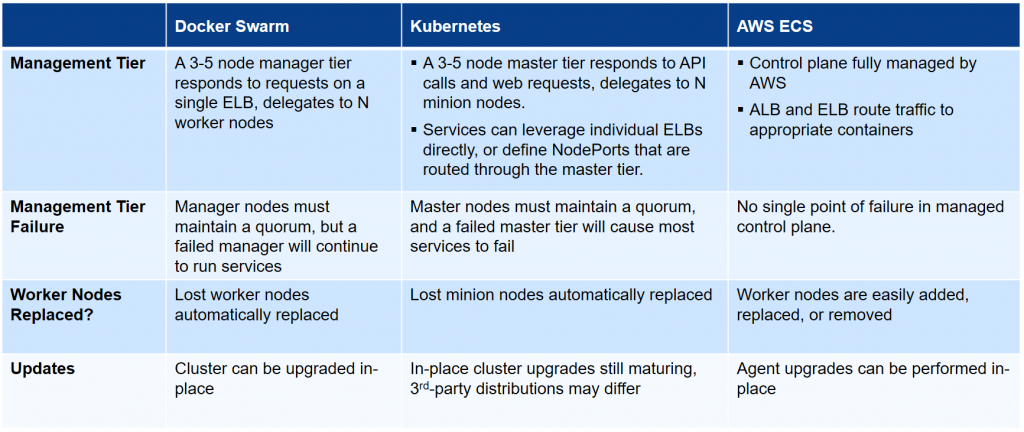
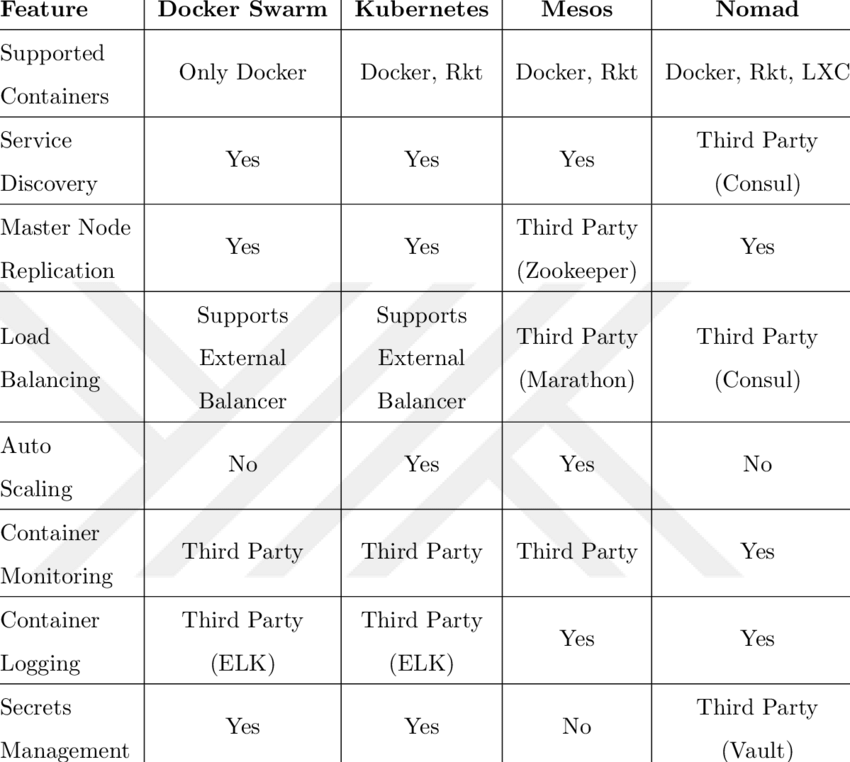
**Principe de fonctionnement**:

**Avantages et inconvénients**:

**Outils**:

Tableau comparatif:

* Nom
* company
* Open Source or not
* supported containers
* complexité
* max number of nodes
* min number of master nodes



* Kubernetes
* Docker Swarm
* Apache Mesos

How many nodes can Kubernetes handle?

5000 nodes

**Kubernetes is not optimised to handle more than 500 nodes per cluster** (though they claim to manage up to 5000 nodes). Therefore, you should be careful of how many nodes you add to your cluster, and optimise the size of the node with the required number.

