

Histoire des containers

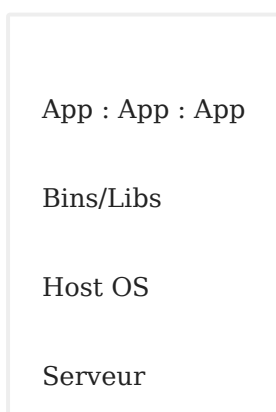
Évolution des applications

TYPE	PASSE	PRESENT/FUTUR
Type d'application	Monolithique	Micro-Services
Cycles de développemenst	Longs	Courts
Environnement	Un seul	Plusieurs
Mise en production	Rares	Fréquentes
Scaling	Vertical	Horizontal
Outils	1 language / 1 stack	Plusieurs langages / stacks

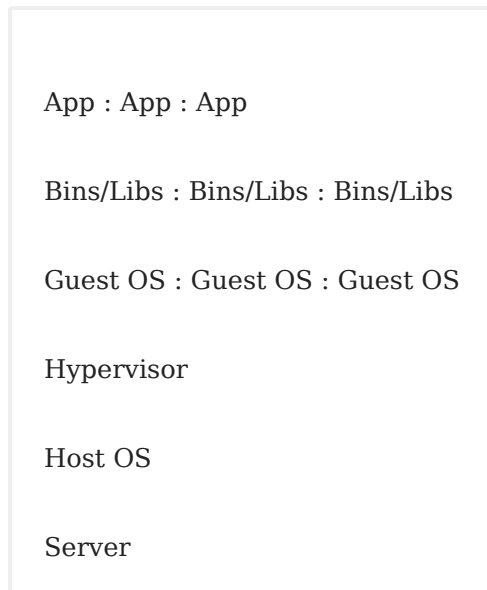
Évolution de l'infrastructure

Passage à la virtualisation

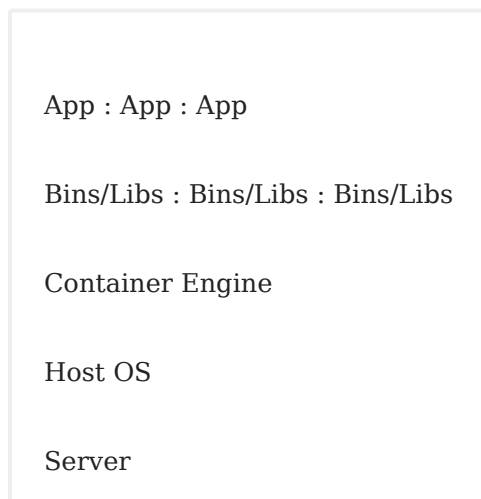
Stack historique



Stack virtuelle



Containers



Container

- Enveloppe permettant de packager une application avec ce dont elle a besoin pour fonctionner
- Peut contenir n'importe quoi (linux kernel)
- Peut être déployé tel quel partout
- Utilise le kernel du serveur hôte
- Isolé : son propre espace de processus + mémoire + stack réseau
- Exécuté directement sur l'hôte
- Permet de décomposer l'infrastructure en petits éléments

Pourquoi des containers

- Meilleures performances que des VMs
- Portabilité d'un environnement à l'autre (multi-cloud)
- Cohérence entre chaque environnement : dev → test → prod
- Modularité simplifiée : une application composée de plusieurs containers
- Gestion facilitée de l'héritage technique grâce à l'isolation

Historique des containers

- 1979 : Unix V7 Chroot
- 1982 : BSD Chroot
- 1996 : GNU + Linux Chroot, in coreutils
- 2000 : BSD Jails
- 2001 : Linux-Vserver
- 2004 : Solaris Container et Zones, by Sun
- 2005 : OpenVZ
 - 2008 : merged in Linux as Linux CGroups + namespaces
- 2008 : LXC, based on Cgroups
- 2011 : Warden, by CloudFoundry
- 2013 : Docker, by dotCloud
- 2014 : LMCTFY, by Google
- 2014 : Rocket, by CoreOS

Conteneurs

- Virtualisation légère
- Un changement avec LXC
 - un système complet avant
 - un seul processus après
- Isolation du reste du système

Reference

- [Aymen El Amri: The Missing Introduction To Containerization](#)
- [A Brief History of Containers: From the 1970s to 2017](#)

Docker

Vue d'ensemble

- L'outil de containérisation le plus connu
 - l'alternative à Docker - rkt - fonctionne aussi avec Kubernetes
- Docker Engine :
 - le runtime Docker
 - fait fonctionner les images docker
- Docker Hub
 - un registre en ligne avec une bibliothèque d'images docker
 - fait pour stocker et récupérer des images docker
 - on peut aussi y construire des images docker en ligne

Intérêt de Docker

- Isolation : un binaire et toutes les dépendances intégrées
- Parité forte : entre les environnements devs, QA, production
- Facilité à mettre en production
- Une image pour déployer partout (laptop, datacenter, VM, etc.)
- Basé sur les Linux Containers (pour l'isolation bas niveau)

Fonctionnement de docker

- `Dockerfile` : liste d'instructions pour une construction reproductible
- Union File System : un filesystem fait de "couches" empilables :
- Une image immuable

Ce que docker permet

- D'abord utilisé sur la machine locale, en développement
- Puis en test / intégration continue
- En prod... heu...

