LES DIFFÉRENTS OBJETS ET COMPOSANTS DE KUBERNETES

Introduction

Lorsque vous commencez à travailler avec Kubernetes (K8), de nombreuses **nouvelles terminologies sont à apprendre**, à comprendre et à mémoriser. Ne serait-il pas formidable qu'au lieu d'avoir à chercher une définition de chaque terminologie, que vous puissiez **accéder à l'ensemble de la terminologie clé de Kubernetes** dans une seule liste ?

Devinez quoi ? C'est exactement le but de cet article ! Ce chapitre contiendra des **explications sur tous les objets et composants nécessaires à la compréhension de Kubernetes** avant de s'attaquer plus tard, dans les chapitres postérieurs à la partie pratique.

Les objets Kubernetes

Kubernetes a pour objectif principal de dissimuler la complexité de la gestion d'un parc de conteneurs, et pour cela, différents objets existent pour vous faciliter la vie, dont notamment :

- `Node` : Un node ("nœud" en fr) est une machine de travail du cluster Kubernetes. Ce sont des unités de travail qui peuvent être physiques, virtuelles mais aussi des instances cloud.
- `Pod` : Il s'agit de l'unité la plus petite de K8s, un pod encapsule le ou les

conteneur(s) formant votre application conteneurisée partageant ainsi la même stack réseau (chaque pod se voit attribuer une adresse IP unique) et le même stockage, plus précisément un volume partagé (tous les conteneurs du pod peuvent accéder aux volumes partagés, ce qui permet à ces conteneurs de partager plus facilement des données).

- `Replicas` : c'est le nombre d'instances d'un Pod ("réplique" en fr)
- `ReplicaSet` : s'assure que les réplicas spécifiés sont actifs
- `Deployment` : défini l'état désiré et fournit des mises à jour déclaratives de vos Pods et ReplicaSets.
- `Service` : Un service peut être défini comme un ensemble logique de pods exposés en tant que service réseau. C'est un niveau d'abstraction au-dessus du pod, qui fournit une adresse IP et un nom DNS unique pour un ensemble de pods. Avec les Services, il est très facile de gérer la configuration de Load Balancing (équilibreur de charge) permettant ainsi aux pods de scaler plus facilement.
- `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.

Les composants des nodes

Kubernetes suit l'architecture maître-esclave, le maître plus communément appelé **master** existe principalement pour gérer votre cluster Kubernetes. . Les esclaves sont quant à eux plus connus sous le nom de **workers** (on les appellent aussi **minions**) et ne sont là que pour fournir de la capacité et n'ont pas le pouvoir d'ordonner à

une autre node ce qu'il peut ou ne peut pas faire. Les composants clés du master et worker sont définis dans la section suivante.

Les composants du Master

Sur un node de type master, nous aurons les composants suivants :

- `kube-apiserver` : point d'entrée exposant l'API HTTP Rest de k8s depuis le maître du cluster Kubernetes. Différents outils et bibliothèques peuvent facilement communiquer avec l'API.
- `kube-scheduler` : Il est responsable de la répartition et l'utilisation de la charge de travail sur les nœuds du cluster selon les ressources nécessaires et celles disponibles.
- `kube-controller-manager` : ce composant est responsable de la plupart des collecteurs qui récupèrent des informations du cluster tout en effectuant des actions de correctives en cas de besoin, en apportant des modifications pour amener l'état actuel du serveur à l'état souhaité. Il est composé de plusieurs contrôleurs, on peut par exemple retrouver un contrôleur de réplication qui va s'assurer que vous avez le nombre désiré de répliques sur vos pods, mais aussi d'autres contrôleurs clés comme , le contrôleur de Endpoints, le contrôleur d'espace de noms et le contrôleur de compte de service.
- `cloud-controller-manager` : effectue les mêmes actions que le `kube-controller-manager` mais pour des fournisseurs de cloud sous-jacents (AWS, Azure, Google Cloud Platform, etc ...).
- `etcd` : il stocke les informations de configuration pouvant être utilisées par chacun des nœuds du cluster. Ces informations sont conservées sous forme

