KUBERNETES

Master IAO
Dept. Informatique

Pr. 0ussama Mohamed REDA
Intervenant:
Dr. Y.T.Benjelloune

Novembre 2020

Module Cloud Computing

Kubernetes

- Kubernetes est un orchestrateur de multiples conteneurs notamment des conteneurs dockers mais pas seulement docker (core os)
- Le but est le lancement orchestré avec des liens forts des conteneurs
- créer de l'abstraction avec la notion de service (pas seulement par IP)
- Maintenir la haute disponibilité (maintenir les conteneurs up et assurer l'etat des services qui ont été décrits)
- Scalability : lancer multiples instances pour un même service pour supporte des charges importantes
- Supporté par plusieurs fournisseurs qui s'adapte à kubernetes :vsphere vmware, google cloud, aws, azure, bare metal (sur une machine physique soit local en maison ou dans une entreprise)

Les modes de fonctionnement kubernetes

- Mode Cluster :un master et des nœuds esclaves
- Mode demo/test avec Minikube:
  - Minikube est une machine virtuelle, qui fait tourner un cluster à nœud unique
  - La machine virtuelle contient le boot2docker, c'est que docker est deja installé et en meme temps il va installer un cluster kubernetes local
  - dans ce mode le seul nœud joue le rôle de master de slave
  - Ce mode nécessite d'avoir VirtualBox
  - Image déployé automatiquement sur VB avec l'essentiel des packages:
    - Docker
    - Kubernetes
- Pour installer minikube:
  - https://kubernetes.io/docs/tasks/tools/install-minikube
- On lance le minikube: $ minikube start

Les notions et concepts de Kubernetes

- Nœuds:
  - serveurs physiques ou virtuels soit en mode master ou simples nœuds d'execution
- Pods: une instance de KBS : entitié de reference de k8s
  - Fournir un ensemble coherent de conteneurs qui peut etre un ou plusieurs conteneurs docker ou autre
  - Exemple d'un serveur wordpress et sa base de donnees.
  - Exemple d'un serveur et plusieurs replicas
- Services:
  - Le service se place au dessus des pods ce qui permet de faire une abstraction des pods
  - Le service évite la communication par IP comme dans le cas des docker (IP peut changer puisqu'on travaille avec les conteneurs)
  - Un service est compose d'un couple IP + Port fixe , qui permet de communiquer avec des conteneurs

Concepts de base de kubernetes

- Volumes :
  - persistent ou non persistent, ils constituent les lieux d'echange entre les pods
  - Persistent dans ce cas on va les stocker à l'exterieur des pods et non persistent à l'interieur des pods
  - Le choix des la persistence est fait qu'on on veut relancer des conteneurs (pods) sans perdre de data, sinon en cas de non persistence si on perd les pods on perd aussi les donnees
- Deploiements : objet de gestion des deploiments
  - Gestion de creation/suppression des pods
  - Gestion des nom de replicas (les replicas sets)
  - Assurer du respect des descriptions des relations entre conteneurs
  - Gestion de scalabilite (gestion des parametres pour la montee en charge)
- Namespaces:
  - cluster virtuel (ensemble de services) à l'interieur de KBS
  - Cloisonner à l'interieur de notre cluster pour des services qui ne travaillent pas ensemble et avoir une coherence des droits des utilisateurs pour acces aux services qui travaillent ensemble et gerer le cloisonnement si on a des services totalement differents
  - Segmenter les pods

MINIKUBE

- Version portable de Kubernetes, local et consomme moins de ressources, constitué d'un cluster sur un seul nœud (seule machine) et fait office aussi de worker
- Le nœud joue à la fois le rôle de master/slave
- Le minikube sert à faire des tests/demos
- Installation de minikube est faite sur virtualbox par image iso ou sur une VM ubuntu

Installation de Minikube

- Installation de minikube:
- On telecharge le fichier minikube binaire en utilisant la commande wget:
  - $ wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
- On copie le fichier binaire et on le stocke dans le dossier :
  - $ sudo cp minikube-linux-amd64 /usr/local/bin/minikube
- On lui procure les permissions d'execution sur le fichier binaire:
  - $ sudo chmod 755 /usr/local/bin/minikube
- On verifie l'installation de Minikube en affichant sa version :
  - $ minikube version

Installation de Minikube

- Pour deployer et gerer les clusters on doit installer kubectl, qui est l'outil de ligne de commande, officiel pour kubernetes:
- D'abord, on telecharge kubectl :
  - $ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl
- Attribuer les droits d'execution au fichier binaire:
  - $ chmod +x ./kubectl
- Deplacer le fichier binaire au path :
  - $ sudo mv ./kubectl /usr/local/bin/kubectl
- Verifier l'installation en verifiant la version de l'instance kubectl:
  - $ kubectl version –o json
- Demarrage de minikube: une fois tous les paquetes instattles on lance le minikube :
  - $ minikube start
- Quand on lance le minikube, le systeme telecharge le fichier ISO Minikube de la source en ligne et le localkube binaire. Ensuite, il cree une machine virtuelle dans VirtualBox, au sein de laquelle il demarre et configure dans un seul nœud cluster.

