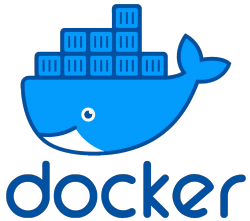


Formation kubernetes

25 - 27 Septembre 2023.

Philosophie Kubernetes et le mouvement cloud native

Conteneurs en production

Mais qu'en est-il de la gestion d'une flotte de plus de 10 conteneurs en production ?

Dans un environnement de production, nous devons :

- Vérifier l'état des conteneurs pour être toujours disponible.
- Autoriser la découverte des services.
- Expédiez facilement la nouvelle version de notre logiciel (mises à jour progressives, déploiements Bleu/Vert, restaurations).
- Faire évoluer les services de manière indépendante.

Orchestrateurs de conteneurs

La polyvalence d'un conteneur est une bénédiction mais aussi un gâchis :

- Nous déployons des tonnes et des tonnes de conteneurs
- L'ajout de nouvelles fonctionnalités et de nouveaux types de ressources peut rendre l'architecture très complexe

Comme dit précédemment, gérer l'architecture des conteneurs en production devient difficile (ex : redémarrer une flotte de conteneurs via la commande CLI peut prendre beaucoup de temps)

Les orchestrateurs automatisent le provisionnement et la gestion des infrastructures conteneurisées : démarrer et arrêter les conteneurs, assurer la découverte de services pour les conteneurs, équilibrer les charges sur l'ensemble du cluster, gérer les secrets, surveiller les conteneurs santé et les redémarrer lorsqu'ils tombent, etc.

Problématique

- Il est relativement facile aujourd'hui, de déployer un seul conteneur vers une machine virtuelle et/ou physique. Cependant, la difficulté s'accroît dès lors que l'on souhaite déployer des dizaines voire des centaines de conteneurs logiciels, en garantissant évolutivité, portabilité et disponibilité.

Problématique.

- Kubernetes amène une solution à cette problématique. Il aide à automatiser le déploiement, la gestion et l'évolutivité des applications conteneurisées.

- Kubernetes le fait en créant un cluster composé de machines physiques et/ou virtuelles, qui sont capables d'exécuter des applications conteneurisées.

Historique

Kubernetes est un logiciel développé originellement par Google et basé sur une dizaine d'années d'expérience de déploiement d'applications énormes (distribuées) sur des clusters de machines.

Son ancêtre est l'orchestrateur **borg** utilisé par Google dans les années **2000**.

L'écosystème logiciel de Kubernetes s'est développée autour la Cloud Native Computing Foundation

Kubernetes à la croisée des chemins.

Kubernetes se trouve au coeur de trois transformations profondes techniques, humaines et économiques de l'informatique:

- **Le cloud**
- **La conteneurisation logicielle**
- **Le mouvement DevOps**

Il est un des projets qui symbolise et supporte techniquement ces transformations. D'où son omniprésence dans les discussions informatiques actuellement.

Un mot sur le cloud computing.

D'après le **NIST** américain *(National Institute of Standards and Technology)* le cloud est : *« un modèle permettant un accès facile et à la **demande**, via le réseau, à un pool **partagé** de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement mises à disposition des utilisateurs ou libérées avec un effort minimum d'administration de la part de l'entreprise ou du prestataire de service fournissant les dites ressources ».*

