ORCHESTRATION

SOMMAIRE

- Introduction à l'orchestration
- Concepts de base de l'orchestration
- Les systèmes d'orchestration
- Orchestration avec Docker
- Kubernetes
- Pratique Kubernetes

SOMMAIRE DÉTAILLÉ

1. Introduction à l'orchestration : définition de l'orchestration, pourquoi l'orchestration est-elle importante, exemples d'utilisation de l'orchestration.
2. Concepts de base de l'orchestration : définitions des termes couramment utilisés dans l'orchestration, tels que les conteneurs, les images, les déploiements, les services, les réseaux, les volumes, etc.
3. Les systèmes d'orchestration : comparaison des différents systèmes d'orchestration populaires, tels que Docker Swarm, Mesos, et Kubernetes.
4. Orchestration avec Docker : comment utiliser Docker pour orchestrer des conteneurs, comment créer des images, comment gérer les déploiements, comment utiliser les réseaux et les volumes avec Docker
5. Kubernetes : Introduction à Kubernetes, comprendre les concepts de base de Kubernetes, tels que les Nodes, les Pods, les Services, les Deployment et les ConfigMap.
6. Pratique : Mettre en pratique ce que vous avez appris en créant un cluster Kubernetes simple, déploiement d'application sur cluster, scaling, rolling updates, Self Healing

INTRODUCTION À L'ORCHESTRATION

- L'orchestration est un processus qui permet de gérer et de coordonner les tâches et les actions d'un système complexe.
- Dans le contexte de l'informatique**, l'orchestration est utilisée pour gérer et coordonner les conteneurs**, **les machines virtuelles, les clusters et les services dans un environnement distribué.**

- La principale raison d'utiliser l'orchestration est **d'améliorer l'agilité, l'évolutivité et la fiabilité des systèmes**. En utilisant l'orchestration, vous pouvez facilement **gérer des centaines ou des milliers de conteneurs ou de machines virtuelles,** déployer des applications en quelques minutes, et équilibrer automatiquement la charge sur les différents éléments du système.

- Exemples d'utilisation de l'orchestration:
  - Déploiement d'une application en conteneurs sur un cluster de machines
  - Gérer la scalabilité des instances pour un service web
  - Automatisation de la configuration de machines
  - Automatisation de la découverte de service pour une application distribuée
  - Gestion de la disponibilité et la tolérance aux pannes d'un système
  - Il est important de noter que les systèmes d'orchestration ne se limitent pas aux conteneurs, mais peuvent également gérer des machines virtuelles et des services répartis sur différents systèmes.

CONCEPTS DE BASE DE L'ORCHESTRATION

- **Conteneurs** : Un conteneur est une **instance d'une image logicielle**, qui permet de lancer une application ou un service dans un environnement isolé. Les conteneurs partagent le noyau de l'hôte, ce qui les rend plus légers et plus rapides à démarrer que les machines virtuelles.

- **Images** : Une image est un fichier qui **contient tout ce dont une application a besoin pour s'exécuter**, y compris le code source, les bibliothèques, les configurations, etc. Les images sont utilisées pour créer des conteneurs.

- **Déploiements** : Un déploiement **décrit comment une application ou un service doit être déployé sur un cluster**, y compris le nombre d'instances souhaitées, les métadonnées, les stratégies de mise à jour, etc.

- **Services** : Un service est une **abstraction qui permet de diriger le trafic vers un groupe de conteneurs**. Les services peuvent être configurés pour assurer la disponibilité, l'équilibrage de charge, et la tolérance aux pannes.

- **Réseaux** : Les réseaux permettent **aux conteneurs de communiquer entre eux et avec l'extérieur**. Ils peuvent être configurés pour isoler les conteneurs, pour créer des sous-réseaux, pour connecter des conteneurs à des services externes, etc.

- **Volumes** : Les volumes sont des **espaces de stockage qui peuvent être utilisés par les conteneurs**. Ils permettent de stocker les données de manière persistante, même si les conteneurs sont redémarrés ou répliqués.

- Il est important de comprendre que ces concepts de base sont liés les uns aux autres et qu'ils sont utilisés de différentes manières selon le système d'orchestration choisi.

LES SYSTÈMES D'ORCHESTRATION

- **Docker Swarm** : **Docker Swarm est un système d'orchestration intégré à Docker** qui permet de gérer des conteneurs sur **plusieurs hôtes**. Il permet de créer des clusters de conteneurs, de déployer des applications sur des clusters, de gérer les réseaux et les volumes, et de gérer la scalabilité des applications.

- **Mesos** : **Mesos est un système d'orchestration distribué qui gère les ressources des machines d'un cluster de manière centralisée**. Il permet de gérer des conteneurs et des machines virtuelles, de planifier les tâches sur des machines libres, et de gérer la scalabilité et la tolérance aux pannes des applications.

- **Kubernetes** : Kubernetes est un système d'orchestration open-source pour les conteneurs qui permet de gérer des clusters de conteneurs à l'échelle. **Il fournit des fonctionnalités avancées pour la gestion de la scalabilité, de la disponibilité, de la tolérance aux pannes, de la sécurité et de la mise à jour des applications**. Il est également extensible pour intégrer des fonctionnalités supplémentaires, et peut être **utilisé avec des outils tels qu'ETCD ou Prometheus pour améliorer les fonctionnalités de base.**

- Il est important de noter que **chacun de ces systèmes d'orchestration a ses propres avantages et inconvénients** et qu'il est important de choisir celui qui **convient le mieux à vos besoins en fonction de vos exigences en matière de scalabilité, de disponibilité, de sécurité, de coûts, etc**.
- Kubernetes est devenu le système d'orchestration le plus populaire et est utilisé dans de nombreux environnements d'entreprise et Cloud, mais d'autres systèmes comme Docker Swarm ou Mesos peuvent être adaptés pour certaines utilisations ou certaines exigences.

