Kubernetes

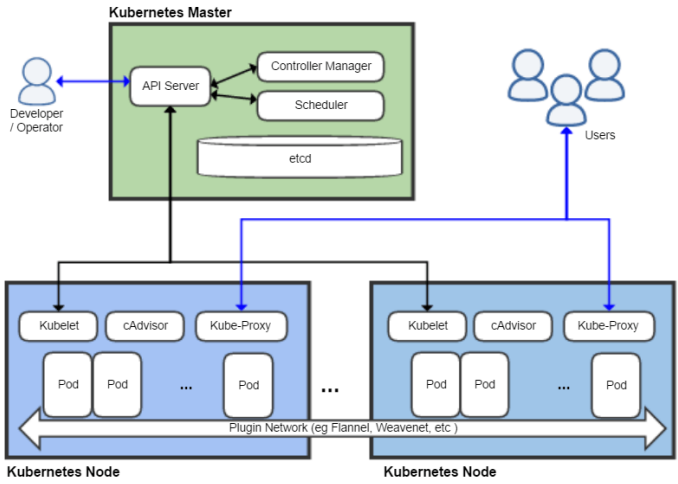
1 Introduction

Kubernetes vise à fournir une « plate-forme permettant d'automatiser le déploiement, la montée en charge et la mise en oeuvre de conteneurs d'applications sur des clusters de serveurs.

Un cluster kubernetes comporte deux types de ressources :

— un master qui contrôle le cluster

— des nodes qui sont les éléments chargés d'exécuter les containers et applications



Nodes : Un node est une machine virtuelle ou physique qui exécutent les conteneurs. Sur un node on retrouve différent services :

— **kubelet** veille aux bon fonctionnement du noeud sur lequel il s'exécute et retourne ces informations aux serveurs maître.

— **kube-proxy** gère le traffic réseau et les règles définies.

— **Docker** exécute les conteneurs.

— **fluentd** se charge de transférer les logs vers les serveurs maîtres.

Master : sur un master on retrouve différent services :

— **kube-apiserver** qui est scalable horizontalement, se charge d'exposer les apis de Kubernetes a l'extérieur du cluster. C'est le front-end des interfaces de configuration de Kubernetes.

— **etcd** stocke les fichiers de configurations du cluster, comme le répertoire /etc sur Linux. Et comme pour les fichiers de configuration dans /etc, une bonne pratique consiste à avoir un backup des fichiers gérés par etcd. Ceci permettra de remettre en route plus rapidement un cluster après incident majeur.

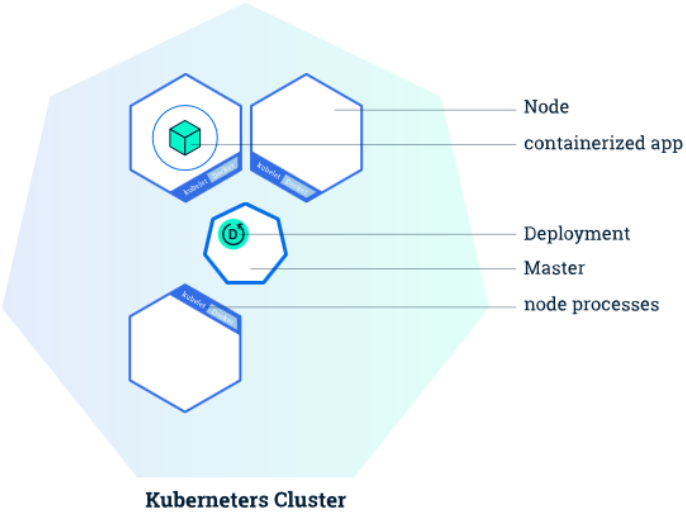
— **kube-controller-manager** exécute plusieurs controlleurs chargés de vérifier : l'état des noeuds, le bon nombre de replicas de pods dans le cluster, lier les services aux pods ou encore gérer les droits d'accès aux différents espaces de noms utilisés dans le cluster.
— **kube-scheduler** s'occupe d'assigner un noeud ou plusieurs à un pod fraichement créé.
— **DNS** intégré a Kubernetes. Les conteneurs utilisent ce service dès leur création.
— **Container Resource Monitoring** permettant de centraliser les données métriques envoyées par les conteneurs, puis de les afficher sous forme de graphes.
— **Cluster-level logging** gère les logs retournés par les conteneurs dans le but de surveiller le comportement des applications et microservices en cours.