Installation de Minikube

- Gestion des commandes avec Minikube
  - Afficher la configuration de Kubectl:
    - $ kubectl config view
  - Afficher les informations du cluster
    - $kubectl cluster-info
  - Verifier les nodes qui s'executent:
    - $ kubectl get nodes
  - Afficher la liste des pods Minikube
    - $ kubectl get pod
  - Connexion à la Machine virtuelle Minikube:
    - $ minikube ssh
  - Pour exiter du shell :
    - $ exit
  - Arreter l'execution du seul nœud du cluster
    - $ minikube stop
  - Verifier le status de minikube en marche ou en arret:
    - $ minikube status
  - Afficher la liste des add-ons installes de Minikube;
    - $ mininkube addons list
  - Pour supprimer le cluter avec un suel nœud:
    - $ minikube delete

Pre-requis de Kubernetes

- 2 Gb de memoire RAM
- 2 CPU
- Ouverture reseau large entre les 2 machines
- Port master : 6443 2379 2380 10250 10251 10252
- Port node : 10250 30000 32767
- Pas de swap

Vagrant

- Vagrant est un outil de commande en ligne qui permet la construction et la gestion de l'environnement des machines virtuelles et un moyen effective pour deployer kuberenetes.
- Vagrant a besoin d'un hyperviseur pour fonctionner tels que VirtualBox, Vmware, ou Hyper-V.
- Le fichier vagrant va installer les nœuds master et worker en specifiant les :
  - addresses ip
  - Le nombre de CPU
  - Memoire RAM
  - L'outil docker
  - Les noms des machines master et workers
- une fois lancé le fichier on a les 2 machines qui tournent et on peut se connecter par le protocole SSH
- Toutes les interactions avec Vagrant sont faites par l'interface de commande en ligne:
  - L'interface est valable en utilisant les commandes vagrant, et sont installés avec Vagrant automatiquement. Les commandes de Vagrant disposent de plusieurs sous commandes
- Pour lancer le cluster on utilise la commande: $ vagrant up
- Pour detruire un cluster ont utilise la commande : $ vagrant destroy -f

Le dépôt de Vagrant

- **Vagrantfile**:
  - la premiere fonction de vagrantfile est de decrire le type de machine demandé pour le projet, et la methode pour configurer et provisonner les machines.
  - Dans ce fichier on trouve les parametres tels que le type du Systeme d'exploitation, le nom des machines virtuelles dans le cluster, le nombre de noeuds
  - le fichier specifie aussi les scripts bootstrap
- Bootstrap.sh: Ce fichier est un ensemble basique des instructions initiales qui sont generalement appliqué au noeud master kuberenetes et les noeuds workers aussi.
  - La mise à jour du fichier /etc/hosts pour inclure tous les noeuds , installe docker, et desactive le firewalld, et installe aussi kubernetes.
- **bootstrap_kmaster.sh**:
  - ce fichier initialise kubernetes et cree le reseau flannel
- **bootstrap_kworker:**
  - ce script va joindre les worker noeuds au cluster

Désactivation du swap

- Il existe deux types de mémoire : RAM et disque dur
- RAM: Random Allocation Memory
  - RAM : stockage des données, paramètres , applications,
  - La RAM est une mémoire volatile (les données stockés dans la RAM sont perdus lorsque l'ordinateur s'etteint)
- Les disques durs
  - Les disques durs sont des supports magnetiques utilises pour le stockage à long terme des donnees et des aplications
  - Les disques durs sont non vollatiles et durables
  - Le CPU ne peut acceder directement aux applications et aux donnees sur disque dur, il doit etre copié sur RAM
  - Pendant le demarrage l'ordinateur copie les logiciels du systeme d'exploitation tels que kernel et init ou systemd et les donnees du disque dur vers la RAM accessible par le CPU

Désactivation du SWAP

- le SWAP fonctionne comme une taille supplémentaire de la RAM, ainsi si la RAM est totalement occupé, toutes les applications supplémentaires s'executeront via la partition d'echange au lieu de la RAM.
- L'inconvenient du SWAP est l'aller et retour vers le disque dur pour amener les donnees et les stocker dans la RAM pour une utilisation par le CPU, ce qui ralentit l'execution des applications.
- L'idee de kubernetes est de regrouper étroitement les instances pour qu'elles soient utilisés à 100 % autant que possible.
- Tous les deploiements doivent etre epinglés avec des limites CPU/memoire.
- Si le planificateur envoie un pod à une machine il ne doit jamais utiliser de swap. Tu ne veux pas echanger car ça va ralentir les choses.
- La commande pour desactiver le swap: $ swapoff –a
- Lors du demarrage, le kernel active automatiquement le swap donc on doit desactiver la ligne de swap dans le fichier /etc/fstab.

