kubernetes

k8s

Concepts de Kubernetes

- Kubernetes est un système de gestion de conteneurs
- Il exécute et gère les applications conteneurisées sur un cluster
- Qu'est-ce que cela signifie vraiment?

Choses de base que nous pouvons demander à Kubernetes de faire

- Démarrer 5 conteneurs en utilisant l'image eshop/api:v1.3
- Placer un équilibreur de charge interne devant ces conteneurs
- Démarrer 10 conteneurs en utilisant l'image eshop/webfront:v1.3
- Placer un équilibreur de charge public devant ces conteneurs
- C'est période de soldes, les pics de trafic, développer notre cluster et ajouter des conteneurs
- Nouvelle version! Remplacer mes conteneurs par la nouvelle image eshop/webfront:v1.4
- Continuer à traiter les demandes pendant la mise à niveau; mettre à jour mes conteneurs un par un

Autres choses que Kubernetes peut faire

- Mise à l'échelle automatique de base (autoscaling)
- Déploiement bleu/vert, déploiement Canary
- Services de longue durée, mais aussi travaux par lots (ponctuels)
- Surengager notre cluster et éjecter les tâches à faible priorité
- Exécuter des services avec des données avec état (bases de données, etc.)
- Contrôle d'accès précis définissant ce qui peut être fait par qui sur quelles ressources
- Intégration de services tiers (catalogue de services)
- Automatiser des tâches complexes (opérateurs)