ORCHESTRATION AVEC DOCKER

- **Utilisation de Docker pour orchestrer des conteneurs** : Docker fournit des commandes pour créer et gérer des conteneurs, telles que "docker run", "docker start", "docker stop" pour démarrer, arrêter et supprimer des conteneurs. Il est également possible de créer des conteneurs à partir d'images existantes en utilisant "docker pull" pour récupérer des images sur un dépôt d'images et "docker create" pour créer un conteneur à partir de cette image.

- **Création d'images** : Les images Docker sont créées en utilisant un fichier "Dockerfile" qui décrit les étapes pour créer l'image. Il peut inclure des instructions pour copier les fichiers, installer des paquets, définir des variables d'environnement, etc. Les images peuvent également être créées à partir d'autres images existantes en utilisant les instructions "FROM" dans un Dockerfile.

- **Gestion des déploiements** : Il est possible de gérer les déploiements avec Docker en utilisant les commandes "**docker service create**" pour créer un service, "**docker service update**" pour mettre à jour un service, et "**docker service rm**" pour supprimer un service. Il est également possible de gérer les déploiements en utilisant des outils de **gestion de configuration tels que Ansible ou Terraform qui peuvent automatiser les tâches de déploiement.**

- **Utilisation des réseaux et des volumes** : Les conteneurs peuvent être connectés à des réseaux et des volumes en utilisant les options de la commande "docker run" ou en définissant les options de réseau et de volume dans un fichier "docker-compose.yml" pour gérer les déploiements.

INTRODUCTION À KUBERNETES

Intelligent scheduling

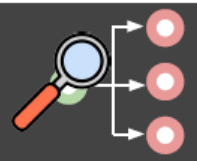
Vous fournissez à Kubernetes un cluster de nœuds qu'il peut utiliser pour exécuter des tâches conteneurisées. Vous indiquez à Kubernetes la quantité de CPU et de mémoire (RAM) que chaque conteneur a besoin. Kubernetes peut adapter les conteneurs sur vos nœuds afin d'utiliser au mieux utilisation de vos ressources.

Self healing

Kubernetes redémarre les conteneurs qui échouent, remplace les conteneurs, tue les conteneurs qui ne répondent pas à votre contrôle de santé défini par l'utilisateur, et ne les annonce pas aux clients jusqu'à ce qu'ils soient prêts à servir.

Horizontal scaling

Kubernetes vous permet de faire évoluer facilement vos applications manuellement avec une simple ligne de commande ou dynamiquement sur la base de métriques standard comme le CPU et la mémoire, ou sur la base de paramètres personnalisés.



Service discovery & Load Balancing

Kubernetes peut exposer un conteneur en utilisant son nom DNS ou en utilisant sa propre adresse IP. Si le trafic vers un conteneur est important, Kubernetes peut équilibrer la charge et distribuer le trafic réseau afin que le déploiement soit stable.

Automated Rollout & Rollback

Vous pouvez décrire l'état souhaité pour vos conteneurs déployés en utilisant Kubernetes, et il peut changer l'état réel vers l'état souhaité à une vitesse contrôlée. Par exemple, vous pouvez automatiser Kubernetes pour créer de nouveaux conteneurs pour votre déploiement, retirer les conteneurs existants et adopter toutes leurs ressources vers le nouveau conteneur.

Secret & Config management

Kubernetes vous permet de stocker et gérer des informations sensibles, comme les mots de passe, les jetons OAuth, et les clés ssh. Vous pouvez déployer et mettre à jour les secrets et la configuration d'application sans avoir à reconstruire vos images de conteneur, et sans exposer les secrets dans votre configuration de la pile.

INTRODUCTION À KUBERNETES

Intelligent scheduling

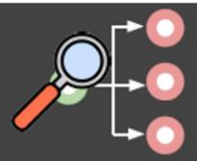
Vous fournissez à Kubernetes un cluster de nœuds qu'il peut utiliser pour exécuter des tâches conteneurisées. Vous indiquez à Kubernetes la quantité de CPU et de mémoire (RAM) que chaque conteneur a besoin. Kubernetes peut adapter les conteneurs sur vos nœuds afin d'utiliser au mieux utilisation de vos ressources.

Self healing

Kubernetes redémarre les conteneurs qui échouent, remplace les conteneurs, tue les conteneurs qui ne répondent pas à votre contrôle de santé défini par l'utilisateur, et ne les annonce pas aux clients jusqu'à ce qu'ils soient prêts à servir.

Horizontal scaling

Kubernetes vous permet de faire évoluer facilement vos applications manuellement avec une simple ligne de commande ou dynamiquement sur la base de métriques standard comme le CPU et la mémoire, ou sur la base de paramètres personnalisés.



Service discovery & Load Balancing

Kubernetes peut exposer un conteneur en utilisant son nom DNS ou en utilisant sa propre adresse IP. Si le trafic vers un conteneur est important, Kubernetes peut équilibrer la charge et distribuer le trafic réseau afin que le déploiement soit stable.

Automated Rollout & Rollback

Vous pouvez décrire l'état souhaité pour vos conteneurs déployés en utilisant Kubernetes, et il peut changer l'état réel vers l'état souhaité à une vitesse contrôlée. Par exemple, vous pouvez automatiser Kubernetes pour créer de nouveaux conteneurs pour votre déploiement, retirer les conteneurs existants et adopter toutes leurs ressources vers le nouveau conteneur.

Secret & Config management

Kubernetes vous permet de stocker et gérer des informations sensibles, comme les mots de passe, les jetons OAuth, et les clés ssh. Vous pouvez déployer et mettre à jour les secrets et la configuration d'application sans avoir à reconstruire vos images de conteneur, et sans exposer les secrets dans votre configuration de la pile.

INTRODUCTION À KUBERNETES

Kubernetes est un système d'orchestration open-source pour les conteneurs qui permet de gérer des clusters de conteneurs à l'échelle. Il fournit des fonctionnalités avancées pour la gestion de la **scalabilité**, de la **disponibilité,** de la **tolérance aux pannes**, de la **sécurité** et de la **mise à jour des applications**.