Ainsi, bien que les conteneurs n'aient besoin que d'un runtime pour s'exécuter, de nombreuses organisations préfèrent les lancer avec un orchestrateur de conteneurs. Parmi ceux-ci, Kubernetes est de loin le plus populaire

Un orchestrateur de conteneurs gère les aspects opérationnels d'exécution d’applications dans des conteneurs - Comme la découverte de services, le redémarrage des conteneurs en cas d’arrêt, l'allocation des ressources systèmes et la gestion des groupes de conteneurs qui doivent être étroitement liés, pour n'en nommer que quelques-uns.

Kubernetes:

**Définition**:

Kubernetes est une plate-forme open-source extensible et portable pour la gestion de charges de travail (workloads) et de services conteneurisés. Elle favorise à la fois l'écriture de configuration déclarative (déclarative configuration) et l'automatisation. C'est un large écosystème en rapide expansion. Les services, le support et les outils Kubernetes sont largement disponibles.

* What does it do ?
* Immutability
* Déclarative configuration

Kubernetes provides you with:

* **Service discovery and load balancing** Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
* **Storage orchestration** Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
* **Automated rollouts and rollbacks** You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container. (Déclarative configuration)
* **Automatic bin packing** You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
* **Self-healing** Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
* **Secret and configuration management** Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

**Principe de fonctionnement**:

**Modules** / Terminologie :

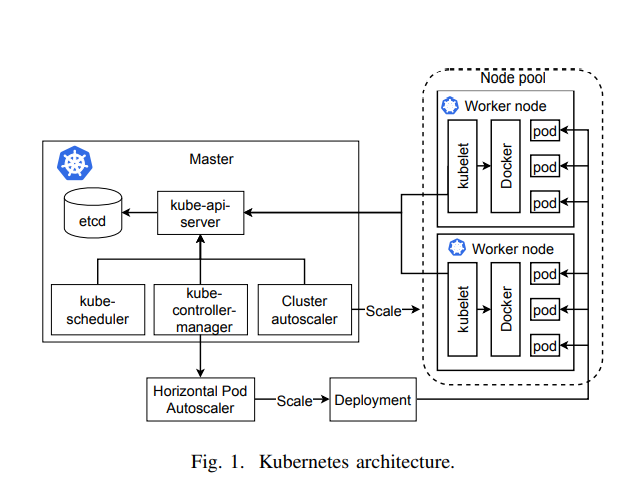
* Cluster : L’ensemble des machines sur lesquelles Kubernetes est déployé
* Nodes : Une unique machine du cluster, le cluster peut se composer d’un seul pod - cas plutôt rare dans la pratique–

Un noeud peut se voir attribuer des rôles :

* Master :
* worker: kublet
* Kubernetes Scheduler
* Services & Pods
* Namespaces & labels and label selectors
* Etcd

Base de données clé-valeur consistante et hautement disponible utilisée comme mémoire de sauvegarde pour toutes les données du cluster.

* Serveur API
* Controllers :
* Réplication
* DeamonSet
* JobController
* Ingress & Load Balancers
* API (REST) and API server
* kube-proxy
* cAdvisor
* Kubectl, yaml, and Configurations
* Network Policies
* Architecture:



**Avantages :**

* Velocity
* Scaling (of both software and teams)
* Abstracting your infrastructure
* Efficiency
* Application availability: Highly redundant cluster configurations reduce application downtime »
* Disaster recovery: Deployment across multiple regions and clouds ensures application survivability during a disaster or outage »
* Reduced latency: Rapid scale out capabilities maximize application performance and reduce latency during peak demand »
* Edge deployments: Clusters can be deployed at scale with a small infrastructure footprint common in many edge environments »
* CapEx reduction: Clusters can scale on-demand to meet capacity requirements as needed, thereby reducing the need for upfront investment in infrastructure »
* Choice and flexibility: Open-source container and cluster technology available across clouds helps avoid vendor lock-in and promote maximum choice and flexibility in best-of-breed deployments »
* Industry mandates: Clusters can be configured and managed to address specific security, privacy, and geopolitical data residency requirements, among others

**Inconvénients**:

* Centralized Approach: (effects scalability)
* Network Traffic and performance degradation
* High coupling (dependencies) , reliance on REST API
* complexité ?