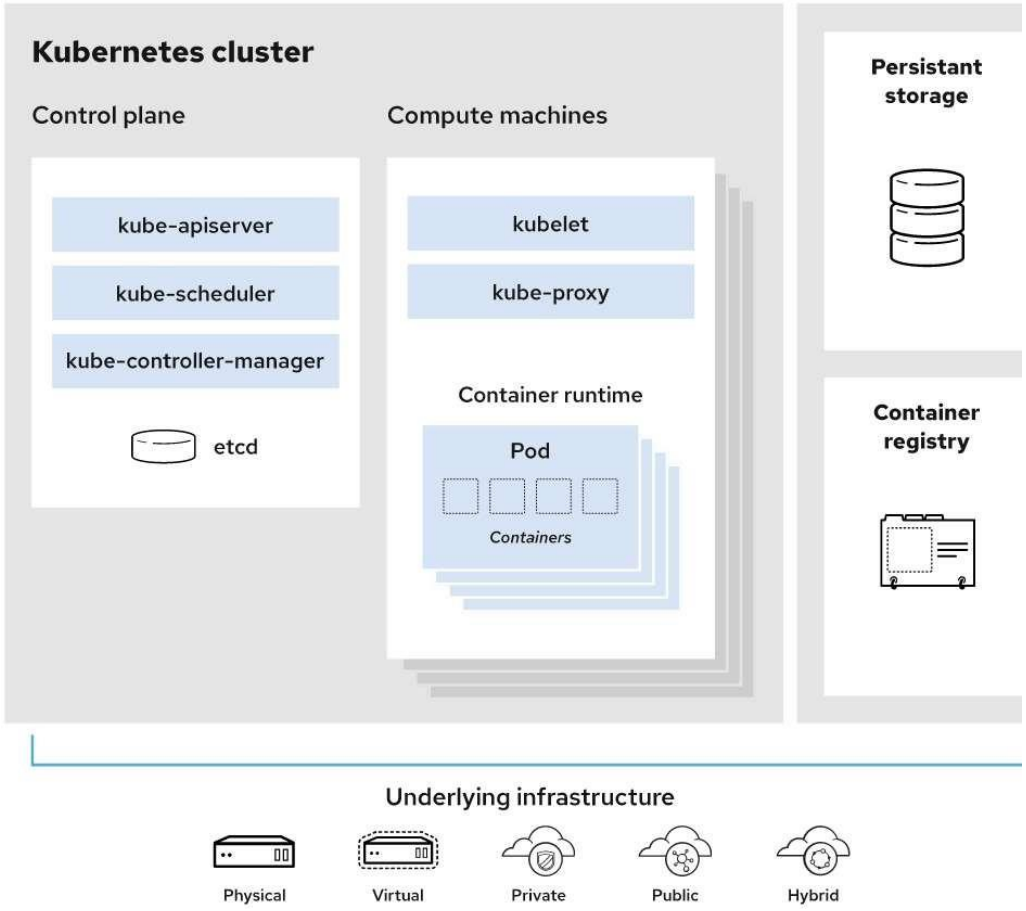
Fonctionnalités

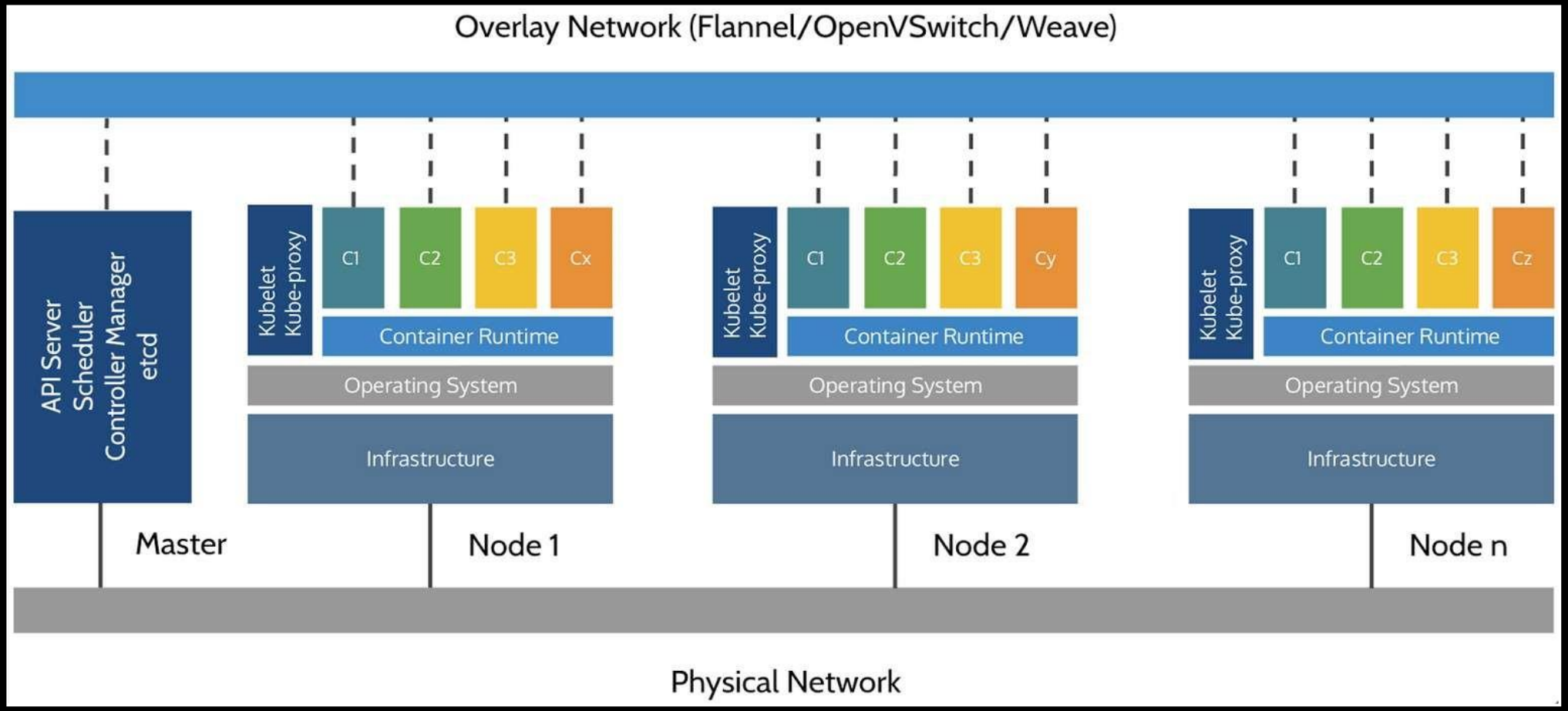
- **Découverte de service et équilibrage de charge** Kubernetes peut exposer un conteneur en utilisant le nom DNS ou en utilisant sa propre adresse IP. Si le trafic vers un conteneur est élevé, Kubernetes est capable d'équilibrer la charge et de distribuer le trafic réseau afin que le déploiement soit stable.
- **Orchestration du stockage** Kubernetes vous permet de monter automatiquement un système de stockage de votre choix, tel que des stockages locaux, des fournisseurs de cloud public, etc.
- **Déploiements et annulations automatisés** Vous pouvez décrire l'état souhaité pour vos conteneurs déployés à l'aide de Kubernetes, et il peut changer l'état réel à l'état souhaité à une vitesse contrôlée. Par exemple, vous pouvez automatiser Kubernetes pour créer de nouveaux conteneurs pour votre déploiement, supprimer des conteneurs existants et adopter toutes leurs ressources dans le nouveau conteneur.

