KUBERNETES

www.inow.fr

contact@inow.fr

KUBERNETES (K8S)

- COE développé par Google, devenu open source en 2014
- Adapté à tout type d'environnements
- Devenu populaire en très peu de temps
- Premier projet de la C.N.C.F

OCI

- Créé sous la LinuxFondation
- But : Créer un standard Open Source concernant la manière de "runner" et le format des conteneurs et images
- Non lié à des produits
- Non lié à des COE
- runC a été donné par Docker à l'OCI comme implémentions de base

K8S : PROJET

- Docs
- Slack
- Stack Overflow
- Hébergé sur Github:
  - Issues
  - Pull Request
  - Release
- Projets en incubation

K8S : COMMUNAUTÉ

- Contributor and community guide
- Décomposée en SIG(Special Interest Group)
- Les SIG sont des projets, centres d'intérêts ou Working Group différents :
  - Network
  - Docs
  - AWS
  - etc
- Chaque SIG peut avoir des guideline différentes.

KUBERNETES : CONCEPTS

- Networking
- Volumes
- Cloud Providers :Load Balancer

KUBERNETES : API RESSOURCES

- PODs
- Deployments
- Ingress et Ingress controller
- NetworkPolicy
- Namespaces
- Services

KUBERNETES : POD

- Ensemble logique composé de un ou plusieurs conteneurs
- Les conteneurs d'un pod fonctionnent ensemble (instanciation et destruction) et sont orchestrés sur un même hôte
- Les conteneurs partagent certaines spécifications du POD :
  - La stack IP (network namespace)
  - Inter-process communication (PIDnamespace)
  - Volumes
- C'est la plus petite unité orchestrable dans Kubernetes

KUBERNETES : POD

- Les PODs sont définis en YAML comme les fichiers docker-compose :

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

KUBERNETES : NETWORKING

- Les conteneurs peuvent communiquer sans NAT
- Un nœud peut accéder aux conteneurs des autres nœuds sans NAT
- Nécessite une solution tierce :
  - Canal : Flannel + Calico
  - Weaves
  - OpenShift
  - OpenContrail
  - Romana
- Ces solutions implémentent CNI (Container Network Interface)

KUBERNETES : CNI

- Projet de la CNCF
- Spécifications sur la configuration d'interface réseaux des conteneurs
- Ensemble de plugins core ainsi que tierce partie
- Docker n'utilise pas CNI mais CNM (Containter Network Model) et son implémentation *libnetwork*.

KUBERNETES : VOLUMES

- Fournir du stockage persistent aux PODs
- Fonctionnent de la même façon que les volumes Docker pour les volumes hôte :
  - EmptyDir ~= volumes docker
  - HostPath ~= volumes hôte
- Support de multiples backend de stockage :
  - GCE : PD
  - AWS : EBS
  - glusterFS / NFS
  - Ceph
  - iSCSI

KUBERNETES : VOLUMES

- On déclare d'abord le volume et on l'affecte à un service :

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: redis-persistent-storage
      mountPath: /data/redis
  volumes:
  - name: redis-persistent-storage
    emptyDir: {}
```

KUBERNETES : NAMESPACES

- Fournissent une séparation logique des ressources par exemple :
  - Par utilisateurs
  - Par projet / applications
  - Autres...
- Les objets existent uniquement au sein d'un namespace donné
- Évitent la collision de nom d'objets

KUBERNETES : LABELS

- Système de clé/valeur
- Organiser les différents objets de Kubernetes (PODs, RC, Services, etc.) d'une manière cohérente qui reflète la structure de l'application
- Corréler des éléments de Kubernetes : par exemple un service vers des PODs

KUBERNETES : LABELS

- Exemple de label :

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  -   name:   nginx
    image:    nginx
    ports:
    - containerPort: 80
```

KUBERNETES : SERVICES

- Abstraction des PODs et RCs, sous forme d'une VIP de service
- Rendre un ensemble de PODs accessibles depuis l'extérieur
- Load Balancing entre les PODs d'un même service

KUBERNETES : SERVICES

- Load Balancing : intégration avec des cloud provider :
  - AWS ELB
  - GCE
- Node Port forwarding : limitations
- ClusterIP : IP dans le réseau privé Kubernetes (VIP)
- IP Externes : le routage de l'IP publique vers le cluster est manuel

KUBERNETES : SERVICES

- Exemple de service (on remarque la sélection sur le label):

```
{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "example-service"
  },
  "spec": {
    "ports": [{
      "port": 8765,
      "targetPort": 9376
    }],
    "selector": {
      "app": "example"
    },
    "type": "LoadBalancer"
  }
}
```

KUBERNETES : DEPLOYMENTS

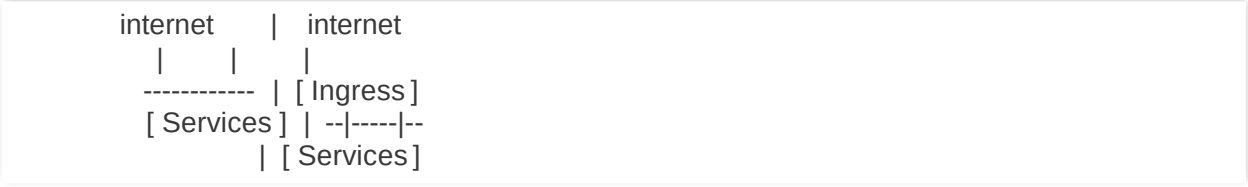
- Permet d'assurer le fonctionnement d'un ensemble de PODs
- Version, Update et Rollback
- Souvent combiné avec un objet de type *service*