Pre-requis d'installation

- Installation de l'outil curl de telechargement des packages
  - $ apt-get update && apt-get install –y apt-transport-https curl
- Installation de la clé gpg pour les paquets de source google
  - $ curl –s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
- Ajouter le dépôt de google dans la source liste pour recuperer les binaires qui permettent de travailler sur kubernetes
  - $ sudo add-apt-repository ‘’deb http://apt.kubernetes.io/ kubernetes-xenial main’’

Installation de kubernetes

- Installation de l'outil kubectl:
  - Kubectl: permet d'interagir à l'utilisateur d'interagir avec le cluster (lister les nodes, lister les pods,)
  - L'utilisateur via les commandes de l'outil kubectl va contacter l'API Server qui va se charger de communiquer aux composants serveurs pour executer les requetes de l'utilisateur, deployer des applications , inspecter et gerer les ressources du cluster et consulter les logs.
  - Installation des binaires kubernetes:
    - $ sudo apt-get install –y kubelet kubeadm kubectl kubernetes-cni
    - $ systemctl enable kubelet
    - Kubeadm: installation du cluster
    - Kubelet: service qui tourne sur les machines (lancement des pods,

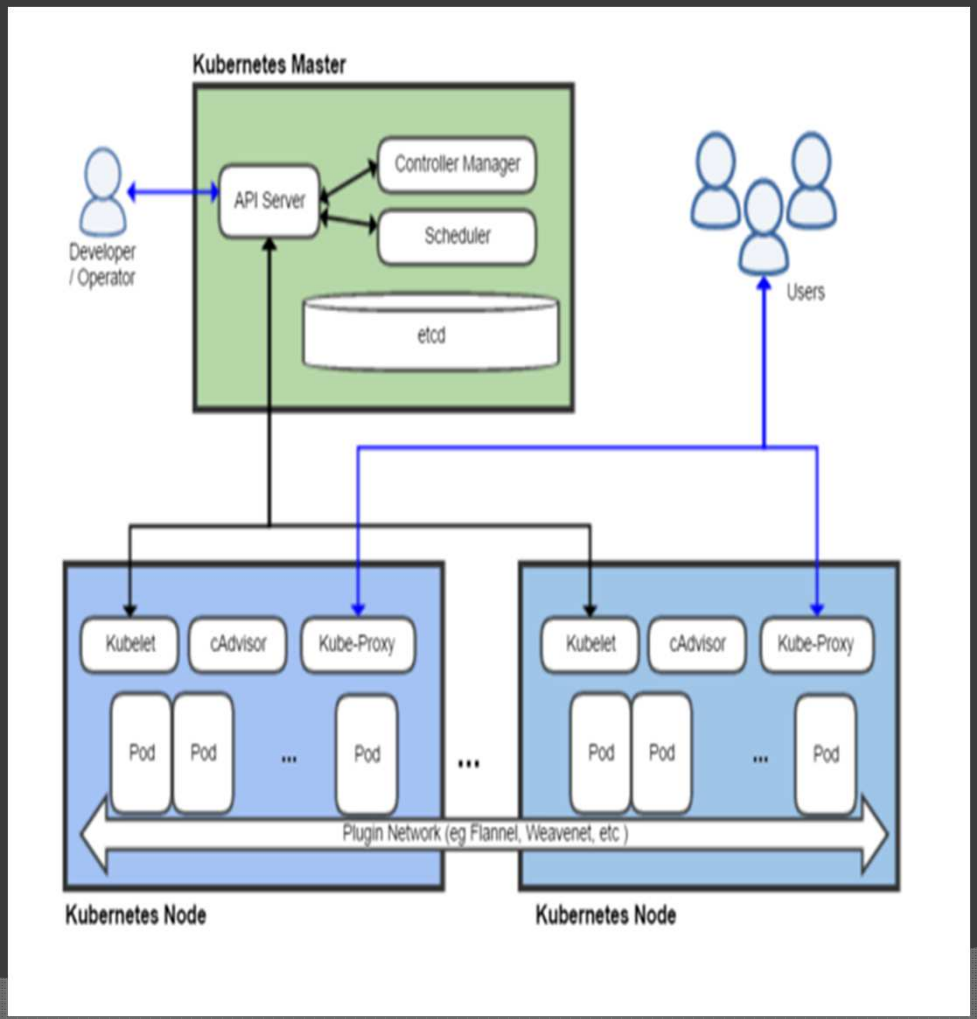
INSTALLATION DE KUBERNETES

- Initialisation sur le master
  - $ kubeadm init - -apiserver-advertise-address=192.168.56.101 - -node-name $HOSTNAME - -pod-network-cidr=10.244.0.0/16
  - 192.168.56.101 l'addresse de la machine
  - 10.244.0.0/16 l'addresse du reseau interne de kubernetes, qui sera utilisé pour attribuer des addresses ip au sein de son reseau
  - $ HOSTNAME est la variable d'environement de la machine
  - Cette commande va generer un token
- Creation du fichier de configuration qui va permettre de travailler avec kubectl
- Kubectl s'appuie sur le fichier de configuration pour savoir le cluster surlequel il travaille et avoir les droits necessaires sur ce cluster
  - $ mkdir –p $HOME/.kube
  - $ sudo cp –i /etc/kubernetes/admin.conf $HOME/.kube/config
  - $ chown $(id –u):$(id –g) $HOME /.kube/config
- Ces changements de droit sont indispensables quand on a un master sur une machine du cluster et les nœuds sur d'autres machines et on veut utiliser kubectl à distance