Fonctionnalités

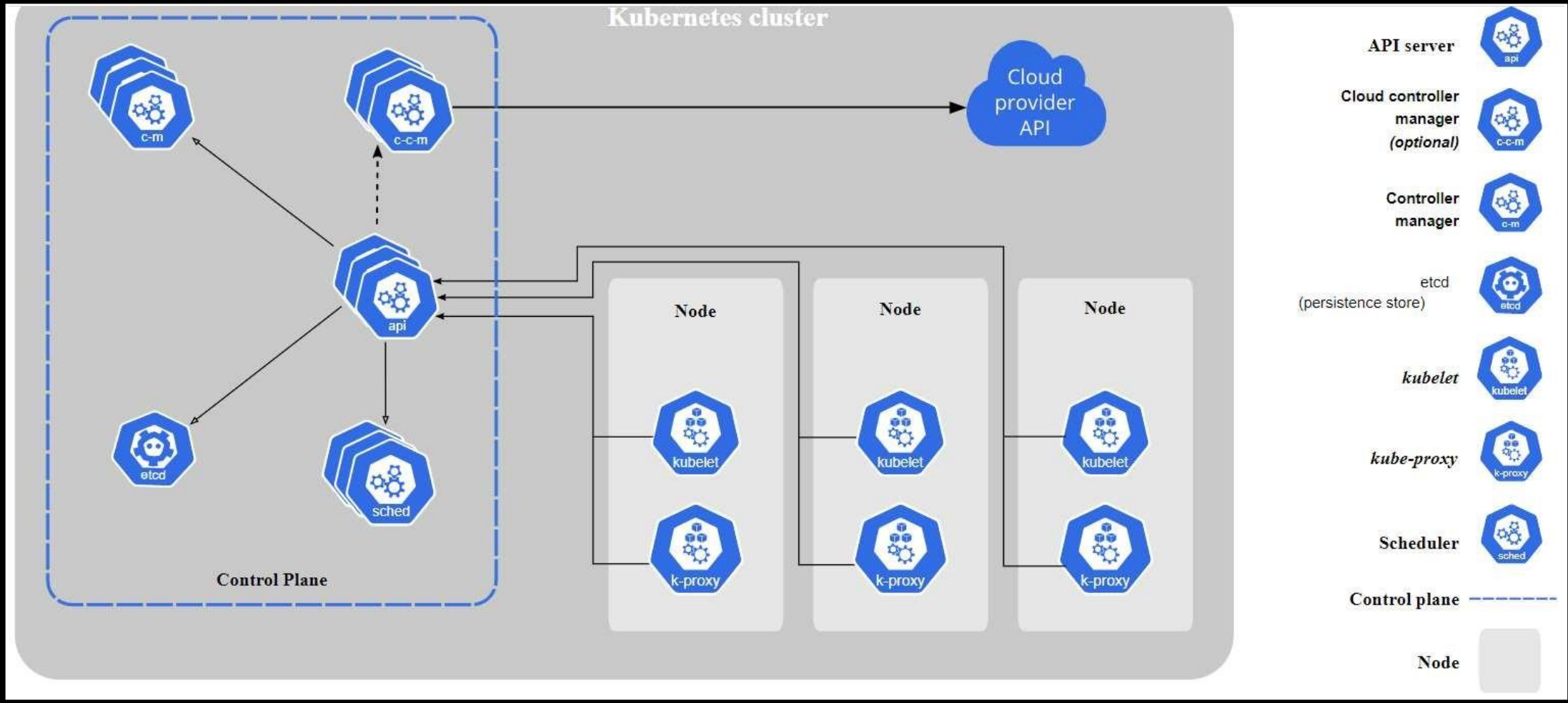
- **Empaquetage** automatique de binaires Vous fournissez à Kubernetes un cluster de nœuds qu'il peut utiliser pour exécuter des tâches en conteneur. Vous indiquez à Kubernetes la quantité de CPU et de mémoire (RAM) dont chaque conteneur a besoin. Kubernetes peut installer des conteneurs sur vos nœuds pour tirer le meilleur parti de vos ressources.
- **L'auto- réparation** Kubernetes redémarre les conteneurs qui échouent, remplace les conteneurs, tue les conteneurs qui ne répondent pas à votre vérification de l'état définie par l'utilisateur et ne les annonce pas aux clients tant qu'ils ne sont pas prêts à être diffusés.
- **Gestion des secrets et de la configuration** Kubernetes vous permet de stocker et de gérer des informations sensibles, telles que les mots de passe, les jetons OAuth et les clés SSH. Vous pouvez déployer et mettre à jour les secrets et la configuration de l'application sans reconstruire vos images de conteneur et sans exposer les secrets dans la configuration de votre pile.

Kubernetes Cluster





Architecture de Kubernetes



Composants de Kubernetes

Composants du plan de contrôle

- Les composants du plan de contrôle prennent des décisions globales concernant le cluster (par exemple, la planification), ainsi que la détection et la réponse aux événements du cluster (par exemple, le démarrage d'un nouveau pod lorsque le champ *replicas* d'un déploiement n'est pas satisfait).
- Les composants du plan de contrôle peuvent être exécutés sur n'importe quelle machine du cluster.
- Pour plus de simplicité, les scripts de configuration démarrent généralement tous les composants du plan de contrôle sur la même machine et n'exécutent pas de conteneurs utilisateur sur cette machine.
- **Le plan de contrôle est également appelé le "master"**

kube-apiserver

- Le serveur API est un composant du plan de contrôle de Kubernetes qui expose l'API Kubernetes.
- Le serveur API est le frontal du plan de contrôle Kubernetes.
- La principale implémentation d'un serveur d'API Kubernetes est kube-apiserver .
- kube-apiserver est conçu pour évoluer horizontalement, c'est-à-dire qu'il évolue en déployant plus d'instances.
- Vous pouvez exécuter plusieurs instances de kube-apiserver et équilibrer le trafic entre ces instances.

etcd

Magasin de valeurs de clé cohérent et hautement disponible utilisé comme magasin de sauvegarde de Kubernetes pour toutes les données du cluster.

Si votre cluster Kubernetes utilise etcd comme magasin de sauvegarde, assurez-vous d'avoir un plan de sauvegarde pour ces données.

kube-scheduler

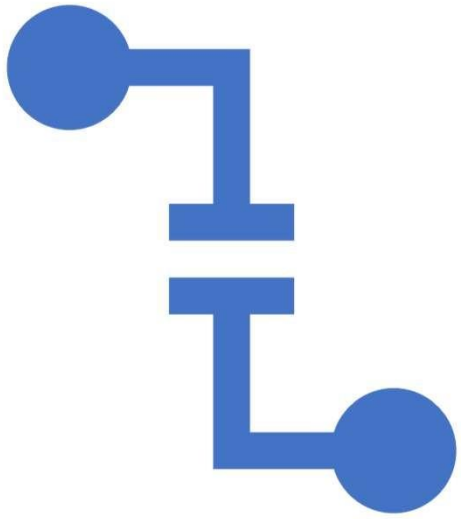
- Composant du plan de contrôle qui surveille les nouveaux Pods sans nœud assigné et sélectionne un nœud sur lequel les exécuter.
- Les facteurs pris en compte pour les décisions de planification comprennent: les besoins en ressources individuelles et collectives, les contraintes matérielles / logicielles / politiques, les spécifications d'affinité et d'anti-affinité, la localisation des données, l'interférence entre les charges de travail et les délais.