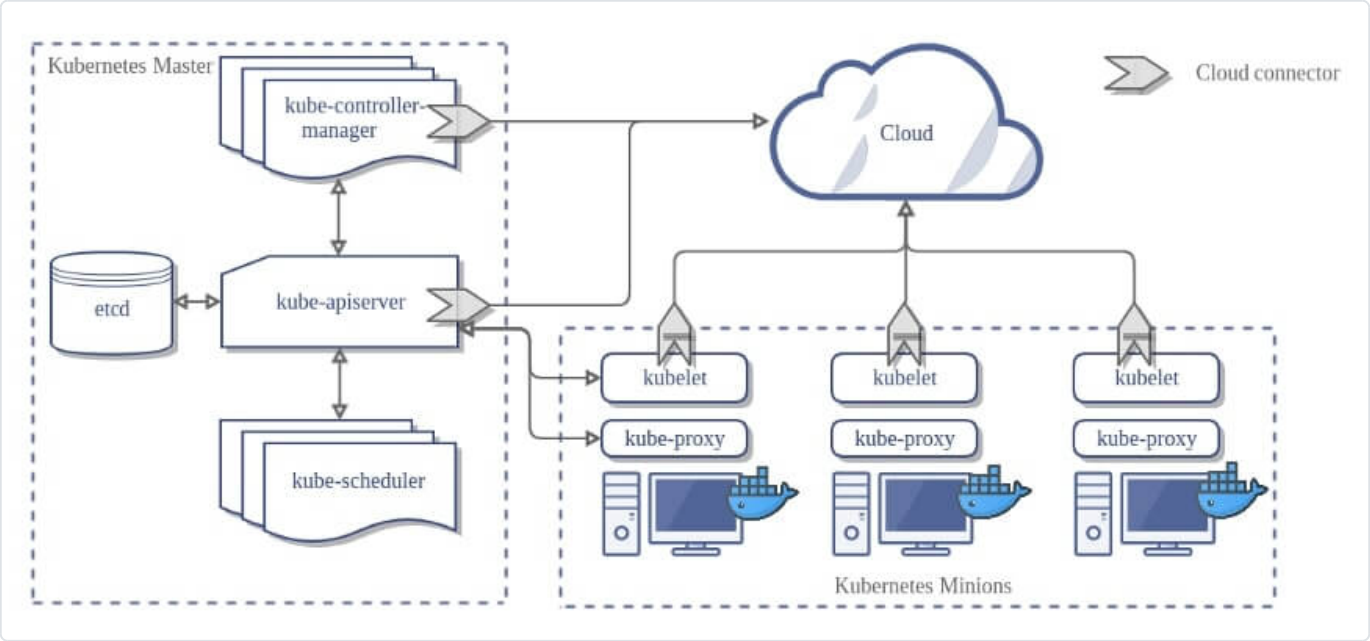
de clé et valeurs à haute disponibilité (le projet est téléchargeable ici)