Mise en place du réseau interne

- L'unité de base de l'ordonnancement dans Kubernetes est appelée « *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 ressources.
- 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* peut définir un volume, comme un répertoire sur un disque local ou sur le réseau, et l'exposer aux conteneurs de ce *pod*.
- Les *pods* peuvent être gérés manuellement au travers de l'API de Kubernetes. Leur gestion peut également être déléguée à un contrôleur.
- Les *pods* sont rattachés au nœud qui les déploie jusqu'à leur expiration ou leur suppression.
- Si le nœud est défaillant, de nouveaux *pods* possédant les mêmes propriétés que les précédents seront déployés sur d'autres nœuds disponibles.
- Ajout d'un pod pour gestion du réseau interne
  - Plusieurs systemes de reseau sont utilises : flannel, weavnet, ,,,,,
  - Le systeme de reseau s'appuie sur la notion de pod, un systeme de conteneur docker qui sont deployés sur chacune des machines du cluster, et qui vont permettre au cluster de coummuniquer entre les machines
  - Kubernetes utilise un bridge pour gerer ces tables
    - **$ sysctl net.bridge.bridge-nf-call-iptables=1**
  - Exemple : Kubernetes lance les pods en utilisant un reseau flannel
  - $ kubectl apply – f https://raw.githubusercontent.com/coreos/flannel/xxxxxxx,,,,xx/Documentation/kube-flannel.yml

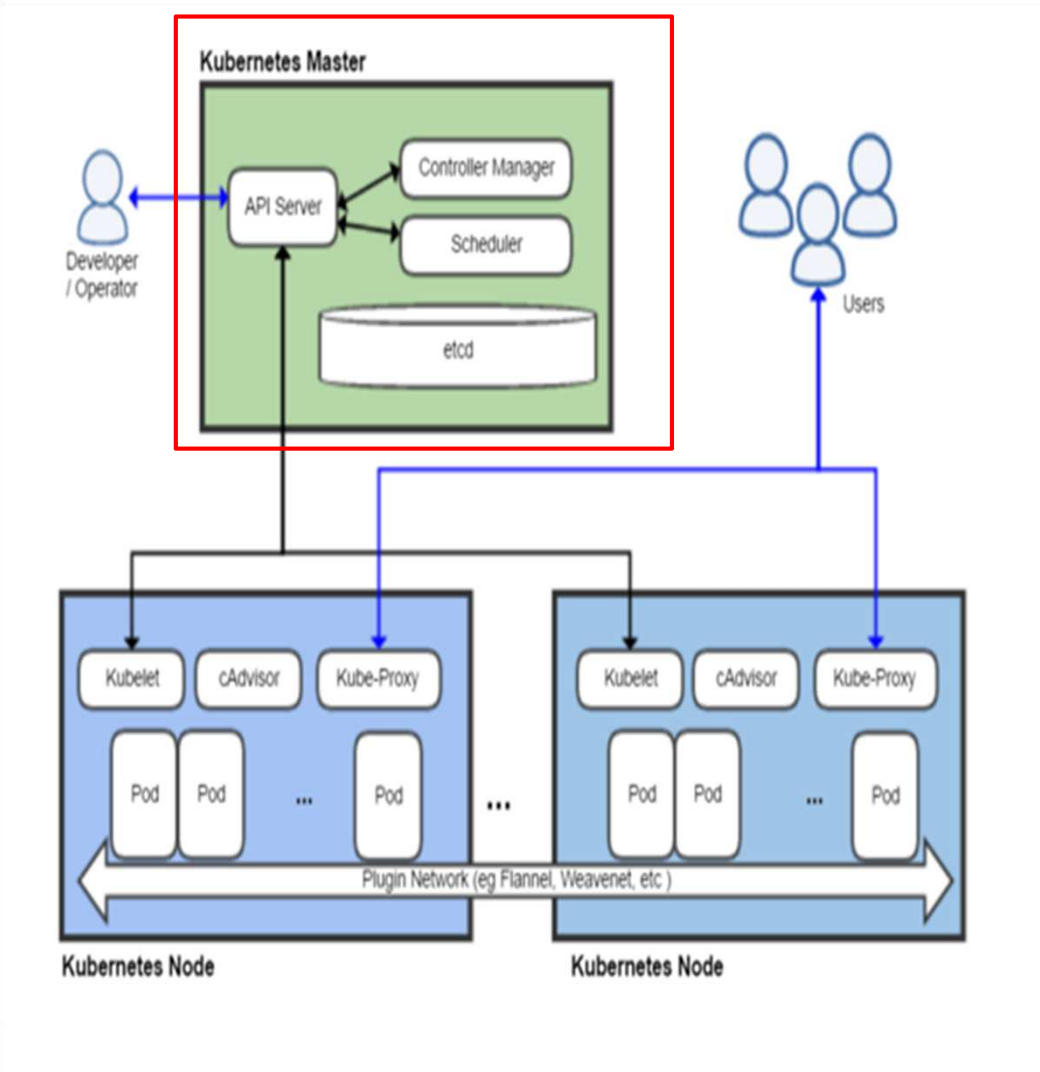
Architecture de kubernetes



Kubernetes obeit à l'architecture maitre/esclave

- Les composants de kubernetes sont divises en composants de kubernetes master et ceux de kubenetes worker.
- Kubernetes master est une unité de contrôle qui charge la repartition de la charge de travail sur les pods, et dirige les communications dans le systeme et controle la santé des noeuds.
- Des qu'il y a une panne dans pod il va arreter et demarrer dans un autre neoud.
- Le kubernetes master comporte plusieurs composants et dans ce cas on a le choix d'installer ses ciomposants sur une seul machine du cluster ou d'un ensemble de machines du cluster pour permettre la haute disponibilté