kube-controller-manager

- Composants du Plan de Controle qui exécutent les processus de controle.
- Logiquement, chacun controleur est un processus distinct, mais pour réduire la complexité, ils sont tous compilés en un seul binaire et exécutés dans un seul processus.
- Ces contrôleurs comprennent:
  - Contrôleur de nœud: responsable de remarquer et de répondre lorsque les nœuds tombent en panne.
  - Contrôleur de réplication: responsable de la gestion du nombre correct de pods pour chaque objet contrôleur de réplication dans le système.
  - Contrôleur Endpoints: remplit l'objet Endpoints (c'est-à-dire joint les services et les pods).
  - Contrôleurs de compte de service et de jetons: créez des comptes par défaut et des jetons d'accès API pour les nouveaux espaces de noms.

cloud-controller-manager

- Un plan de contrôle Kubernetes qui intègre une logique de contrôle spécifique au cloud. Le gestionnaire de contrôleur cloud vous permet de lier votre cluster à l'API de votre fournisseur cloud et sépare les composants qui interagissent avec cette plate-forme cloud des composants qui interagissent simplement avec votre cluster.
- Le cloud-controller-manager n'exécute que les contrôleurs spécifiques à votre fournisseur de cloud. Si vous exécutez Kubernetes dans vos propres locaux ou dans un environnement d'apprentissage à l'intérieur de votre propre PC, le cluster n'a pas de gestionnaire de contrôleur cloud.
- Comme avec le kube-controller-manager, le cloud-controller-manager combine plusieurs boucles de contrôle logiquement indépendantes en un seul binaire que vous exécutez en tant que processus unique. Vous pouvez mettre à l'échelle horizontalement (exécuter plusieurs copies) pour améliorer les performances ou pour aider à tolérer les échecs.



cloud-controller-manager

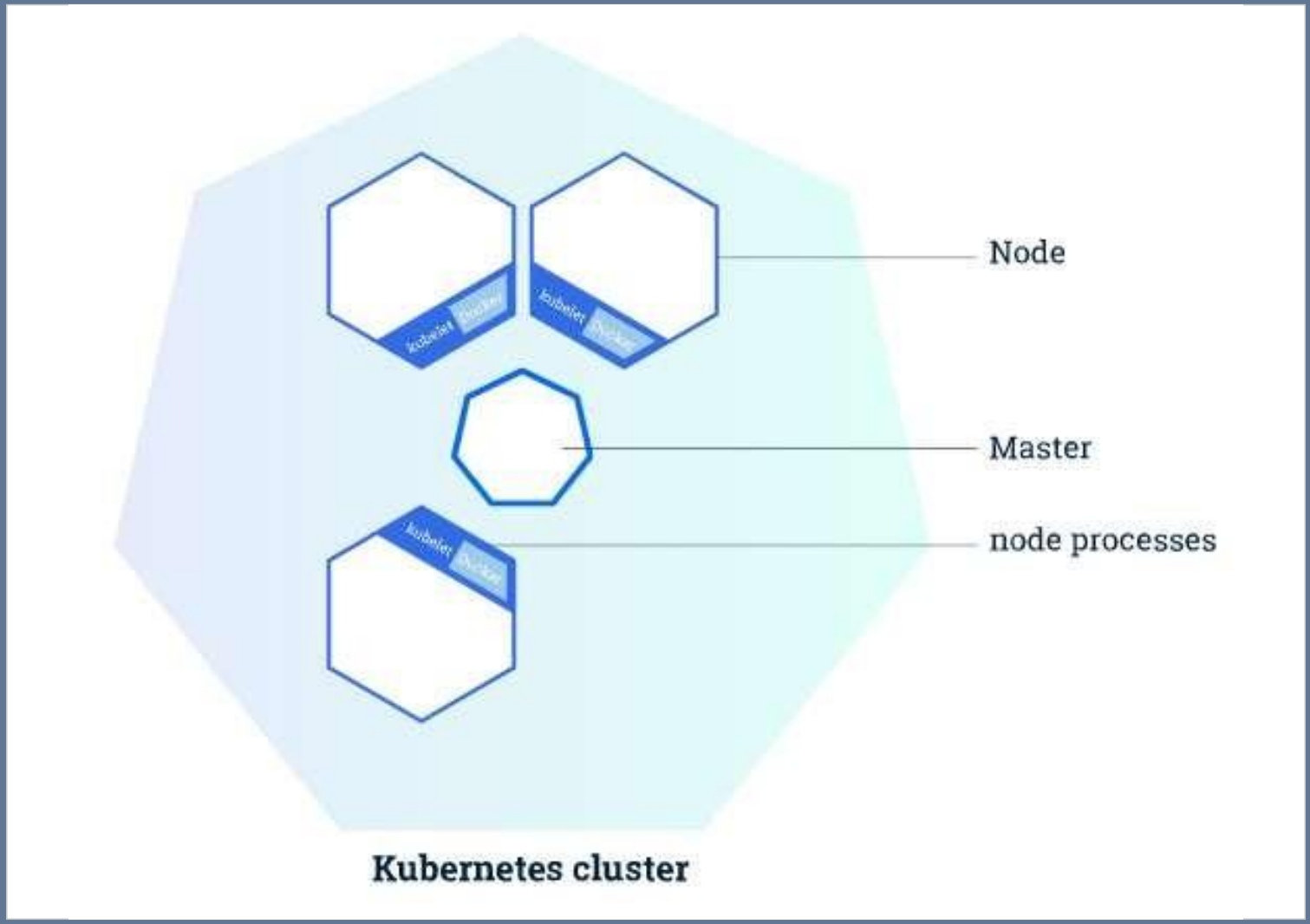
- Les contrôleurs suivants peuvent avoir des dépendances de fournisseur de cloud:
  - Contrôleur de nœud: pour vérifier le fournisseur de cloud afin de déterminer si un nœud a été supprimé dans le cloud après avoir cessé de répondre
  - Contrôleur de route: pour configurer des routes dans l'infrastructure cloud sous-jacente
  - Contrôleur de service: pour créer, mettre à jour et supprimer des équilibreurs de charge de fournisseur de cloud