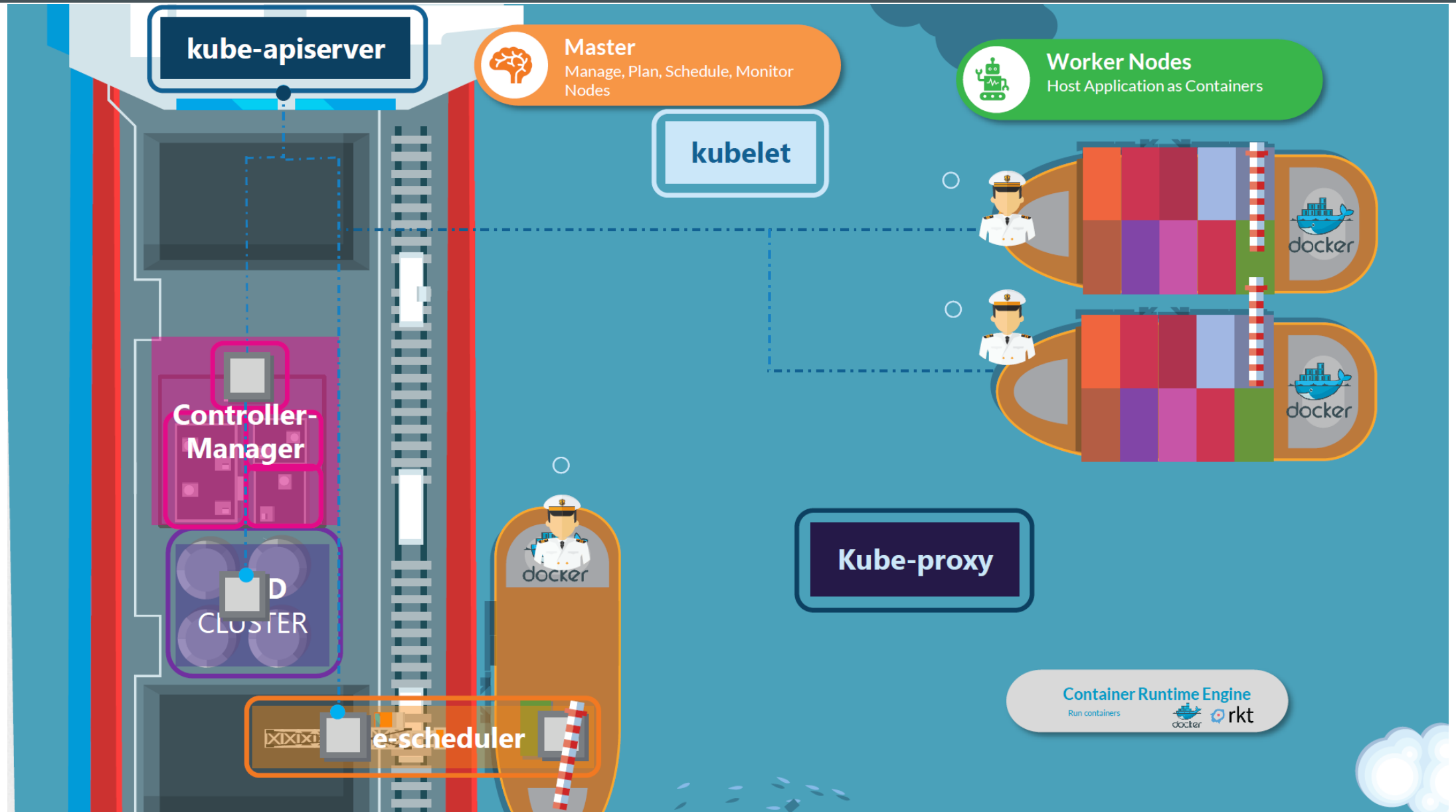
Pour utiliser Kubernetes, il est important de comprendre les concepts de base, tels que:

- **Nodes**: c'est une machine physique ou virtuelle **qui exécute les conteneurs,**
- **Pods**: qui est le plus petit et le plus fondamental élément dans Kubernetes, **un pod est une ou plusieurs instances de conteneur qui partagent le même espace de nom et les mêmes ressources.**
- **Services**: qui **fournit une façon de grouper des Pods qui ont une fonctionnalité commune**, il permet également la mise en place d'un point d'accès stable à une application ou un groupe d'application
- **Deployments** : qui gère les **mises à jour et la scalabilité des Pods**,
- **ConfigMap**: qui permet de **stocker des informations de configuration**,

Kubernetes est généralement utilisé avec un ensemble d'outils complémentaires pour améliorer les fonctionnalités de base, tels que **ETCD** pour stocker les données de configuration, **Prometheus pour la surveillance et l'analyse des données de métriques, ou Helm pour la gestion des déploiements.**

Il est important de comprendre que Kubernetes est un système complexe qui nécessite de la pratique et de la compréhension pour maîtriser ses concepts et fonctionnalités. Il est aussi conseillé de suivre des cours et des tutorials pour comprendre les concepts dans la profondeur et pratiquer sur des plateformes en ligne pour des simulations.