Pods : L'unité de base de l'ordonnancement dans Kubernetes est appelé "pod". C'est une vue abstraite de composants conteneurisés. Un pod consiste en un ou plusieurs conteneurs qui ont la garantie d'être co-localisés sur une machine hôte et peuvent en partager les ressources18. Chaque pod dans Kubernetes possède une adresse IP unique (à l'intérieur du cluster), qui permet aux applications d'utiliser les ports de la machine sans risque de conflit. Un Pod est lié à un node et peut être répliqué.

Kubernetes permet à des clients (utilisateurs et composants internes) d'attacher des paires clés-valeurs appelées "labels" à n'importe quel objet d'API dans le système, par exemple les pods et les nodes.

1.1 Kubernetes Deployments

Un déploiement permet a Kubernetes de créer et mettre a jour les instances des applications.



Obtenir la liste des nodes :

```
kubectl get nodes
```

Obtenir les informations des nodes :

```
kubectl describe
```

Obtenir les logs des pods :

```
kubectl logs
```

Exécuter une commande dans un container d'un pod :

```
kubectl exec -it $POD_NAME bash
```

Créer un déploiement :

```
kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/
    kubernetes-bootcamp:v1 --port=8080
```

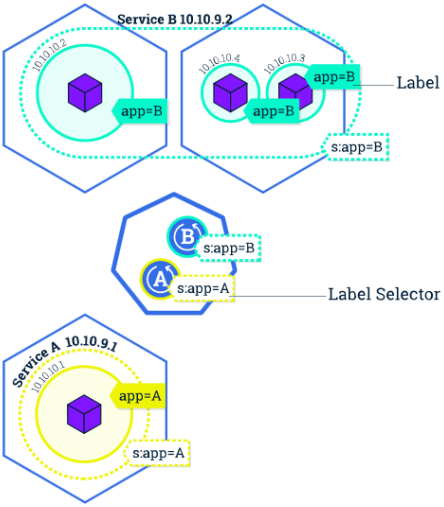
Lister les déploiement :

```
kubectl get deployments
```

2 Services

Les pods kubernetes sont mortels et ont un cycle de vie. Quand un node s'arrête, les pods s'arrêtent aussi.
Un service est défini avec un fichier YAML. Les pods concernés par ce service sont déterminés grâce au Laleb Selector.

— découverte, routage, assignation d'IP
— équilibrage de charge avec round-robin connexions réseau
— n'est pas lié au réplication controller



3 Réplication

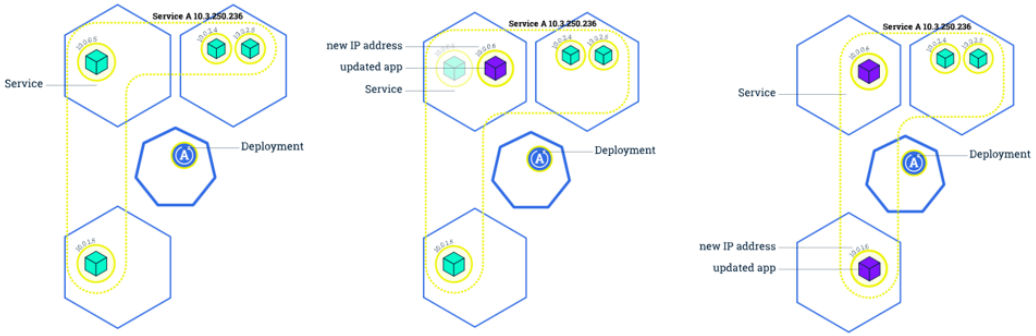
La réplication de pods permet de s'assurer que le service est toujours disponible peu importe la charge du système.

Répliquer des pods : préciser le nom du déploiement et le nombre de réplicats.

```
kubectl scale deployments/kubernetes-bootcamp --replicas=4
```

4 Mise à jour

Le service se charge de la répartition de charge entre les pods disponibles.



5 Configuration

Créer un configMap

```
kubectl create configmap example-redis-config --from-file=docs
    /user-guide/configmap/redis/redis-config
```

Créer une spécification de pod :

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: kubernetes/redis:v1
    env:
    - name: MASTER
      value: "true"
    ports:
```

```
    - containerPort: 6379
    resources:
      limits:
        cpu: "0.1"
    volumeMounts:
    - mountPath: /redis-master-data
      name: data
    - mountPath: /redis-master
      name: config
  volumes:
    - name: data
      emptyDir: {}
    - name: config
      configMap:
        name: example-redis-config
        items:
        - key: redis-config
          path: redis.conf
```

Créer le pod :

```
kubectl create -f docs/user-guide/configmap/redis/redis-pod.
    yaml
```