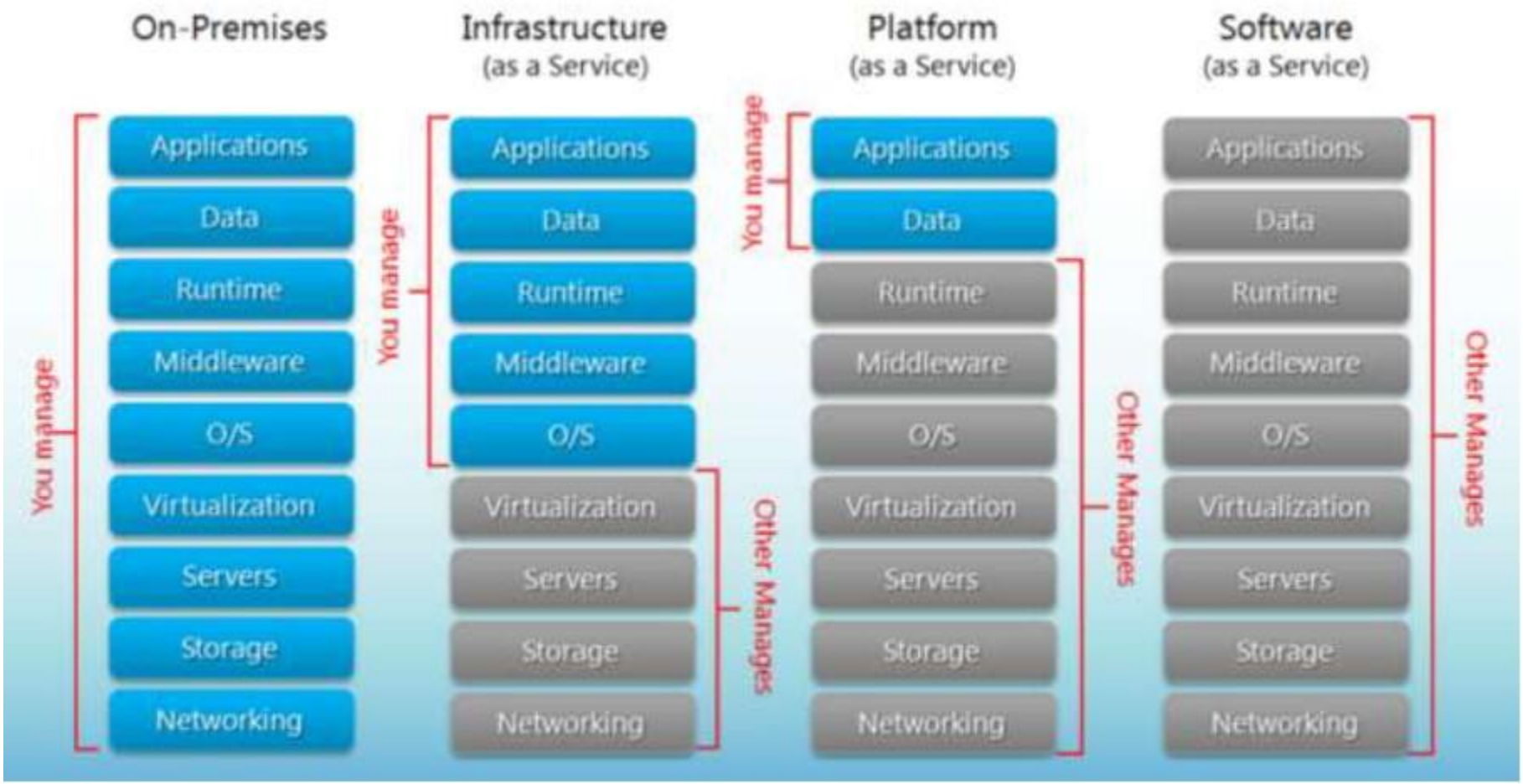
Un mot sur le cloud computing.

Pour organiser cela on définit trois niveaux à la fois techniques et économiques de l'informatique:

Software as a Service: location de services à travers internet pour les usagers finaux

Plateform as a Service: location d'un environnement d'exécution logiciel flexible à destination des développeurs

Infrastructure as a Service: location de ressources **"matérielles"** à la demande pour installer des logiciels sans avoir à maintenir un data center.


On-Premises
Infrastructure
(as a Service)
Platform
(as a Service)
Software
(as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
You manage
You manage
Other Manages
Other Manages
Other Manages

La conteneurisation

Les technologies de **conteneurisation** permettent donc de faire des boîtes isolées avec les logiciels pour apporter l'uniformisation du déploiement:

- Un façon standard de packager un logiciel (basée sur le)
- Cela permet d'assembler de grosses applications comme des legos
- Cela réduit la complexité grâce:
  - à l'intégration de toutes les dépendance déjà dans la boîte

Le mouvement DEVOPS

- Dépasser l'opposition culturelle et de métier entre les développeurs et les administrateurs système.
- Intégrer tout le monde dans une seule équipe et …
- Calquer les rythmes de travail sur l'organisation agile du développement logiciel
- Rapprocher techniquement la gestion de l'infrastructure du développement avec l'infrastructure as code.