ARCHITECTURE KUBERNETES



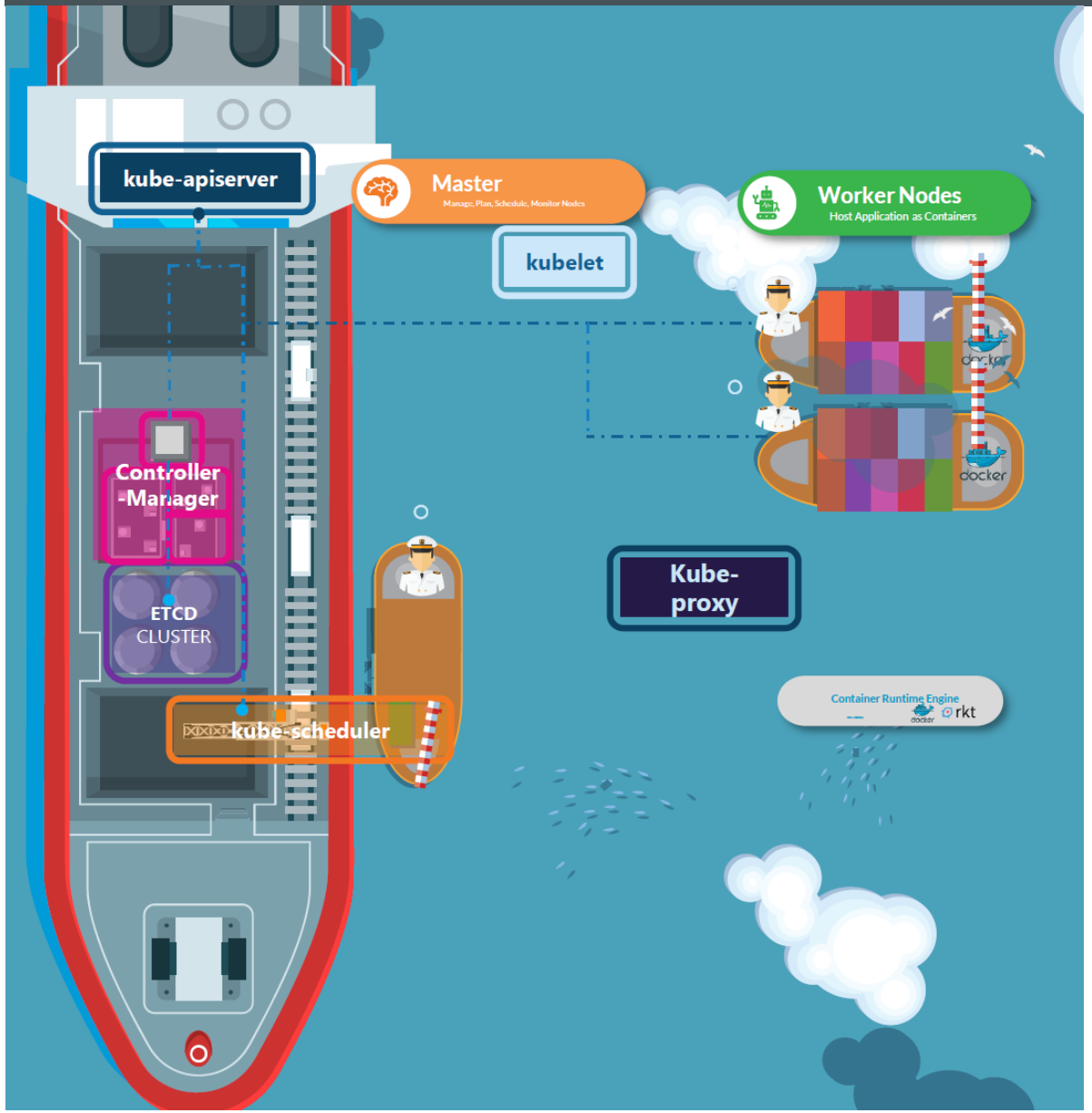
ARCHITECTURE KUBERNETES

- **Kubelet** : Kubelet est un composant qui s'exécute sur chaque node et est responsable de veiller à ce que les pods et les conteneurs soient en bonne santé. Il surveille les pods et les conteneurs pour s'assurer qu'ils sont en cours d'exécution, qu'ils ont suffisamment de ressources, etc.
- **Kube-apiserver** : Kube-apiserver est le point d'entrée principal pour les commandes et les requêtes envoyées à Kubernetes. Il expose une API REST pour les commandes kubectl et les autres clients, il s'assure que les requêtes sont authentifiées
- **Controller Manager** : Le Controller Manager est un composant de Kubernetes qui surveille l'état du cluster et prend des décisions pour maintenir l'état désiré. Il est responsable de la gestion des déploiements, des ReplicaSets et des autres objets de contrôle de l'état.

ARCHITECTURE KUBERNETES

- **ETCD** est un système de stockage de données distribué qui est utilisé pour stocker les données de configuration de base de Kubernetes, telles que les informations sur les Nodes, les Pods, les Services et les Deployments.
- **kube-proxy** : kube-proxy est un composant qui s'exécute sur chaque node et est responsable de la mise en place du réseau pour les services Kubernetes. Il gère les communications entre les services et les pods sur différents nodes.
- **kube-scheduler** : kube-scheduler est un composant de planification de Kubernetes qui assigne les pods à des nodes pour l'exécution. Il utilise des algorithmes de planification pour équilibrer les charges et utiliser efficacement les ressources disponibles sur les nodes.

ARCHITECTURE KUBERNETES



- Ensemble, ces composants travaillent ensemble pour gérer les conteneurs et les services dans un cluster Kubernetes.
- Les données de configuration et les états sont stockés dans ETCD,
- Tandis que kube-apiserver expose une API pour les commandes et les requêtes.
- Les actions de planification et de contrôle de l'état sont gérées par le Controller Manager et le kube-scheduler,
- Les communications réseau sont gérées par le kube-proxy,
- Kubelet surveille l'état des pods et des conteneurs sur chaque node.

