Documentation K8S et choix technique :

Ce guide a pour objectif de clarifier pourquoi nous avons choisi Kubernetes (K8s) plutôt que Minikube, et de détailler les étapes de mise en place de notre infrastructure K8s en utilisant Vagrant et CRI-O comme gestionnaire de conteneurs…

Qu'est-ce que Kubeadm ?

Kubeadm est un outil permettant de mettre en place un cluster Kubernetes minimum viable sans configuration trop complexe. De plus, Kubeadm facilite l'ensemble du processus en exécutant une série de vérifications préalables pour s'assurer que le serveur dispose de tous les composants et configurations essentiels pour exécuter Kubernetes.

Minikube est un outil **local** principalement utilisé pour **développer et tester Kubernetes** sur une seule machine. Il permet de créer un cluster Kubernetes sur une seule machine virtuelle ou un environnement local pour les tests rapides, le développement et l'apprentissage de Kubernetes.

Pourquoi utiliser Kubeadm plutôt que Minikube ?

Kubeadm et Minikube sont tous deux des outils utilisés dans le cadre de Kubernetes, mais ils répondent à des besoins différents.

Voici les raisons pour lesquelles vous pourriez préférer Kubeadm à Minikube :

Caractéristique	Minikube	Kubeadm
Objectif principal	Développement et tests locaux sur un seul nœud.	Déploiement d'un cluster Kubernetes multi-nœuds en production.
Environnement cible	Local (machine unique, souvent sur une VM ou un conteneur).	Environnements distribués (machines physiques ou virtuelles).
Utilisation en production	Non recommandé pour la production. Idéal pour le développement et l'apprentissage.	Conçu pour la production et les environnements de test réalistes.
Facilité de configuration	Très simple à configurer. Installation rapide en local.	Plus complexe, nécessite une configuration manuelle pour plusieurs nœuds, mais offre plus de contrôle.
Sécurité et gestion avancée	Limitée, surtout pour des tests locaux ou des démonstrations simples.	Offre des options avancées de sécurité et de gestion (certificats, tokens, etc.)
Runtime de conteneurs	Utilise Docker ou d'autres runtimes locaux comme containerd.	Utilise n'importe quel runtime compatible (Docker, containerd, CRI-O).
Support CI/CD	imité pour des environnements CI/CD à grande échelle.	Adapté pour être intégré dans des pipelines CI/CD et des environnements complexes de déploiement.

Minikube est idéal pour les **développeurs** ou **apprenants** qui veulent rapidement tester Kubernetes sur leur machine locale. Il est simple, rapide à configurer et utile pour des tests de petites applications.
Kubeadm est conçu pour **les environnements de production**, avec des fonctionnalités avancées pour gérer des clusters Kubernetes complexes, distribués et évolutifs. Il est plus adapté aux environnements multi-nœuds et aux exigences des entreprises.

Quelles sont les conditions préalables à la configuration de Kubeadm ?

Voici les conditions préalables à la configuration du cluster Kubeadm Kubernetes.

Minimum deux nœuds Ubuntu [un nœud maître et un nœud de travail]. Vous pouvez avoir plus de nœuds de travail selon vos besoins.

Le nœud maître doit disposer d'un minimum de 2 vCPU et de 2 Go de RAM.

Pour les nœuds de travail, un minimum de 1 vCPU et 2 Go de RAM est recommandé.

10.X.X.X/X gamme réseau avec adresses IP statiques pour les nœuds maître et de travail. Nous utiliserons la série 192.x.x.x comme plage de réseau de pods qui sera utilisée par le plugin de réseau Calico. Assurez-vous que la plage d'adresses IP du nœud et la plage d'adresses IP du pod ne se chevauchent pas.

Remarque : Si vous configurez le cluster dans le réseau d'entreprise derrière un proxy, assurez-vous de définir les variables proxy et d'avoir accès au registre de conteneurs et au hub Docker. Vous pouvez également contacter votre administrateur réseau pour ajouter registry.k8s.io liste blanche afin d'extraire les images requises.

Exigences du port Kubeadm

Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

Pour plus des d'détails visiter : https://shorturl.at/E8uah

Pourquoi utiliser Vagrant pour construire l'infrastructure Kubernetes ?

Vagrant est utilisé pour faciliter la gestion des machines virtuelles (VMs) et permet de déployer une infrastructure reproductible et consistante. Dans votre cas, Vagrant va être utilisé pour créer des machines virtuelles qui serviront de nœuds dans votre cluster Kubernetes. Cela présente plusieurs avantages :

Automatisation et reproductibilité : Vagrant permet de définir une configuration d'infrastructure via des fichiers de configuration (Vagrantfile). Cela facilite la recréation rapide de l'infrastructure à tout moment.

Environnements isolés : Chaque VM peut être configurée avec des rôles spécifiques (master, worker) et des configurations réseaux, ce qui simule un véritable cluster Kubernetes multi-nœuds.

Flexibilité : Vous pouvez facilement configurer des nœuds maîtres (master) et des nœuds travailleurs (worker), tout en définissant des ressources pour chaque VM (mémoire, CPU).

Site officiel : Vagrant by HashiCorp

Line de téléchargement : Install | Vagrant | HashiCorp Developer

Choix du gestionnaire de conteneurs : Docker, containerd ou CRI-O ?

Dans Kubernetes, plusieurs options de runtime de conteneurs sont disponibles pour exécuter les conteneurs. Les principales options sont Docker, containerd et CRI-O.

Voici pourquoi nous avons choisi CRI-O dans votre projet :

Performance et légèreté : CRI-O est un runtime de conteneurs léger, conçu spécifiquement pour Kubernetes. Il est optimisé pour une intégration native avec K8s et offre de meilleures performances que Docker.

Conformité Kubernetes : CRI-O implémente le CRI (Container Runtime Interface) de manière stricte, ce qui garantit une meilleure compatibilité avec Kubernetes. En comparaison, Docker est un runtime générique et moins optimisé spécifiquement pour K8s.

Support pour les conteneurs OCI : CRI-O utilise les images de conteneurs conformes à l'Open Container Initiative (OCI), offrant ainsi une grande compatibilité avec différents types d'images de conteneurs.

Simplicité de gestion : CRI-O offre une gestion plus simple des conteneurs dans Kubernetes, sans les fonctionnalités supplémentaires de Docker qui ne sont pas nécessaires dans un environnement Kubernetes.

Site officiel CRI-O : cri-o

Utlistation CRI-O en kubeadm : cri-o/tutorials/kubeadm.md à main · cri-o/cri-o

Comment configurer un cluster Kubernetes à l'aide de Kubeadm

Voici des liens présent guides d'installation et configuration :

https://devopscube.com/setup-kubernetes-cluster-kubeadm/

Install Kubernetes Cluster on Ubuntu 22.04 using kubeadm | ComputingForGeeks

Deploy Kubernetes Cluster on AlmaLinux 8 with Kubeadm | ComputingForGeeks