Objectifs du DEVOPS

- **Rapidité (velocity) de déploiement logiciel** (organisation agile du développement et livraison jusqu'à plusieurs fois par jour) : Implique l'automatisation du déploiement et ce qu'on appelle la **CI/CD** c'est à dire une infrastructure de déploiement continu à partir de code.

- **Passage à l'échelle (horizontal scaling)** des logiciels et des équipes de développement (nécessaire pour les entreprises du cloud qui doivent servir pleins d'utilisateurs)

Objectifs du DEVOPS

Meilleure organisation des équipes :

- meilleure compréhension globale du logiciel et de son installation de production car le savoir est mieux partagé.

- organisation des équipes par thématique métier plutôt que par spécialité technique (l'équipe scale mieux).

Architecture logicielle optimale pour kubernetes

Kubernetes est très versatile et permet d'installer des logiciels traditionnels "monolithiques" (gros backends situés sur une seule machine).

Cependant aux vues des transformations humaines et techniques précédentes, l'organisation de **Kubernetes** prend vraiment sens pour le développement d'applications microservices:

- des applications avec de nombreux de "petits" services.
- **chaque service a des problématiques très limitées (gestion des factures = un logiciel qui fait que ça)**
- les services communiquent par le réseaux selon différents modes/API **(REST, gRPC, job queues, GraphQL)**