ETCD

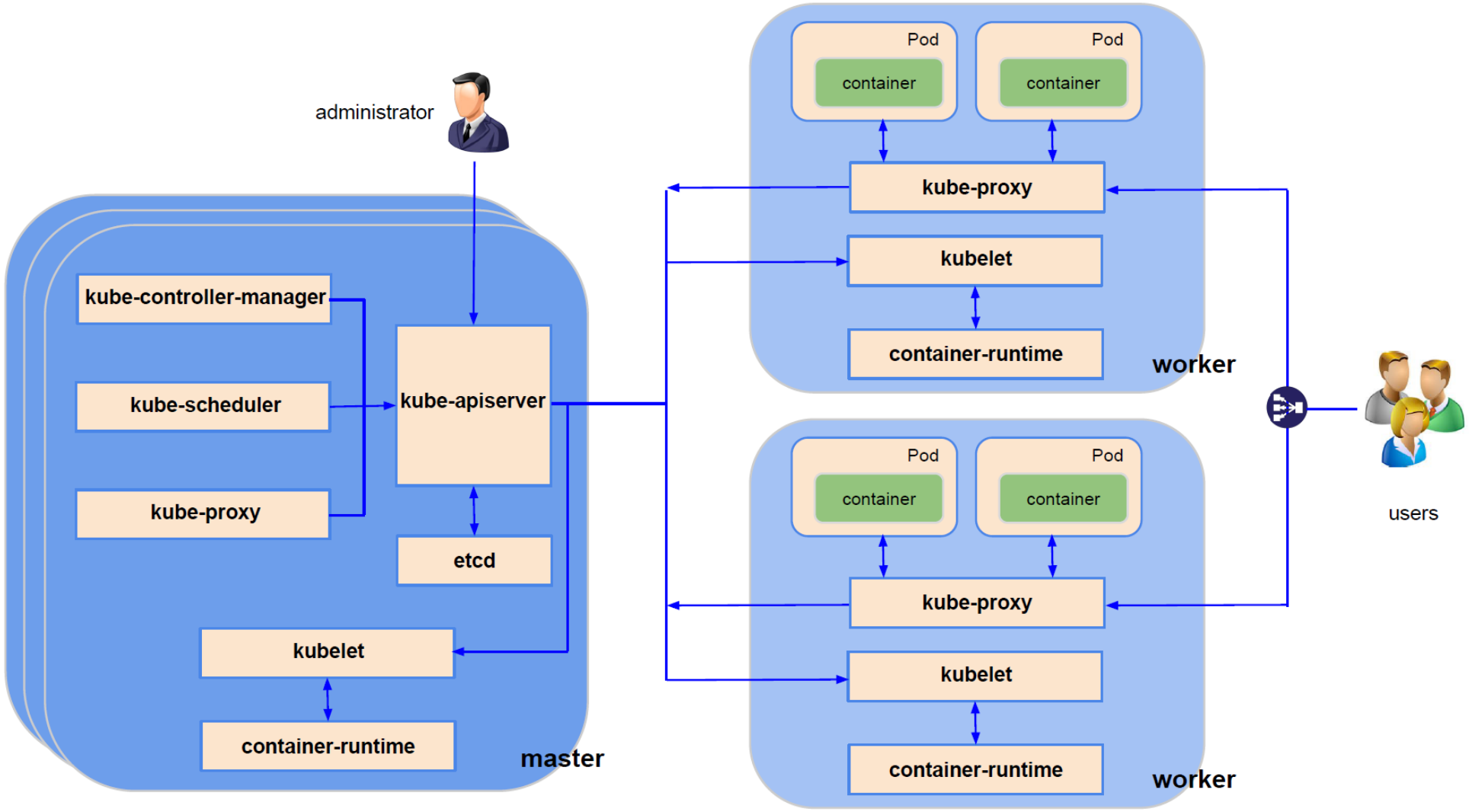
- **ETCD** est un **système de stockage de données distribué** qui permet de stocker et de **récupérer des données de configuration et de métadonnées pour les systèmes distribués, tels que les clusters Kubernetes**. Il est utilisé pour **stocker les données de configuration** de base de Kubernetes, telles que les informations sur les **Nodes, les Pods, les Services et les Deployments.**

- ETCD est basé sur la technologie **Raft**, une implémentation d'un protocole de consensus distribué qui garantit la **fiabilité et la tolérance aux pannes des données stockées. Il permet à plusieurs instances ETCD de fonctionner en cluster pour garantir la disponibilité des données même en cas de panne d'un nœud.**

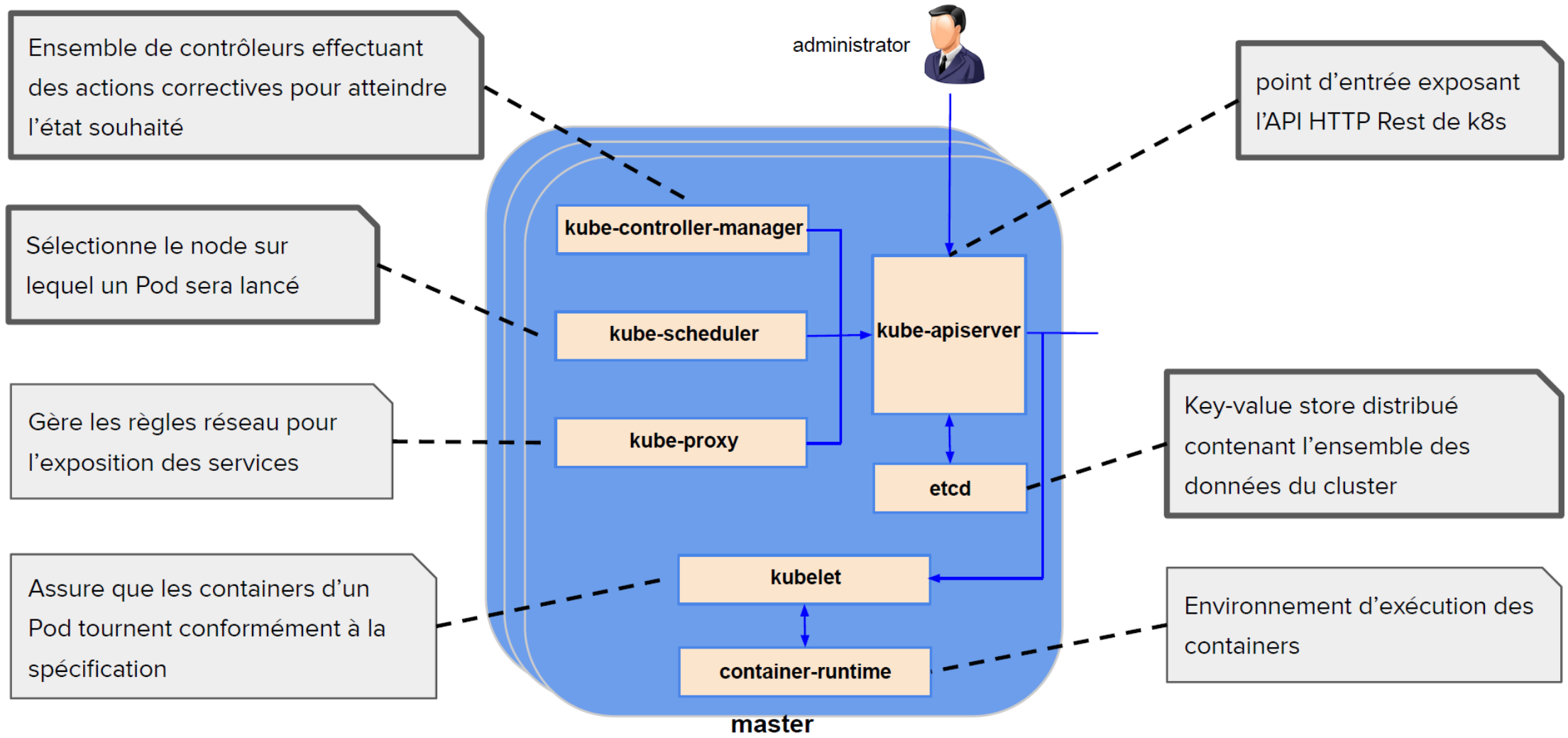
- ETCD est écrit **en Go et est open-source**. Il est également facilement extensible pour intégrer des fonctionnalités supplémentaires. Il permet **l'accès aux données via une API REST**, ce qui facilite son utilisation avec différents systèmes et outils. Il est également possible de l'utiliser avec des outils de **gestion de configuration tels que Ansible ou Terraform.**