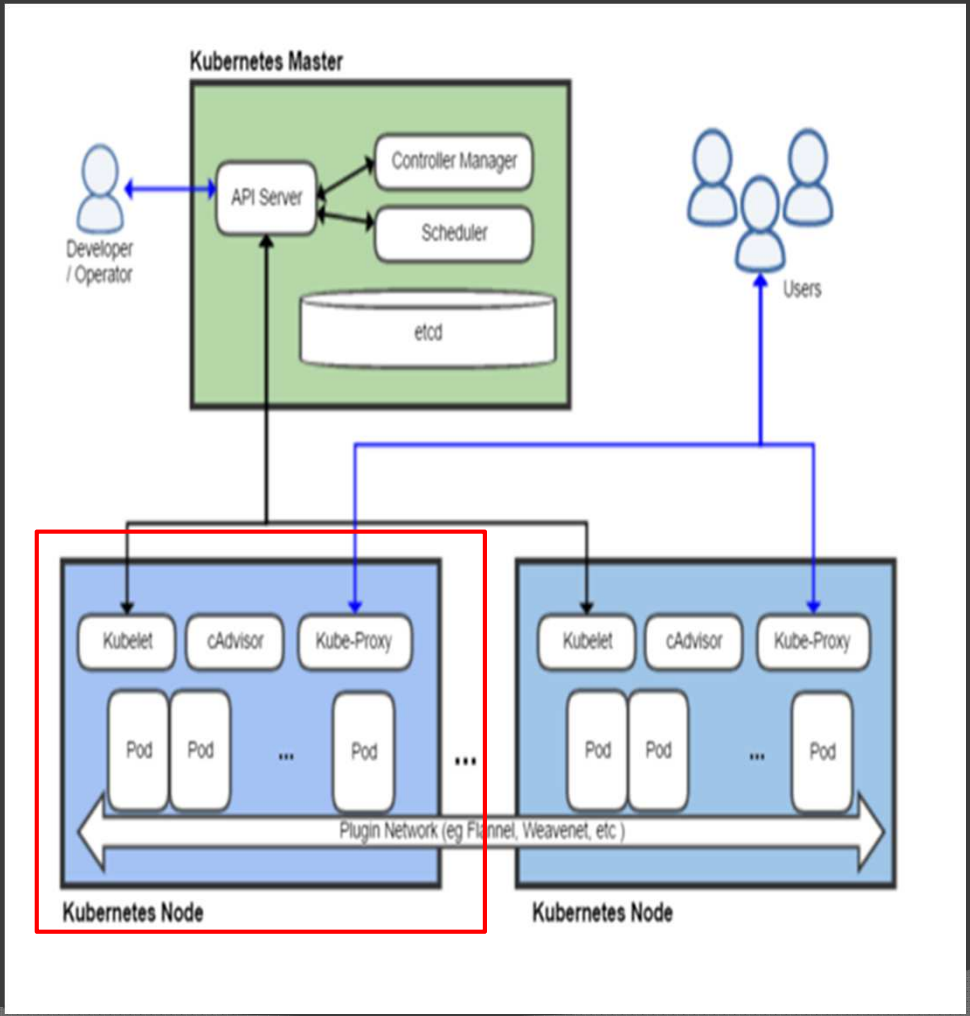
Kubernetes Master



Les compodants deployés du plan de contrôle de kubernetes:

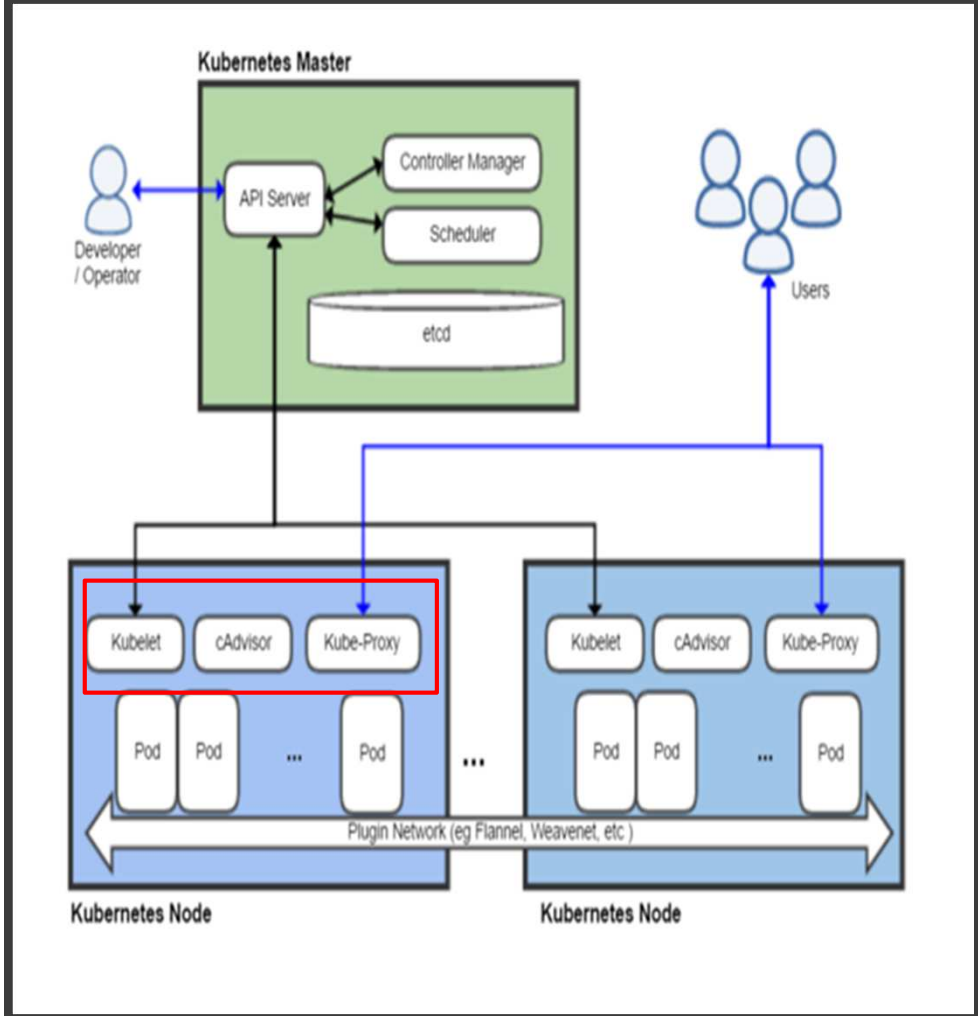
- Etcd: SGBD interne distribué et persistant qui stocke l'etat du cluster (tous les evenement sui se produisent)
- API Server: conteneur web avec une API REST qui gere la communication avec les composants internes et externes
- Scheduler: l'ordonnanceur qui perme de determiner la machine la moins chargé pour deployer le pod, donc il doit envoyer toutes les informations sur les nœuds (processus, memoire,,,)
- Controller Manager: des procesus dans lequel s'executent les princpaux controleurs tels que DaemonSet Controller et le processus Replication Controller