Les nœuds

- Les nœuds exécutant nos conteneurs exécutent une collection de services:
  - un moteur de conteneur (généralement Docker)
  - kubelet (l'agent du nœud)
  - kube-proxy (un composant réseau)
- Les nœuds étaient anciennement appelés "minions"

Composants du nœud

- **kubelet**
  - Un agent qui s'exécute sur chacun nœud dans le cluster. Il s'assure que les conteneurs s’exécutent dans un Pod.
  - Le kubelet prend un ensemble de PodSpecs qui sont fournis via divers mécanismes et garantit que les conteneurs décrits dans ces PodSpecs sont en cours d'exécution et sains. Le kubelet ne gère pas les conteneurs qui n'ont pas été créés par Kubernetes.
- **kube-proxy**
  - kube-proxy est un proxy réseau qui s'exécute sur chaque nœud dans votre cluster.
  - kube-proxy maintient les règles de réseau sur les nœuds. Ces règles réseau permettent la communication réseau avec vos pods à partir de sessions réseau à l'intérieur ou à l'extérieur de votre cluster.
  - kube-proxy utilise la couche de filtrage de paquets du système d'exploitation s'il en existe une et qu'elle est disponible. Sinon, kube-proxy transfère le trafic lui-même.
- **Container runtime**

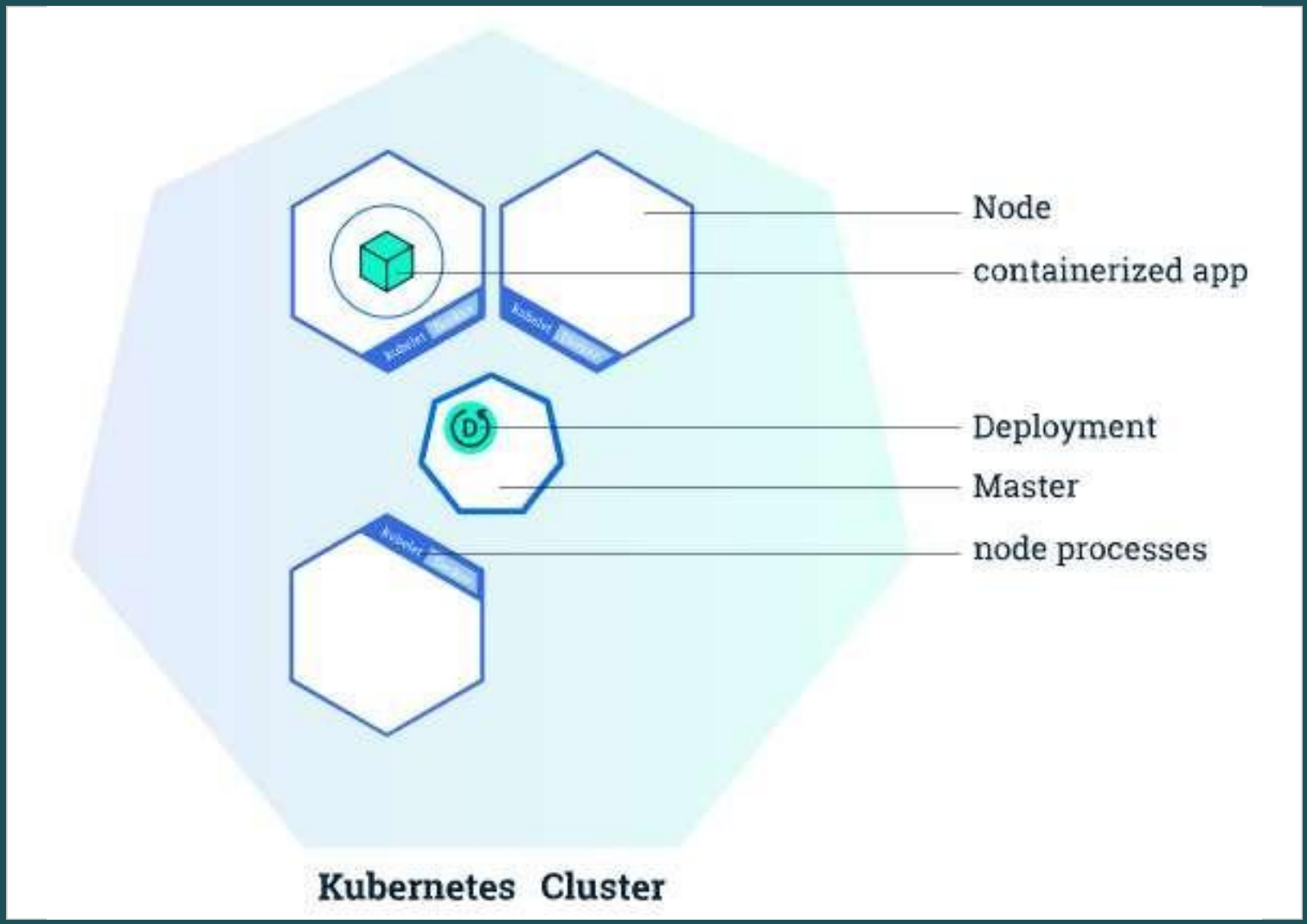


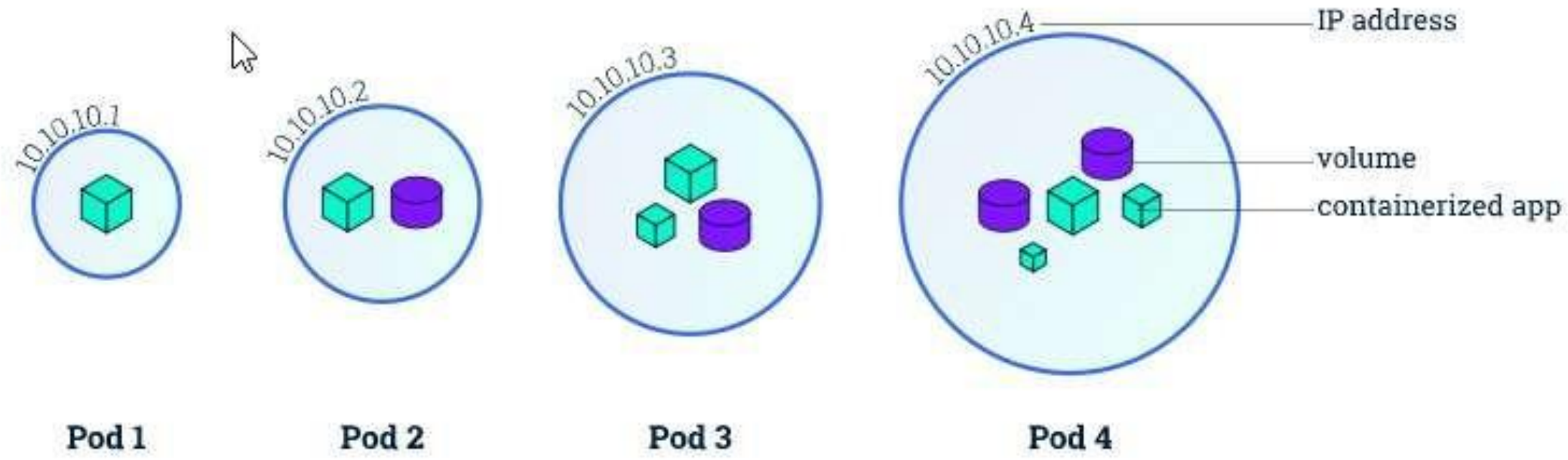
Cluster Kubernetes

Interagir avec Kubernetes

- Nous interagissons avec un cluster Kubernetes via l'API Kubernetes
- L'API Kubernetes est (principalement) RESTful
- Elle nous permet de créer, lire, mettre à jour, supprimer des ressources
- Quelques types de ressources courants sont:
  - **node** (une machine - physique ou virtuelle - dans notre cluster)
  - **pod** (groupe de conteneurs fonctionnant ensemble sur un nœud)
  - **service** (point de terminaison réseau stable pour se connecter à un ou plusieurs conteneurs)