- En utilisant ETCD, **Kubernetes peut garantir que tous les Nodes d'un cluster ont une vue cohérente de la configuration de l'application et peuvent prendre des décisions éclairées en cas de panne d'un nœud**. Il permet également de stocker des données de configuration pour les addons et les outils complémentaires de Kubernetes.

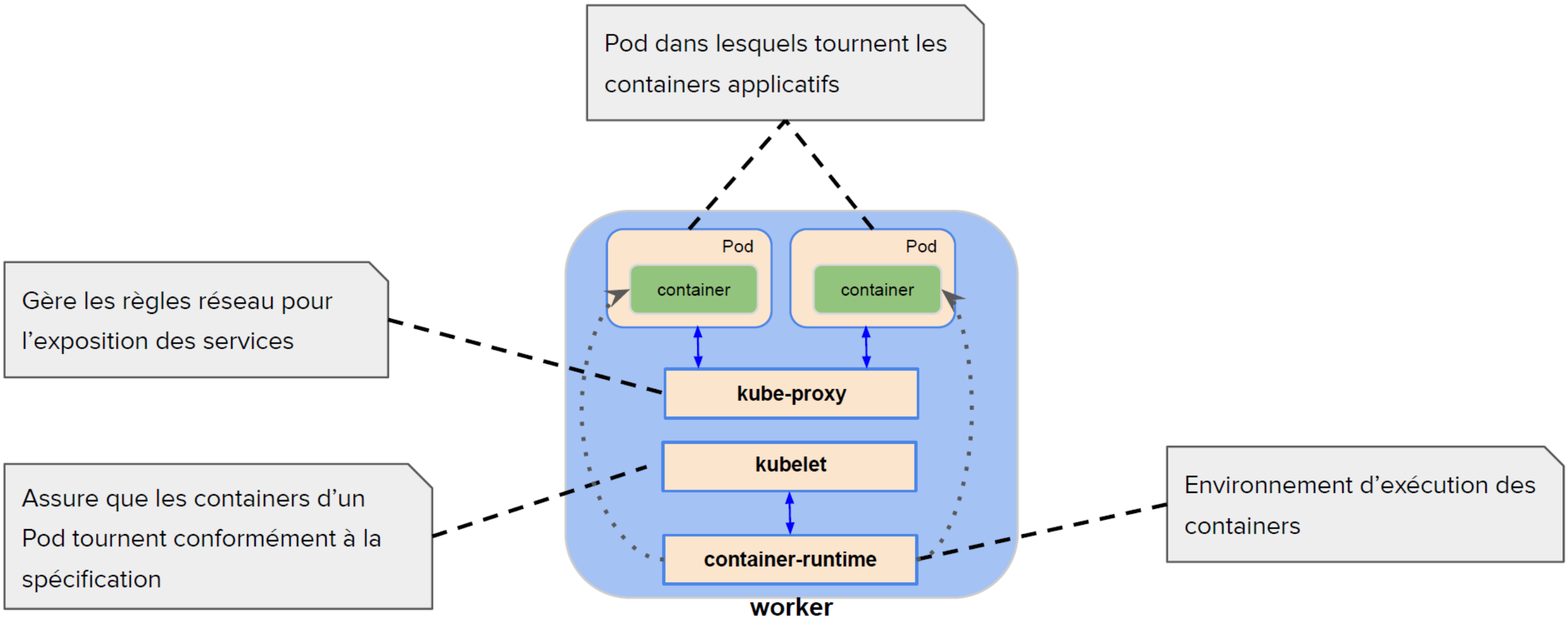
ARCHITECTURE KUBERNETES - VISION D'ENSEMBLE



ARCHITECTURE KUBERNETES - MASTER



ARCHITECTURE KUBERNETES - WORKER



ARCHITECTURE KUBERNETES - VISION D'ENSEMBLE

```
$ kubectl get pod -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-86c58d9df4-q84tg               1/1     Running   0          5m
coredns-86c58d9df4-qm6sx               1/1     Running   0          5m
etcd-k8s-node-1                        1/1     Running   0          4m
kube-apiserver-k8s-node-1              1/1     Running   0          4m
kube-controller-manager-k8s-node-1     1/1     Running   0          4m
kube-proxy-nc9cb                       1/1     Running   0          5m
kube-proxy-wll56                       1/1     Running   0          2m
kube-proxy-x77gd                       1/1     Running   0          2m
kube-scheduler-k8s-node-1              1/1     Running   0          4m
weave-net-29x5x                        2/2     Running   0          30s
weave-net-77ppt                        2/2     Running   0          30s
weave-net-q7fb5                        2/2     Running   0          30s
```