Quelques variantes de Kubernetes

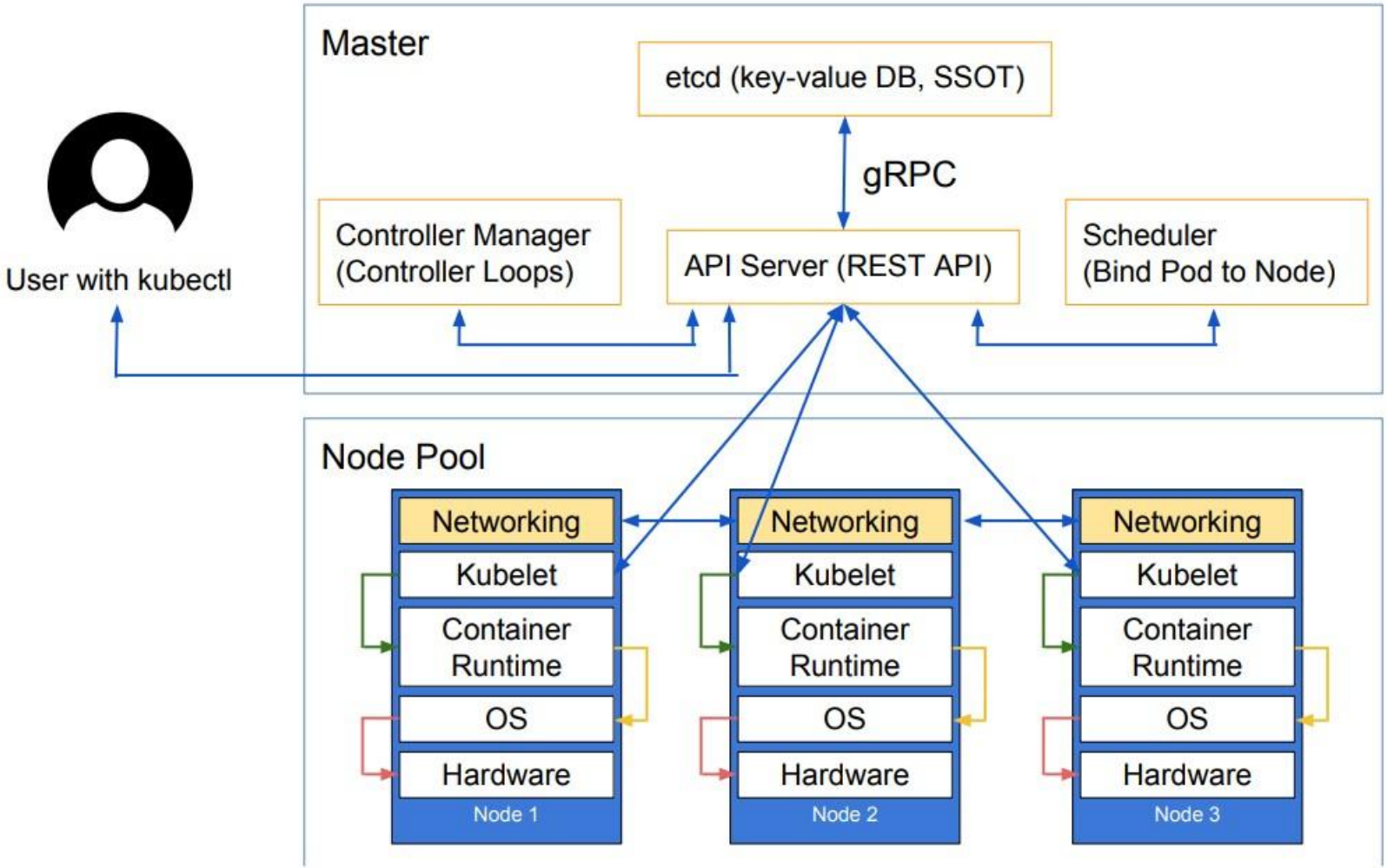
Variantes Kubernetes

Google Kubernetes Engine (GKE) (Google Cloud Plateform): L'écosystème Kubernetes développé par Google. Très populaire car très flexible tout en étant l'implémentation de référence de Kubernetes.

Azure Kubernetes Services (AKS) (Microsoft Azure): Un écosystème Kubernetes axé sur l'intégration avec les services du cloud Azure (stockage, registry, réseau, monitoring, services de calcul, loadbalancing, bases de données…).

Elastic Kubernetes Services (EKS) (Amazon Web Services): Un écosystème Kubernetes assez standard à la sauce Amazon axé sur l'intégration avec le cloud Amazon (la gestion de l'accès, des loadbalancers ou du scaling notamment, le stockage avec Amazon EBS, etc.).

Architecture Kubernetes


Master
etcd (key-value DB, SSOT)
gRPC
Controller Manager
(Controller Loops)
API Server (REST API)
Scheduler
(Bind Pod to Node)
User with kubectl
Node Pool
Networking
Kubelet
Container
Runtime
OS
Hardware
Node 1
Networking
Kubelet
Container
Runtime
OS
Hardware
Node 2
Networking
Kubelet
Container
Runtime
OS
Hardware
Node 3

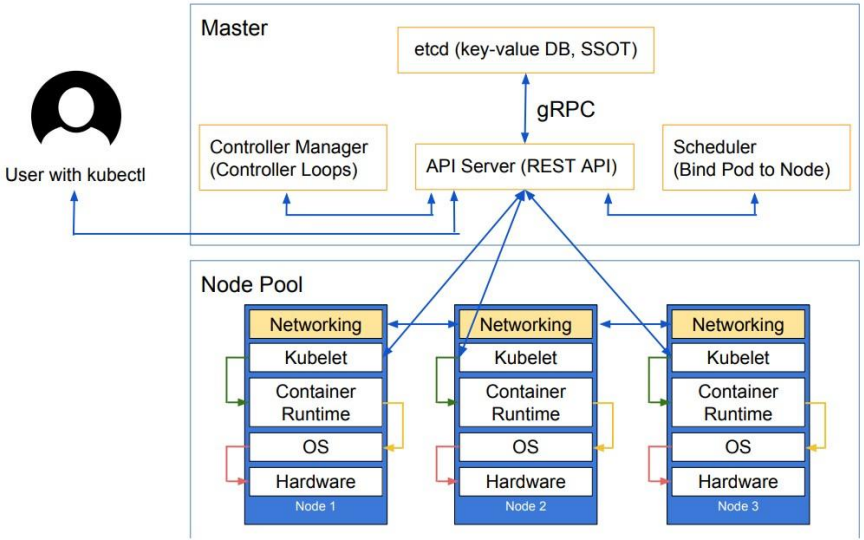
Objets kubernetes

Kubernetes propose une abstraction qui dissimule la complexité de la gestion d'un parc de conteneurs. Kubernetes s'appuie sur plusieurs "objets" :

- **Node** : Machine de travail du cluster Kubernetes.
- **Pod** : Plus petite unité de Kubernetes, permet d'encapsuler l'application conteneurisée.
- **Replicas** : Nombre d'instances d'un pod.
- **ReplicaSet** : Assure que les réplicats spécifiés sont actifs.
- **Deployment** : Définit l'état désiré et fournit des mises à jour des pods et replicaSet.
- **Service** : Niveau d'abstraction au-dessus du pod, il fournit une adresse IP et un nom DNS unique pour un ensemble de pods.
- **Endpoint** : Représente l'adresse IP et le port d'un service, il est automatiquement créé lors de la création d'un service avec les pods correspondants.