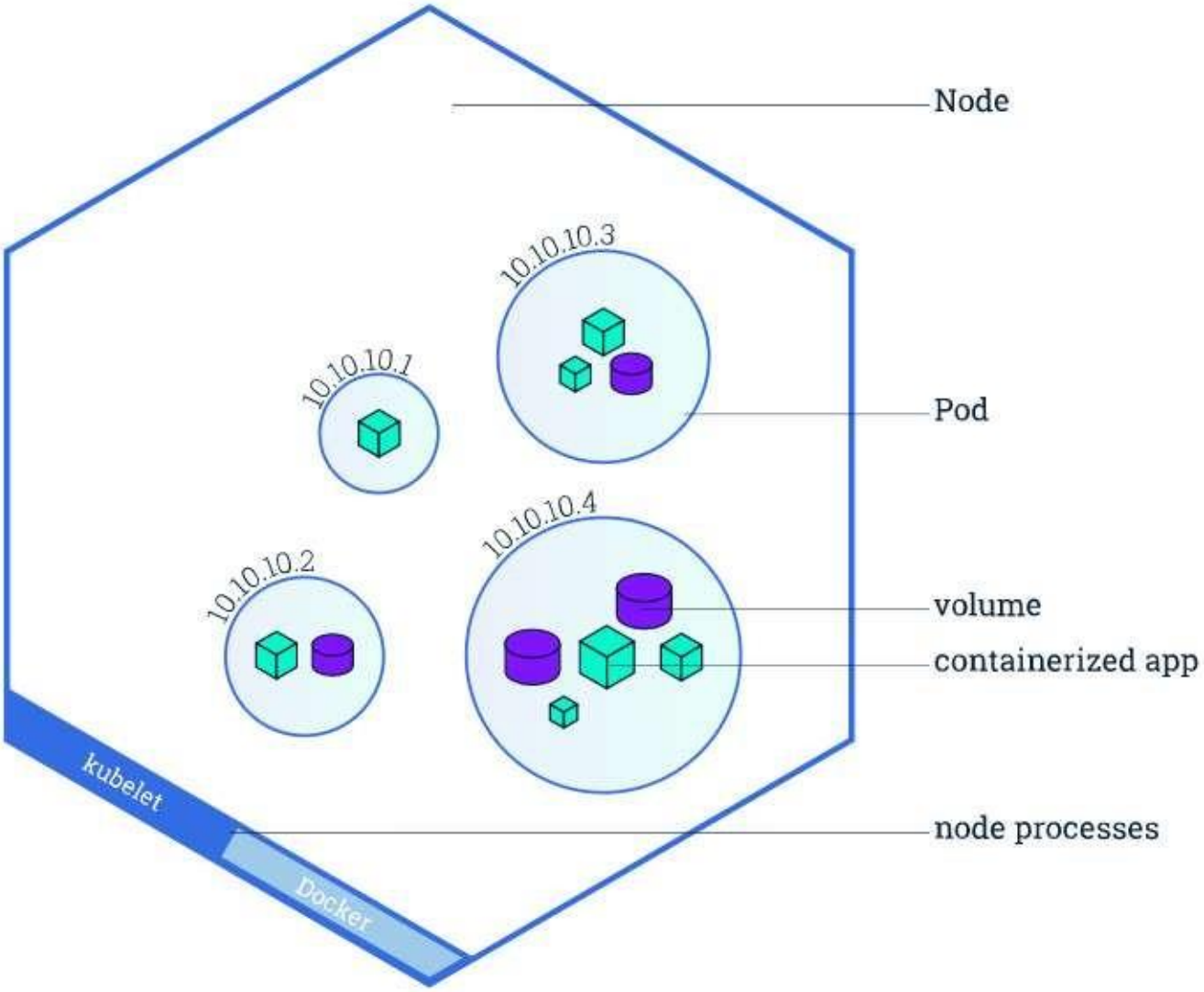
Déploiement d'une application sur Kubernetes

