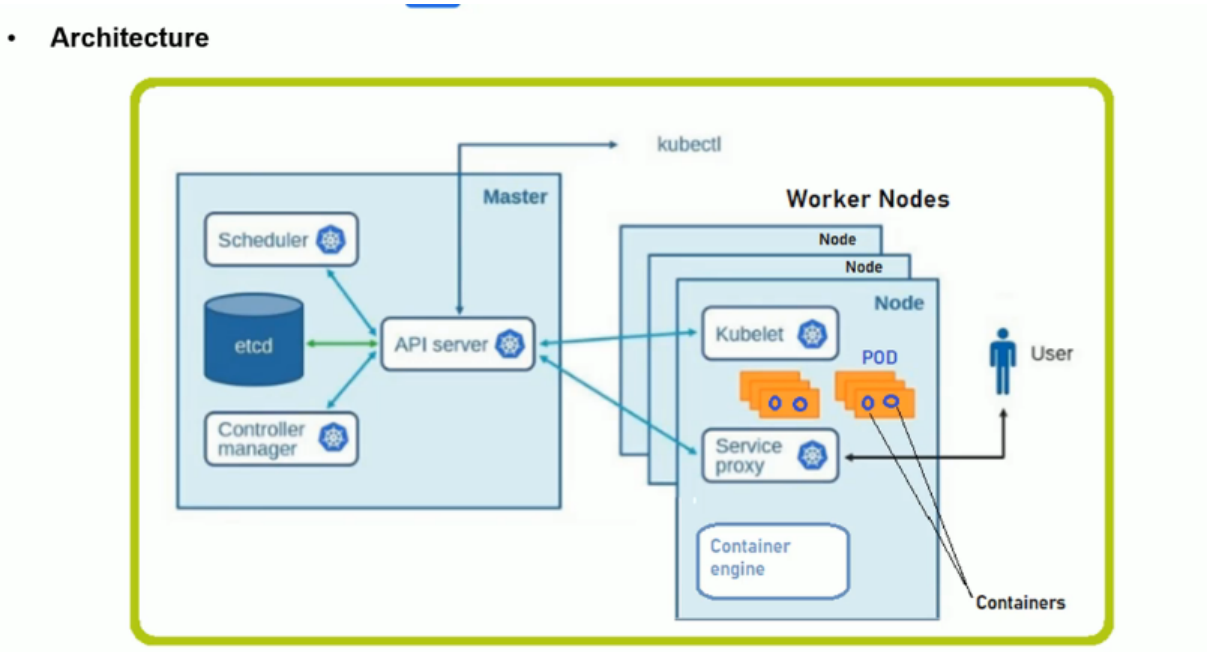
Architecture



composants pour donner une vue plus complète de l'architecture de Kubernetes (K8s):

1. **Nœud Maître (Master Node)**: Il s'agit du cerveau de l'ensemble du système qui gère l'état global du cluster.
    - `API Server`: Point d'entrée pour toutes les commandes REST dans un cluster Kubernetes. Il valide et traite les requêtes, et orchestre la communication entre les différents composants.
    - `etcd`: Base de données distribuée clé-valeur qui stocke la configuration et l'état du cluster Kubernetes. Il est utilisé par le nœud maître pour maintenir une cohérence des données à travers le cluster, et il est essentiel pour la haute disponibilité et la fiabilité du système.
    - `Scheduler (Planificateur)`: Attribue les Pods aux nœuds travailleurs en fonction de divers critères comme les ressources disponibles et les contraintes. Il joue un rôle clé dans la gestion de l'équilibrage de charge au sein du cluster.
    - `Controller Manager`: Contient les contrôleurs pour gérer différents aspects du cluster. Composant du nœud maître qui exécute différents contrôleurs, tels

que le contrôleur de nœud et le contrôleur de réplication. Ces contrôleurs assurent le bon fonctionnement et l'état désiré du cluster Kubernetes.

- `Cloud Controller Manager`: Permet d'interagir avec les fournisseurs de services cloud sous-jacents.

2. **Nœud Travailleur (Worker Node)**: Les serveurs où les applications sont réellement exécutées.
   - `Kubelet`: Agent qui s'exécute sur chaque nœud travailleur pour s'assurer que les conteneurs dans un Pod sont en cours d'exécution comme prévu. Il communique avec le nœud maître pour recevoir les instructions et rapporter l'état des Pods.
   - **Container Runtime**: Environnement d'exécution des conteneurs (Docker, containerd, etc.).
   - `Kube-proxy`: Routeur qui prend en charge l'accès réseau aux services. Composant qui s'exécute sur chaque nœud travailleur pour gérer les règles de réseau et effectuer la redirection du trafic vers les conteneurs. Il permet ainsi la communication efficace entre les Pods et les services externes.
3. `Pods`: Les unités de base dans Kubernetes qui hébergent un ou plusieurs conteneurs. Ils partagent le même espace réseau et peuvent communiquer entre eux facilement. Les Pods sont éphémères et sont gérés par des contrôleurs pour maintenir l'état désiré.
4. `Services`: Abstraient la façon dont le réseau interagit avec un ensemble de Pods.
5. **Ingress Controllers et Resources**: Gèrent l'accès externe aux services dans un cluster.
6. `ConfigMaps et Secrets`: Objets utilisés pour gérer les données de configuration et les informations sensibles dans un cluster Kubernetes. ConfigMaps stockent des paramètres non-confidentiels, tandis que Secrets sont utilisés pour stocker des données sensibles comme les mots de passe et les clés API.
7. **Volume**: Permet un stockage persistant pour les Pods.
8. **StatefulSets, Deployments, DaemonSets**: Contrôleurs qui gèrent différents types de déploiements de Pods.
9. **Namespace**: Espaces de noms pour segmenter les ressources du cluster.
10. **Roles et RoleBindings**: Pour définir des autorisations au sein du cluster.

11. **Custom Resource Definitions (CRDs)**: Étendent les ressources disponibles dans l'API de Kubernetes.
12. **Service Account**: Identités que les Pods peuvent utiliser pour accéder à l'API Kubernetes.
13. **Network Policies**: Définissent comment les Pods communiquent entre eux.
14. `Horizontal Pod Autoscaler`: Ajuste automatiquement le nombre de Pods dans un déploiement ou un replica set.
15. `Vertical Pod Autoscaler`: Ajuste automatiquement les ressources de calcul allouées aux Pods.
16. **Affinity et Anti-Affinity**: Règles pour déterminer comment les Pods sont placés par rapport les uns aux autres.
17. **Taints et Tolerations**: Permettent de contrôler où les Pods peuvent ou ne peuvent pas être programmés.
18. **Operators**: Permettent la gestion personnalisée des applications dans le cluster.
19. **Jobs et CronJobs**: Exécutent des tâches ponctuelles ou récurrentes.

Ces composants interagissent ensemble pour fournir une plate-forme robuste et extensible pour gérer des applications conteneurisées à grande échelle.