Node : Les composants.

- Kubernetes suit l'architecture maître-esclave (master-worker).
  - le master est chargé de la gestion du cluster
  - Le worker



Les composants du Master

Sur le node master, on distingue les composants suivants :

- **kube-apiserver** : point d'entrée exposant l'API HTTP Rest de kubernetes depuis le maître du cluster kubernetes.
- **kube-scheduler** : Répartit la charge de travail sur les nœuds du cluster en fonction des ressources disponibles et nécessaires.
- **Kube-controller-manager** : Ce composant est coeur du cluster, il centralise les informations sur le cluster en effectuant des actions correctives en cas de besoin.Il est constitué des diverses controleurs : controleur de réplication - contrôleur d'espaces noms… .

Les composants du master

- **Cloud-controller-manager** : effectue les mêmes actions que le kube-controller-manager mais pour les cloud provider (AWS - Azure - Google…).

- **etcd** : Il stocke les informations de configuration pouvant être utilisées par chacun des nœuds du cluster.

Les composants du worker

Sur chaque noeud worker nous distinguons les éléments suivants :

- **Kubelet** : Il s'exécute dans les nœuds chargé de relayer les informations au master. Il maintient en état les conteneurs d'un pod.
- **Kube-proxy** : Permet d'activer l'abstraction réseau du service kubernetes. Il expose les services vers l'extérieur.
- **Environnement d'exécution de conteneurs** : Il faut vous aussi une solution d'exécution d'applications conteneurisées,vous pouvez utiliser soit le moteur Docker mais Kubernetes prend également en charge l'utilisation de rkt comme moteur d'exécution du conteneur (que nous verrons pas dans ce cours).

Comment communiquer avec le cluster.

- **Kubeclt (ligne de commande)** : Outil en ligne de commande kubernetes. permet l'exécution de commande sur le cluster Kubernetes pour la création et la gestion d'objets. Le kubeClt favorise l'interaction avec l'API Kubernetes.

- **Dashboard (interface web)** : Dashboard est l'interface utilisateur Web officielle de Kubernetes.

MiniKube

- **Minikube** est un outil facilitant l'exécution locale de Kubernetes.
- Minikube exécute un cluster kubernetes à noeud unique (à la fois master et worker) dans une machine virtuelle.
- **Minikube** prend en charge les fonctionnalités Kubernetes suivantes:
  - *DNS*
  - *NodePorts*
  - *ConfigMaps et Secrets*
  - *Dashboards*
  - *Container Runtime:* ***Docker, CRI-O,*** *et* ***containerd***
  - *Activation de la CNI (Container Network Interface)*
  - *Ingress*

Cluster kubernetes.

Création d’un premier cluster avec MiniKube

Vous pouvez dès maintenant démarrer minikube afin de créer vore premier cluster Kubernetes.

```
minikube start.
```

Obtenir les noms des champs configuratbles du noeud Minikube

```
minikube config -h
```

Quelques manipulations usuelles.

Manipulation du cluster Kubernetes avec Minikube

Vérification de l'état du cluster :

```
minikube status
```

Récupérer la liste des noeuds du cluster

```
kubectl get nodes
```

Manipulation du cluster Kubernetes avec Minikube

Récupérer les information sur le cluster

```
kubectl cluster-info
```

Consulter les logs

```
minikube logs
```

Les options suivantes sont disponibles :

-f ou --follow : suivre en permanence les logs de Minikube (correspond à un tail -f sous Linux)

-n ou --length : nombre de lignes à afficher (50 par défaut)

--problems : afficher uniquement les logs qui pointent vers des problèmes connus

Manipulation du cluster Kubernetes avec Minikube

Vérifier que votre Minikube est à jour

```
minikube update-check
```

Pour supprimer votre cluster kubernetes

```
minikube delete
```

https://minikube.sigs.k8s.io/docs/start/

Travaux pratiques(TP2)