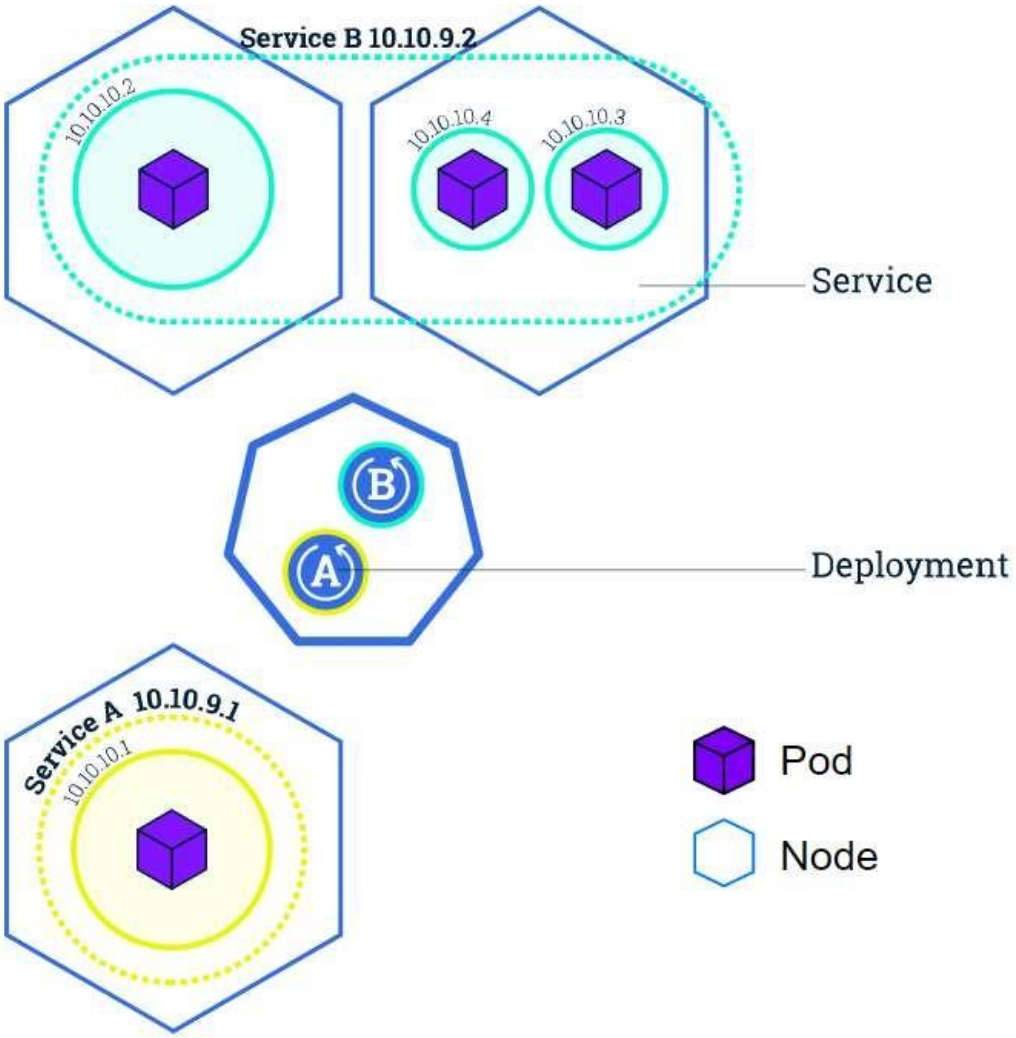



10.10.10.1
10.10.10.2
10.10.10.3
10.10.10.4
IP address
volume
containerized app
Pod 1
Pod 2
Pod 3
Pod 4

PODs

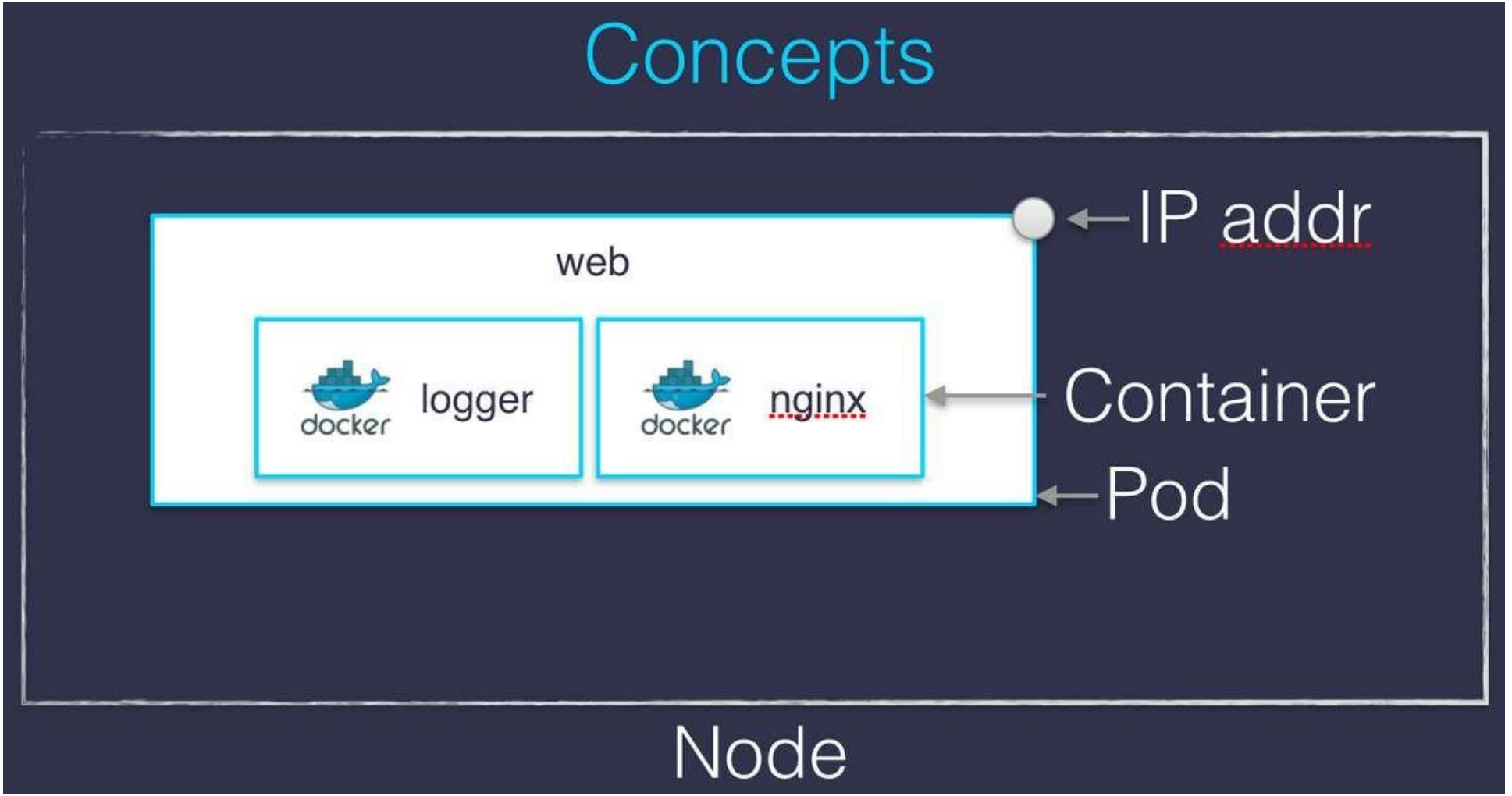
Node


Node
10.10.10.3
10.10.10.1
Pod
10.10.10.4
10.10.10.2
volume
containerized app
kubelet
Docker
node processes



Services & Labels


Network
Request


Concepts
IP addr
web
logger
docker
nginx
docker
Container
Pod
Node