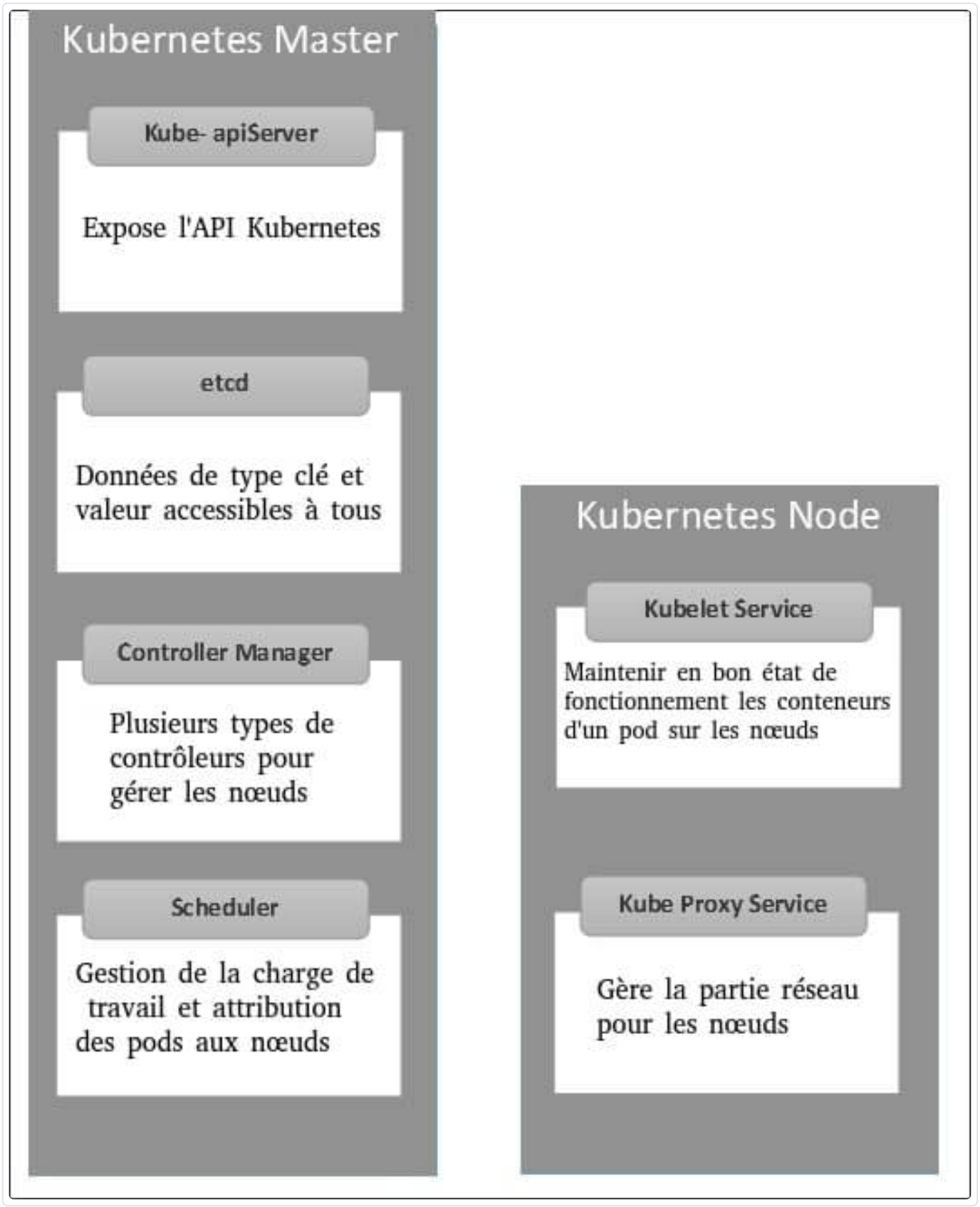
Les composants du Workers

Sur chaque worker, nous aurons les composants suivants :

- `kubelet` : Il s'agit d'un agent qui s'exécute dans chaque nœud chargé de relayer les informations au Master. Il interagit avec la base de données `etcd` du Master pour récupérer des informations afin de connaître les tâches à effectuer. Il assume la responsabilité de maintenir en bon état de fonctionnement les conteneurs d'un pod et s'assure qu'ils tournent conformément à la spécification. Il ne gère pas les conteneurs qui n'ont pas été créés par Kubernetes. Il communique avec le Master et redémarre le conteneur défaillant en cas de crash.
- `kube-proxy` : il active l'abstraction réseau du Service Kubernetes en maintenant les règles du réseau et permet l'exposition des 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).

Si on reprend tous les composants vus précédemment, nous aurons alors un **schéma d'architecture du cluster Kubernetes** ressemblant à ceci :





Information

Pour de la haute disponibilité, il faut au minimum deux répliques du Master dans des zones de disponibilités différentes.

Communication avec le cluster

Pour communiquer avec le cluster Kubernetes, on peut utiliser le binaire **kubectl**, ou une interface web de management nommé **Dashboard**.

kubeclt (ligne de commande)

L'outil en ligne de commande Kubernetes, kubectl, vous permet d'exécuter des commandes sur les clusters Kubernetes afin de créer et gérer des objets k8s et interagir avec l'API Kubernetes. Vous pouvez utiliser kubectl pour déployer des applications, inspecter et gérer les ressources du cluster et afficher les logs. Nous aurons l'occasion d'utiliser cet outil dans les chapitres suivants.

Dashboard (interface web)

Dashboard est l'interface utilisateur Web officielle de Kubernetes. Il fournit une vue d'ensemble des applications en cours d'exécution sur votre cluster Kubernetes. Il permet aux administrateurs de surveiller et gérer les opérations de leurs clusters Kubernetes, dépanner les applications conteneurisées et gérer les ressources du cluster.

Conclusion

J'espère que vous vous êtes bien familiarisé avec la terminologie Kubernetes, qu'il convient d'utiliser afin d'utiliser le même lexique que ça soit pour comprendre aisément les chapitres suivants ou pour une communication plus agréable avec votre équipe technique . Dans le chapitre suivant, nous verrons comment mettre en place notre premier cluster kubernetes avec l'outil Minikube.