Kubernetes Node



- Le Node appelé aussi Worker est une machine unique (ou une machine virtuelle) ou des conteneurs (les services sous forme de charges de travail) sont deployés sous forme de pod (les pods regroupent des conteneurs)
- Puisque les nodes contiennent des pods, donc chaque node du cluster doit executer le programme de conteneurisation de docker (Docker Engine)

Les Composants d'un Worker Node



Les composants d'un worker node sont responsables du deploiement des pods, et conteneurs,en plus, disposent des composants par default qui sont:

- Kubelet: Responsable de l'etat d'execution du nœud (c'est-à-dire, d'assurer que tous les conteneurs sur un nœud sont en bonne santé organisés en Pods et envoyer des informations en permanence au kubernetes master (plan de contrôle).
- Kube-proxy: Responsable d'effectuer le routage du trafic vers le conteneur approprié (selectionné pour executer cette tache) en se basant sur l'addresse IP et le numero de port de la requete entrante (service demandé par l'utilisateur)
- cAdvisor: Agent qui surveille et recupere les donnees de consommation des ressources locales des performances comme le processeur, la memoire, ainsi que l'utilisation disque et reseau des conteneurs du Node.et envoyer ses donnees au composant scheduler (ordonnanceur) dans le composant master qui va se charger de repartir les charges selon l'etat des ressources des noeuds

Les Pods

- Groupe de containers deployés ensemble
- Les conteneurs du Pod
  - Demarrés, arretes, repliqués en groupe
  - Partagent le meme Network, NameSpace, IP et Ports
  - Communiquer ensemble en utilisant localhost
  - Partager des donnees via des shared volumes

- Le Pod est l'entité de base qu'on va repliquer et qu'on va mettre en reseau sur differentes nœuds du cluster
- Les commandes qui sont utilisés pour les pods:
- $ kubectl create –f mypod.yml //creation de pods par le fichier mypod.yml qui contient les services à demmarer, le nombre de replicas, les volumes partagés.
- $ kubectl get pods //liste des pods
- $ kuebectl delete pod_id // supprimer un pod par son numero d'identification

Espace de stockage

- Un espace de stockage est un volume qui peut être attaché à un conteneur ou à un groupe de conteneurs dans un même pod.
- Un espace de stockage peut être persistant indépendamment du conteneur.
- Par exemple il permet aux conteneurs de partager un même répertoire dans un même pod.

Kubernetes : joindre les nœuds au master

- Apres l'installation du reseau flannel on verifie l'etat des pods
  - $ get pods –all –namespace //cette commmande liste tous les pods qui tournent sur la machine
  - $ kubectl get nodes // lister les nœuds du cluster
- On joint le node au master
  - $ kubeadm join IP_master:port - -token numero_token - -discovery-token-ca-cert-hash numero_hashage
  - Apres cette commande on verifie bien si notre node a rejoint notre master

Autocompletion

- L'**auto-completion** est un outil pour gagner en rapidité lorsque vous tapez des commandes dans un terminal ou une console. Il sera également un outil de sécurité indispensable pour rédiger des lignes de commande sans se tromper
- Disposer de l'autocompletion:
  - Prerequis :installation du paquet bash-completion:
    - $ apt-get install bash-completion
  - Sourcer le fichier bashrc pour rendre l'autocompletion disponible en copiant le module d' autocompletion bash de kubectl dans le fichier bashrc
    - Echo ‘’source <kubectl completion bash)’’ >> ~/.bashrc
    - Exemple : on peut tester l'autocompletion en tapant :
    - $ kubectl get
    - Ou meme $ kube