KUBERNETES : DEPLOYMENTS

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx-svc
  labels:
    app: nginx
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: nginx
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: my-nginx
spec:
```

KUBERNETES : INGRESS RESOURCE

- Définition de règles de routage applicatives (HTTP/HTTPS)
- Traffic load balancing, SSL termination, name based virtual hosting
- Définies dans l'API et ensuite implémentées par un Ingress Controller



KUBERNETES : INGRESS RESOURCE

- Exemple :

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  namespace: default
  name: traefik
  annotations:
    kubernetes.io/ingress.class: "traefik"
spec:
  rules:
    - host: traefik.archifleks.net
      http:
        paths:
          - backend:
              serviceName: traefik-console
              servicePort: 8080
```

KUBERNETES : INGRESS CONTROLLER

- Routeur applicatif de bordure (L7 Load Balancer)
- Implémente les Ingress Resources
- Plusieurs implémentations :
  - Træfık
  - nginx

KUBERNETES : NETWORKPOLICY

- Contrôle la communication entre les PODs au sein d'un namespace
- Pare-feu entre les éléments composant une application :

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
 name: test-network-policy
 namespace: default
spec:
 podSelector:
  matchLabels:
    role: db
 ingress:
  - from:
     - namespaceSelector:
        matchLabels:  project:
        myproject
     - podSelector:
        matchLabels:
         role: frontend
    ports:
```

KUBERNETES : RUN ET ADMINISTRATION

- WebUI (Kubernetes Dashboard)
- Kubectl (Outil CLI)
- Objets: *Secret* et *ConfigMap* : paramétrages, plus sécurisés que les variables d'environnements

KUBERNETES : COMPOSANTS

- Kubernetes est écrit en Go, compilé statiquement.
- Un ensemble de binaires sans dépendances
- Faciles à conteneuriser et à packager
- Peut se déployer uniquement avec des conteneurs sans dépendances d'OS

KUBERNETES : COMPOSANTS

- kube-apiserver : API server qui permet la configuration d'objet Kubernetes (Pods, Service, Replication Controller, etc.)
- kube-proxy : Permet le forwarding TCP/UDP et le load balancing entre les services et les backend (Pods)
- kube-scheduler : Implémente les fonctionnalités de scheduling
- kube-controller-manager : Responsable de l'état du cluster, boucle infinie qui régule l'état du cluster afin d'atteindre un état désiré

KUBERNETES : COMPOSANTS

- kubelet : Service "agent" fonctionnant sur tous les nœuds et assure le fonctionnement des autres services
- kubectl : Client qui permet de piloter un cluster Kubernetes

KUBERNETES : KUBELET

- Service principal de Kubernetes
- Permet à Kubernetes de s'auto configurer :
  - Surveille un dossier contenant les *manifests* (fichiers YAML des différents composant de K8s).
  - Applique les modifications si besoin (upgrade, rollback).
- Surveille l'état des services du cluster via l'API server (*kube-apiserver*).
- Dossier de manifest sur un noeud master :

```
ls /etc/kubernetes/manifests/
kube-apiserver.yaml kube-controller-manager.yaml kube-proxy.yaml  kube-schedule
```

KUBERNETES : KUBELET

- Exemple du manifest *kube-proxy* :

```
apiVersion: v1
kind: Pod
metadata:
  name: kube-proxy
  namespace: kube-system
  annotations:
    rkt.alpha.kubernetes.io/stage1-name-override:  coreos.com/rkt/stage1-fly
spec:
  hostNetwork: true
  containers:
  - name: kube-proxy
    image: quay.io/coreos/hyperkube:v1.3.6_coreos.0
    command:
    - /hyperkube
    - proxy
    - --master=http://127.0.0.1:8080
    ---proxy-mode=iptables
    securityContext:
```

KUBERNETES : KUBE-APISERVER

- Les configurations d'objets (Pods, Service, RC, etc.) se font via l'API server
- Un point d'accès à l'état du cluster aux autres composants via une API REST
- Tous les composants sont reliés à l'API server

KUBERNETES : KUBE-SCHEDULER

- Planifie les ressources sur le cluster
- En fonction de règles implicites (CPU, RAM, stockage disponible, etc.)
- En fonction de règles explicites (règles d'affinité et anti-affinité, labels, etc.)

KUBERNETES : KUBE-PROXY

- Responsable de la publication de services
- Utilise *iptables*
- Route les paquets à destination des PODs et réalise le load balancing TCP/UDP

KUBERNETES : KUBE-CONTROLLER-MANAGER

- Boucle infinie qui contrôle l'état d'un cluster
- Effectue des opérations pour atteindre un état donné
- De base dans Kubernetes : replication controller, endpoints controller, namespace controller et serviceaccounts controller

KUBERNETES : NETWORK-POLICY-CONTROLLER

- Implémente l'objet NetworkPolicy
- Contrôle la communication entre les PODs
- Externe à Kubernetes et implémenté par la solution de Networking choisie :
  - OpenShift
  - OpenContrail
  - Romana
  - Calico