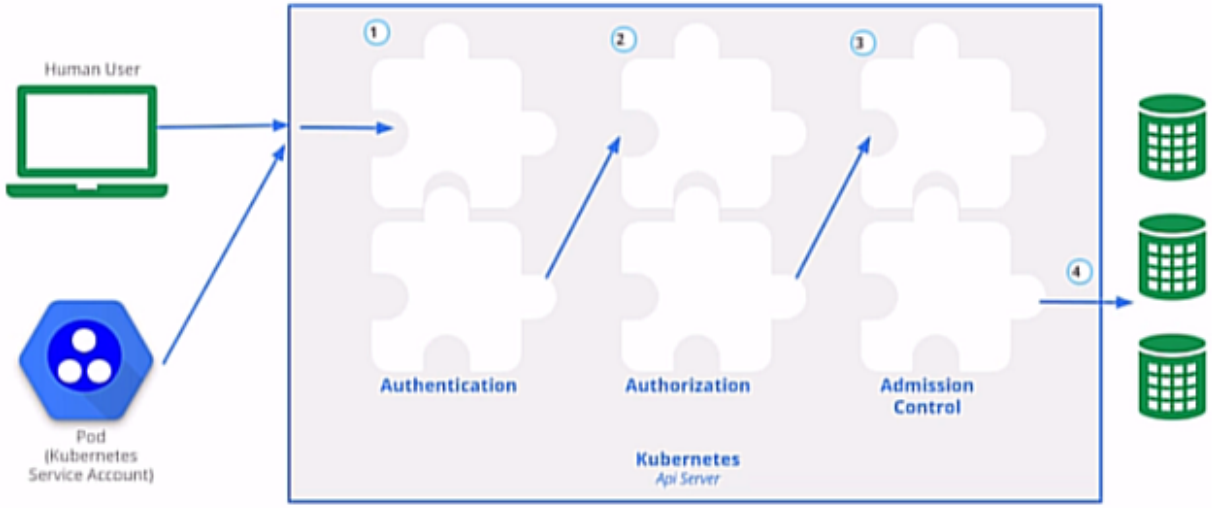
Processus tournant sur les masters

Processus tournant sur chacun des nodes (masters et workers)

Processus répliqués sur le cluster (ex: serveur DNS add-on)

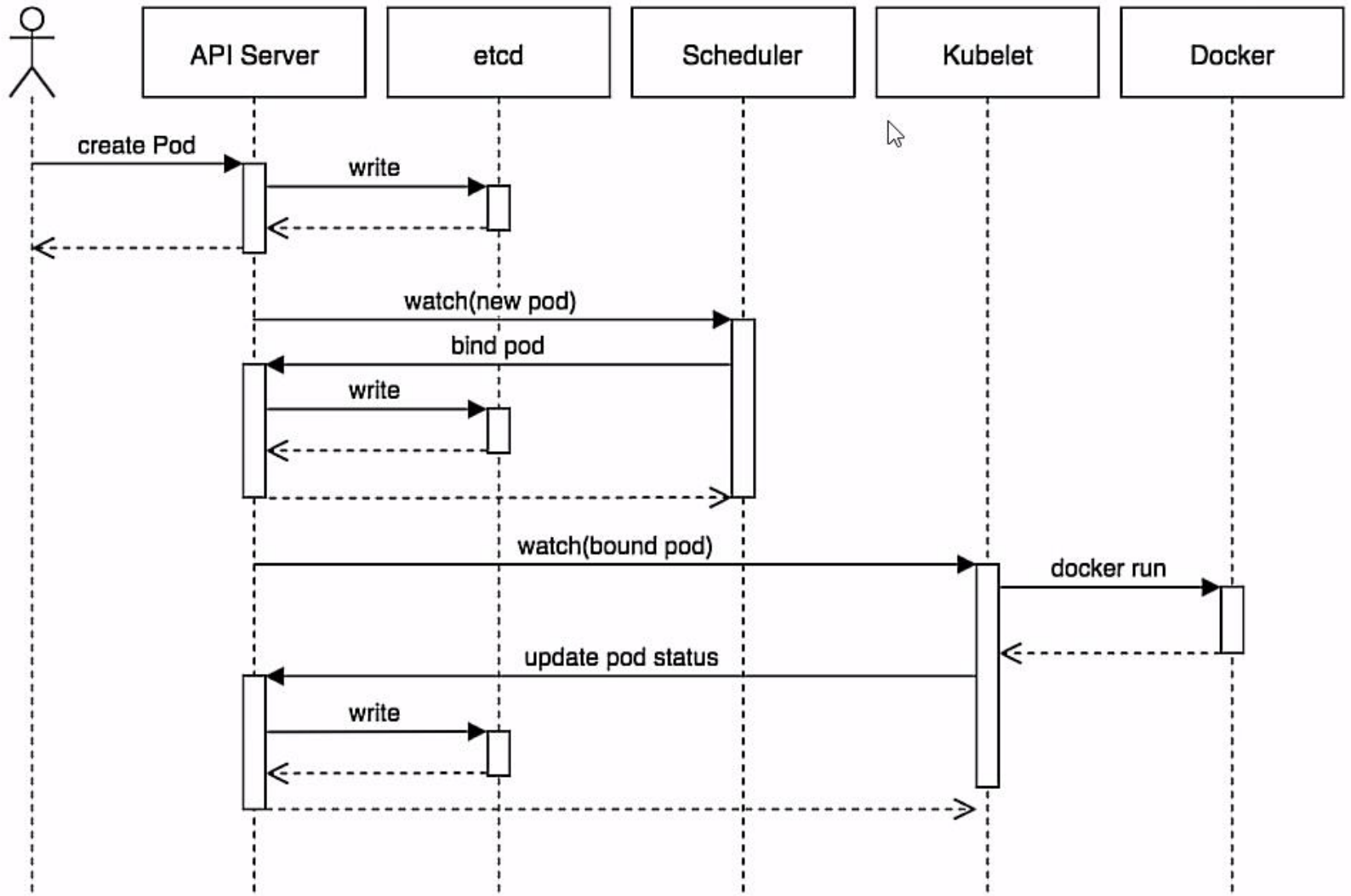
ARCHITECTURE KUBERNETES - API SERVER

- API Rest / spécification OpenAPI
  - https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.14
- Clients: kubectl, interface web, application tournant dans le cluster
- Chaque requête passe dans un pipeline
  - authentification
  - autorisation
  - admission contrôleurs



https://kubernetes.io/docs/admin/accessing-the-api/

ARCHITECTURE KUBERNETES - API SERVER



https://blog.heptio.com

PRATIQUE KUBE
CLUSTER KUBE SIMPLE

- Création d'un cluster Kubernetes simple:

