Chapitre 3 : Kubernetes

Owner	louis reynouard
Tags	
Created time	@February 15, 2024 3:39 PM

Si vous souhaitez aller plus loin je vous recommande fortement cette vidéo youtube qui reprend tous les grands concepts de ce chapitre

Kubernetes, qu'est ce que c'est?

Kubernetes est une plateforme open-source de gestion de conteneurs. Il automatise les opérations liées aux conteneurs, telles que le déploiement, la mise à l'échelle, la surveillance et la gestion des erreurs, offrant ainsi une infrastructure robuste pour les applications modernes.

Prenons un exemple concret. Nous avons notre application encapsulée dans notre image docker. Pour répondre à l'attente de nos clients, nous basculons cette application en production. Se pose alors 2 problématiques:

- Nos clients ne peuvent pas se permettre de trouver une page error 404 (on est pas ici dans une logique de vie ou de mort mais imaginons).
- Nos clients auront besoin de l'application.. mais pas tout le temps ! et vous l'aurez deviné, ils vont en avoir besoin plus ou moins tous en même temps sur les créneaux de rush.

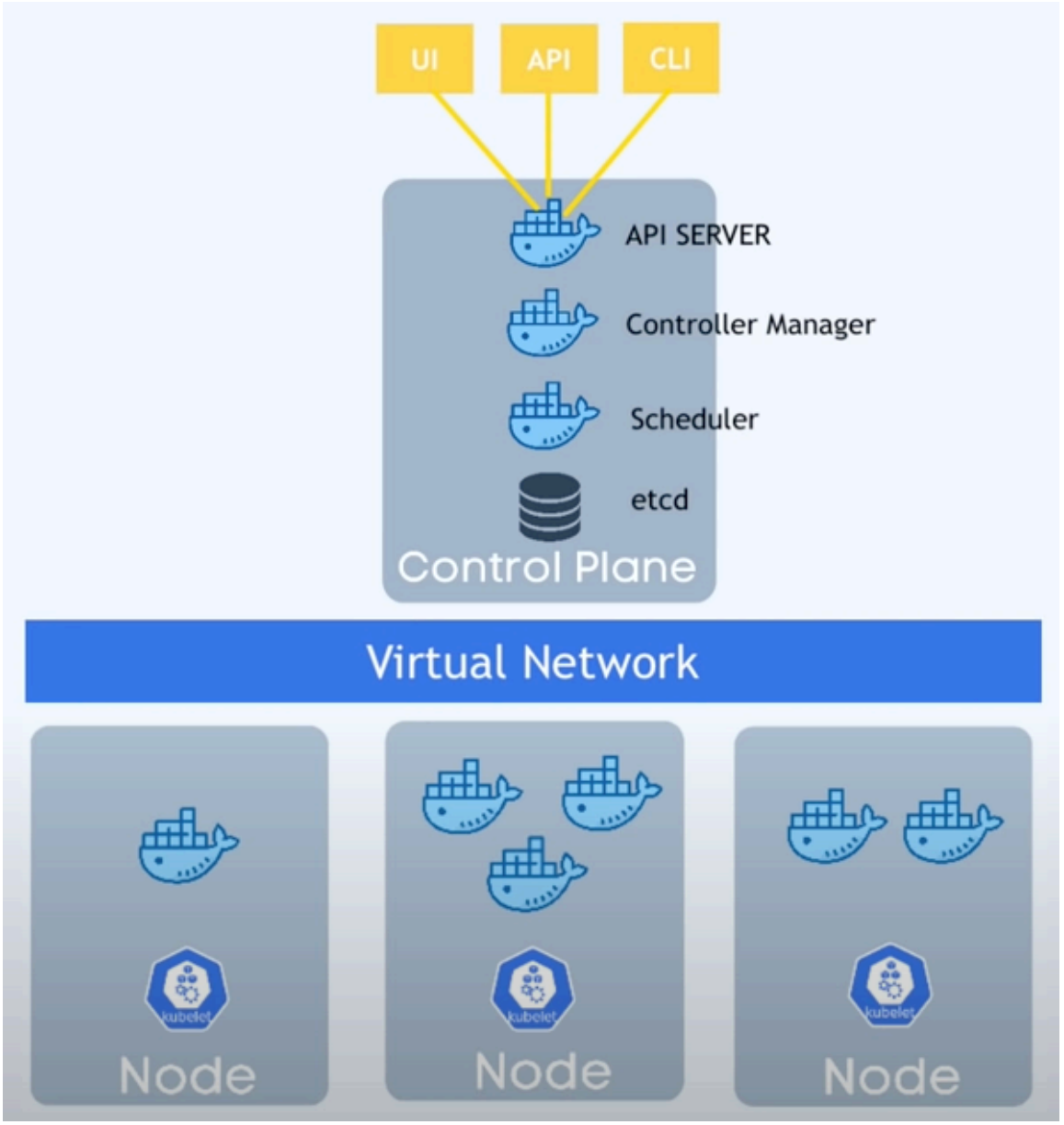
Alors comment dimensionner correctement nos infrastructures en s'assurant que nos clients ne trouvent jamais porte close?