Fichier bashrc

- Le fichier .bashrc (/etc/bash.bashrc) est un script exécuté chaque fois qu'une nouvelle session de terminal est démarrée en **mode interactif** .
- C'est ce qui se passe lorsque vous ouvrez une nouvelle fenêtre de terminal en appuyant sur Ctrl + Alt + T, ou simplement pour ouvrir un nouvel onglet de terminal.
- En revanche, une session de terminal en **mode de connexion** vous demandera un nom d'utilisateur et un mot de passe et exécutera le script ~/.bash_profile .
- C'est ce qui se produit, par exemple, lorsque vous vous connectez à un système distant via SSH.
- Le fichier .bashrc lui-même contient une série de configurations pour la session de terminal. Cela inclut la configuration ou l'activation: coloriage, complétion, historique du shell, alias de commandes, etc.
- Le fichier .bashrc distribué avec Ubuntu est bien commenté et vous pourrez comprendre la plupart de ses actions en le lisant.

ALIAS

- Dans le fichier /etc/.bashrc on peut ajouter des alias pour faciliter les commandes tapés
- alias k=‘kubectl’
- alias kcc=‘kubectl config current-context’
- alias kg=‘kubectl get’
- alias kgp=‘kubectl get pods’
- Etc,,,,

ACCES DISTANT

- Pour accéder d'une machine à distante on doit :
  - Installer kubectl sur la machine distante :
  - $ curl –s https://packages.cloud.google.com/apt/doc/apt-key.gpg
  - $ Sudo add-apt-repository

    ‘’deb http://apt.kubernetes.io/ kubernetes-xenial main’’

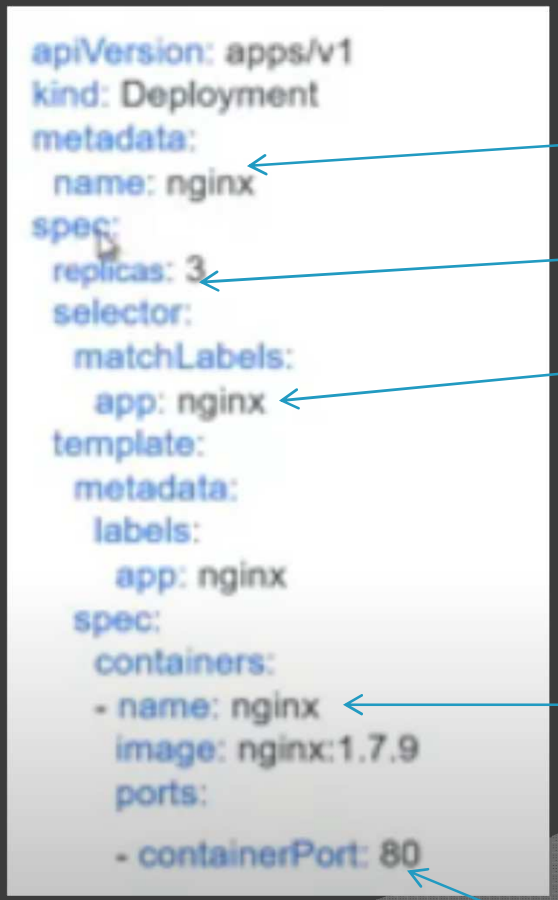
  Créer le repertoire : $ mkdir ~/.kube
  - Recuperer le token pour se connecter au master et pour cela on se connecte au master par ssh et on copie le token
  - $ ssh user@IP_master ‘’sudo cat /etc/kubernetes/admin.conf’’ > .kuber/conf

POD & DEPLOIEMENT

- Le deploiement est une representation logique (configuration) de un ou plusieurs pods
- L'option run de kubectl lance un deploiment de pod en ligne de commande :
  - $ kubectl run myshell –it –image busybox --sh
  - Cette commande va créer un deploiement de pod qui porte le nom myshell et va créer un terminal it, et ce pod se base sur une image qui va telecharger le hub docker puisque kubernetes se base sur docker, et enfin sh est la commande qu'on veut passer au pod pour l'executer.
  - Le nœud master ne comporte que les pods systemes, donc cette commande va lancer le pod sur le node, on verifie ceci sur le nœud worker : $ docker ps
  - Pour supprimer tous les pods deployés de myshel on doit supprimer le deploiement myshell :
  - $ delete pods myshell
  - Kubectl run anothershell –it --image busybox –sh
- L'option exec de kubectl permet l'execution d'une commande dans un pod ou se connecter en bash

POD & deploiement

- Pour deployer un pod on va créer un fichier pod en format JSON ou en YAML qui decrit l'architecture des pods.
- Declarer les conteneurs qu'on veut mettre dans les pods, et les replicas de ces conteneur et les mises à jour
- Apres avoir deploye, le contrôleur de kubernetes va se charger de maintenir l'etat des pods, ainsi en cas de defaillance d'un pods le controleur va le desactiver et l'enlever du load balancer et créer un autre pour continuer le fonctionnement dans les memes conditions exigés par le descripteur.