- Utilisation de minikube pour installer un cluster Kubernetes localement sur votre ordinateur
- Utilisation de kubeadm pour installer un cluster Kubernetes sur des machines virtuelles ou physiques
- Suivre les instructions pour installer minikube ou kubeadm sur leur ordinateur personnel et vérifier que le cluster est opérationnel en exécutant des commandes de base de kubectl.
- **Voir le document "Kubernetes - Installation" fourni par le formateur.**

PRATIQUE KUBE
CLUSTER KUBE SIMPLE

1. Exercice 1 : Installation de minikube sur votre ordinateur personnel:
    1. Assurez-vous d'avoir Virtualization (Hyper-V, Virtualbox, VMware) installé sur votre ordinateur.
    2. Téléchargez la dernière version de minikube à partir de https://github.com/kubernetes/minikube/releases
    3. Installez minikube sur votre ordinateur en suivant les instructions d'installation pour votre système d'exploitation
        1. eg Sur Ubuntu :

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \
      && chmod +x minikube

sudo install minikube /usr/local/bin
```

PRATIQUE KUBE
CLUSTER KUBE SIMPLE

4. Vérifiez que minikube est installé en exécutant la commande **minikube version**
5. Démarrez minikube en utilisant la commande **minikube start --driver=docker**
6. Vérifiez que minikube est en cours d'exécution en utilisant la commande **minikube status**
7. Utilisez kubectl pour interagir avec le cluster minikube en utilisant la commande **kubectl config use-context minikube**
8. Exécutez des commandes de base de kubectl telles que **kubectl get nodes**, **kubectl get pods**, etc. pour vérifier que le cluster est opérationnel.

PRATIQUE KUBE
CLUSTER KUBE SIMPLE

1. Exercice 2 : Installation de kubeadm sur des machines virtuelles ou physiques :

2. Installez Docker sur chaque machine en suivant les instructions d'installation pour votre système d'exploitation. Sur Ubuntu vous devrez utiliser les commandes suivantes :

```
• sudo apt-get update
• sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common
• curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
• sudo apt-get update
• sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

PRATIQUE KUBE
CLUSTER KUBE SIMPLE

1. Installez kubeadm, kubelet et kubectl sur chaque machine en suivant les instructions d'installation pour votre système d'exploitation. Par exemple, sur Ubuntu vous devrez utiliser les commandes suivantes :

- curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
- echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list
- sudo apt-get update
- sudo apt-get install -y kubeadm kubelet kubectl

PRATIQUE KUBE
CLUSTER KUBE SIMPLE

1. Initialisez le contrôleur de cluster sur une des machines en utilisant la commande:
   - sudo kubeadm init
2. Configurez kubectl pour utiliser le nouveau cluster en utilisant les commandes générées par la commande kubeadm init. Eg :

```
mkdir -p $HOME/.kube && sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config && sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

3. Ajoutez les nœuds au cluster en utilisant la commande kubeadm join sur chaque machine en utilisant les informations générées par la commande kubeadm init sur le contrôleur de cluster.
4. Vérifiez que le cluster est opérationnel en exécutant des commandes de base de kubect.

PRATIQUE KUBE
DÉPLOIEMENT D'UNE APPLICATION SUR UN CLUSTER KUBERNETES

- Déploiement d'une application sur un cluster Kubernetes :
- **Voir "Kubernetes - Commandes" fourni par le formateur.**

- Utilisation de kubectl pour créer des objets tels que des pods, des services, des déploiements, etc.
- Utilisation de fichiers YAML pour décrire les objets Kubernetes à créer.
  - Déployer une application de conteneur simple telle que "hello-world" en utilisant kubectl et un fichier YAML.

https://github.com/ahmetb/kubectx

PRATIQUE KUBE
SCALING

- Scaling:

- Utilisation de kubectl pour redimensionner les déploiements en ajoutant ou en supprimant des réplicas
- Utilisation de HPA (Horizontal Pod Autoscaler) pour automatiser le redimensionnement des déploiements en fonction de la charge CPU ou de la mémoire
- Redimensionner un déploiement en ajoutant ou en supprimant des réplicas et vérifier les changements en utilisant kubectl.
- Configurer un HPA pour un déploiement de leur choix et observer comment le nombre de réplicas varie en fonction de la charge CPU ou de la mémoire.

PRATIQUE KUBE
MISES À JOUR EN ROULEMENT:

- Mises à jour en roulement:

- Utilisation de kubectl pour mettre à jour les images de conteneur dans un déploiement
- Utilisation de la stratégie de mise à jour en roulement pour effectuer les mises à jour de manière sûre et contrôlée
- Mettre à jour l'image d'un conteneur dans un déploiement en utilisant kubectl et la stratégie de mise à jour en roulement.
- Mettre à jour l'image d'un conteneur dans un déploiement de leur choix en utilisant kubectl et la stratégie de mise à jour en roulement.

LES COMPOSANTS KUBERNETES

- Voir "Kubernetes - Composants" fourni par le formateur.

LES COMPOSANTS KUBERNETES

- Voir "Kubernetes - Configuration" fourni par le formateur et effectuer les manipulations.

LES COMPOSANTS KUBERNETES

- https://gitlab.com/lucj/k8s-exercices/-/tree/master