C'est ce que nous allons voir avec kubernetes. Concrètement, cette plateforme va nous permettre de facilement dupliquer notre image docker de nos applications, et ce en fonction de la demande ou de potentielle panne (Ex: nous demandons à Kubernetes d'avoir au minimum deux instances de notre application, si l'une des deux crash alors "il" en relancera automatiquement une etc..)

Les composants de Kubernetes

Voici une présentation des principaux composants de Kubernetes :

1. **Cluster** :
   Un cluster Kubernetes est un ensemble de nœuds regroupés pour gérer et exécuter des applications. Il contient le maître (master) qui gère le cluster et les nœuds où les applications s'exécutent.
2. **Maître (Master)** :
   Le maître est le cerveau du cluster Kubernetes. Il gère les opérations globales et coordonne les nœuds (on verra ce que c'est juste après). Le maître comprend plusieurs composants :
   - **API Server** : Expose l'API Kubernetes que vous utilisez pour gérer et configurer le cluster. En résumé, ce composants vous permets d'interagir avec votre cluster kubernetes.
   - **Etcd** : Base de données de configuration détenant en permanence les informations de configuration et d'état du cluster kubernetes. En cas de crash global, les fonctionnalités de restauration se feront via les snapshot sauvegardés dans l'*ETCD*
   - **Control Manager** : Garde une trace de tout ce qu'il se passe dans les noeuds. Gère ensuite les tâches de contrôle du cluster, comme la création et la suppression de ressources etc..
   - **Scheduler** : Décide où exécuter les conteneurs en fonction des ressources disponibles et des exigences des conteneurs. Il permet une bonne répartition de la charge de travail entre les différentes instances de votre application.



3. **Nœud (Node)** :
   Un nœud est une machine qui exécute des conteneurs. Chaque nœud est géré par le maître et contient plusieurs composants :
   - **Kubelet** : Agent s'exécutant sur chaque nœud qui communique avec le maître et s'assure que les conteneurs sont en cours d'exécution dans un Pod.
   - **Kube-proxy** : Gère la mise en réseau du cluster et met en œuvre les règles de réseau définies par l'utilisateur.
4. **Pod** :
   Un
   pod est la plus petite unité dans Kubernetes, pouvant contenir un ou

plusieurs conteneurs qui partagent un espace réseau et de stockage. Les pods sont planifiés et déployés sur les nœuds. Par convention, nous essayons de ne déployer qu'une application par *pod*.

> ⚠️ **Chaque pod dispose de son Ip. Si le pod est relancé, l'ip change !**

5. **Service** :
   Un
   service Kubernetes permet d'exposer les pods et d'assurer la connectivité réseau. Il fournit une IP stable et un nom DNS pour les pods, ce qui permet aux applications de se connecter à d'autres services sans se soucier de la dynamique des pods.
6. **Ingress** :

   L'Ingress est un objet Kubernetes qui gère la gestion des règles d'accès au cluster pour les services en fonction des noms d'hôte ou des chemins. Cela permets donc d'accéder à notre pod (et donc notre application) depuis l'extérieur du cluster.
7. **ConfigMap et Secret** :
   Les
   ConfigMaps et les *Secrets* sont des objets Kubernetes permettant de stocker et gérer des configurations et des secrets (comme des clés d'API) séparément des conteneurs. Nous pourrons stocker toutes les informations non sensibles dans les *Configmaps*, les autres dans les *Secrets*.
8. **Volume** :
   Les
   volumes Kubernetes permettent aux conteneurs de stocker et d'accéder à des données en dehors de leur cycle de vie. Ils sont utilisés pour gérer le stockage persistant dans les conteneurs qui n'est pas géré par kubernetes de manière standard. Sans ces *volumes*, lorsqu'un pod est arrêté, nous ne pourrions conserver ses données...
9. **Réplication Controller (ou Deployment)** :
   Le
   Replication Controller (ou le *Deployment*) gère la duplication des pods et assure leur disponibilité en permanence, même en cas d'échec des nœuds.

Il permets de créer un format standard de pod que nous pouvons répliquer au besoin (en cas de crash ou de grandes demandes du service).

10. **StateFulset :**

    Les deployment ne permettent pas de dupliquer des éléments qui disposent d'un état (state en anglais). Prenons l'exemple d'une application disposant d'une base de données. Nous pouvons dupliquer facilement l'application avec *deployment*, cependant la base de données doit rester constante d'une application à l'autre. Pour dupliquer la base de données, nous utiliserons donc l'objet StateFulSet qui prend en charge la gestion de l'état.

11. **Horizontal Pod Autoscaler (HPA)** :
    Le HPA ajuste automatiquement le nombre de répliques d'un pod en fonction de la charge de travail en utilisant les métriques définies.

En résumé, Kubernetes offre une infrastructure robuste pour le déploiement et la gestion d'applications conteneurisées, simplifiant ainsi le développement et le déploiement d'applications évolutives et fiables dans des environnements variés. Nous aurons l'occasion d'utiliser les différents composants abordés ici durant le TP.

TP - Déploiement de l'Application Dockerisée sur Kubernetes avec Minikube

👉 Finissez la partie 3 du TP