Exemple de fichier de déploiement

SERVICES

- Le service est un moyen d'acceder aux pods par un couple IP/PORT
- Tout d'abord on cree un pod avec un deploiement de nom ngninx:
  - $ kubectl create deployment monnginx - -image nginx4
- Pour inspecter le deploiement et afficher les metadonnes on a la commande
  - $ kubectl describe deploy monnginx
- Pour inspecter les pods cree par ce deploiement :
  - $ kubectl describe pod ID_PODS
  - ID du Pod est extrait par la commande: $ kubectl get pods
- Apres cette operation, le conteneur est crée mais on a pas de port pour y acceder

Exposition des ports :services

- Le service est un moyen d'acceder aux pods
  - Il ya plusieurs methodes pour créer le service:
    - Nodeport: exposition du port pour rendre publique le pod à l'exterieur du cluster, soit kubernetes va attribuer un port compris entre (30000-32767), soit on va forcer le choix du port
    - Clusterip: on expose le pod mais à l'interieur du cluster et pas publiquement
    - loadBalancerIP: dans le cas du cloud on attribue directement des IP aux pods ce qui permet d'y acceder
    - Exeternalname= se baser sur une URL pour acceder aux pods
- Exemple: Créer un sevice pour le pod du serveur web nginx:
  - $ kubectl create service nodeport monnginx - - tcp =8080:80
    - Le port 80 est pour l'acces au serveur nginx
    - Le port 8080 est pour un acces au pod par un autre pod à l'interieur du cluster
    - Si on veut acceder de l'exterieur on doit acceder par un numero de port avec IP_machine:numero_port
- Pour afficher les services sur le systeme:
  - $ kubectl get svc

SCALING

- Le scaling consiste à créer plusieurs instances du même service, dans l'objectif de supporter la charge.
- On va tout d'abord deployer un pod unique ensuite on créer un 2eme pod pour scaler le premier:
  - $ kubectl create deploy monnginx - -image nginx
  - Verifier son installation par :
  - $ kubectl get pods
  - Créer un service pour le cluster
  - $ kubectl create service nodeport monnginx - -tcp 8080:80
  - On verifie le service créé par $ kubectl get svc et on retrouvera le port d'exposition 8080:N°_PORT_EXPOSITION
  - On peut y acceder par l'adresse:
  - https://IP_MACHINE:N°_PORT_EXPOSITION
  - On ouvre un terminal sur le pod pour l'editer:
  - $ kubectl exec - ti nom_pod /bin/bash avec le nom du pod extrait par la commande $ kubectl get pods et on indique la commande qu'on veut executer dans ce cas /bin/bash
  - Une fois on a le bash on peut changer le fichier index du site lié au premier pod
  - $ echo ‘’instance 1’’ > /usr/share/nginx/html/index.html

SCALING

- Pour scaler horizontalement on augmente le nombre de replicas:
- $ kubectl scale deployment monnginx - - replicas=2
- On veifie l'existence du 2eme pod par la commande:
- $ kubectl get pods
- On remarque qu'on a créer un 2eme pod qui un nom different et qu'on peut se connecter a ce pod par la commande :
- $ kubectl exec - ti nom_pod_2   /bin/bash
- de la meme façon on change le fichier index du site mnnginx :
- $ echo ‘’instance 1’’ > /usr/share/nginx/html/index.html
- A ce stade on
- While true; do P_MACHINE:N°_PORT_EXPOSITION; done
- Donc on peut appeler le meme addresse IP et le mme port et on aura les 2 pods qui vont s 'executer par alternative grace au princpe du Load balancing
- Dans notre cas le load balancing n'est pas sequentielle (ROUD ROBIN) mais plutot sur le fait sit la ressource est utilise ou pas.
- Ainsi on peut ajuter plusieurs replicas du meme service avec le meme addresse ip et meme port d'exposition
- Dans le cas ou ont arreter les replicas et se limiter à 1 seul rpeplicas, on parametre l'option replicas par la valeur 1:
- $ $ kubectl scale deployment monnginx - - replicas=1

- Le scaling automatique (autoscale) consiste à definir un minimum de relicas et un maximum de replicas en fonction de la charge
  - $ kubectl autoscale deployment monnginx --min=1 ---max=5

Les informations

- Kubernetes dispose de plusierus commandes pour receuillir des informations sur les nœuds du cluster :
- $ kubectl get nodes
- $ kubectl get nodes - o wide //cette commande nous donnes des informations detaillés sur les noeurs :addresse IP, OS utilisé, les versions de kernel et de docker
- Pour avoir des informations sur un neoud precis on a les commandes :
- $ kubectl describe nodes nom_noeud //cette commande permet d'avoir des informations detaillés sur le nœud (labels, role, date de creation, addresse IP, nombre de CPU affécté, memoire, os image, version kubelet name, les ressources alloues)
- Lister les compsants du neoud :
- $ kubectl get componentstatuses
- Liste les pods qui tournent en arriere plan sur differnts neouds:
- $ kubectl get daemonsets –n kube -system

Exercice

- 2 machines ubuntu sur VB ou vmware
- Machine master IP: 162.168.1.100
- Machine node IP: 192.168.1.101
- Créer un reseau local entre les 2 machines :(ifconfig, fichier /etc/hosts)
- Ping entre les 2 machines avec IP et HOSTNAME
- Installer kubectl kubelet sur master et node (voir le cours)
- Installer kubeadm sur master (voir le cours)
- Initialiser le cluster kubernetes et le dossier de configuration
- Joindre la machine node
- Executer les commandes vu dans le cours