- Comment faire avec tous ces containers à gérer ?

Limites de docker

Des questions ouvertes

- Comment gérer et planifier le cycle de vie des containers ?
- Comment monter en charge ?
 - ex: redimensionner (rescale) l'infrastructure automatiquement ? (ex: php-fpm)
- Comment gérer tous ces containers
 - ex: faire communiquer tout ce beau monde si j'ai un conteneur par processus ?
 - ex: configurer les reverse proxy sait où envoyer les requêtes
- Comment faire la maintenance ?
 - gérer les pannes ?
 - comment mettre à jour mon application ?
- Comment gérer les task queue / les workers ?

Notion d'orchestration

- Arrangement automatisé
- Coordination automatisée
- Gestion automatisée
- De SI et middleware et service

L'écosystème docker

En construction et très changeant (pire que l'écosystème JS ?)

- Docker Compose
- Docker Swarm
- Universal Control Plane
- Trusted Registry
- Apache Mesos
- Kubernetes

- Rancher
- etc.

Quel orchestrateur choisir ?

- Choix difficile il y a encore 2 ans
 - de très nombreuses solutions
 - certains FLOSS, d'autres non
- Qui va survivre jusqu'à l'année prochaine ?
- Docker vient d'annoncer que Kubernetes était désormais géré nativement en plus de son orchestrateur maison (Swarm)
 - pas une bonne nouvelle pour Swarm ...
 - une bonne nouvelle pour la standardisation de Kubernetes ?

➡ Dans ce cours, nous allons nous concentrer sur Kubernetes

Références

- [Docker Docs: Deploy on Kubernetes](#)
- [Wikipedia: Orchestration](#)

Dockerfile

```
# start from alpine base image
FROM alpine

# run apk to install bash
RUN apk add --no-cache bash

# copy mykubeapp application file
ADD mykubeapp /mykubeapp

# execute mykubeapp command
CMD ["/mykubeapp"]

# expose port 8088
EXPOSE 8088
```

Construire une image

Construire une nouvelle image

```
$ docker build -t demo/testimage:1 ./
```

Lister les images locales

```
$ docker images
```

Lancer une image

Lancer le container

```
$ docker run hello-world
```

Lancer le container, connecter à un pseudo-terminal et forwarder stdin dedans

```
$ docker run -it ubuntu:14.04 /bin/bash
```

Lister les containers actifs

```
$ docker ps
```

Utilisation d'un registre

Télécharger l'image depuis un registre

```
$ docker pull some.registry.url/some-repository/some-image:some-tag
```

Pousser l'image sur un registre

```
$ docker push some.registry.url/some-repository/some-image:some-tag
```

Installer un registre privé

```
$ docker run -d \  
  -p 5000:5000 \  
  --restart=always \  
  --name registry \  
  registry:2
```

Références

- [Mise en place d'un docker registry privé](#)
- [Docker Documentation: Deploy a registry server](#)

À propos de Kubernetes

Étymologie

Du grec ancien .

κυβερνήτης (kubernêtês) \ci.ver.'ni.tis\ masculin :

1. Pilote, timonier
2. Guide, gouverneur

A par la suite donné le dérivé latin gubernator.

Objectif

Fournir un système d'orchestration opensource pour les containers Docker

Il est fait pour :

- gérer de multiples containers
- gérer des services au long court
- ordonnancer des containers
- organiser l'accès aux ressources depuis ces containers

Origine

- Inspiré du projet Borg
 - ... dont Google se sert depuis 15 ans
 - Réécriture d'une partie de Borg en Go (FIXME)
- Passage en opensource en 2014
 - Donné à la Cloud Native Computing Foundation (CNCF)
 - Longtemps "expérimental" (version 1.0 en 2015)
- De nombreux contributeurs
 - Google, CoreOS, Redhat, Microsoft, IBM, Mesosphere, VMware, HP, Amazon, Huawei, Oracle, Citrix, eBay, Reddit, MasterCard ...
- Promu en 2018 par la CNCF (FIXME)
 - LE orchestrateur (depuis 2017-2018)

Que permet Kubernetes ?

- Orchestrer (liens entre les containers)
- Créer de l'abstraction (notion de service, pas d'IP)
- Apporter de la haute-disponibilité
- Redimensionner : multiples instances (automatiquement ou "à la main")
- De nombreux fournisseurs: vsphere (vmware), google cloud, aws, azure, bare metal (physique)
- Déployer une application sous forme de containers
 - rapidement
 - de manière prévisible
 - sans interruption de service
- Automatiser le déploiement, l'organisation et les garanties sur les containers
 - Déclarer l'architecture cible (nombre de répliques, contenu, stratégie de mise à jour...)
 - Laisser le système travailler à maintenir cette cible (comme avec puppet, ansible, salt, ...)
 - Organiser les containers en groupes et faire du load balancing
- Distinguer l'application de l'architecture sous-jacente
- Détecter certains problèmes et les résoudre tout seul (ex: pannes de nœuds => rescheduling sur un autre nœud)

Intérêt de Kubernetes

- Opensource
- Hautement modulaire
- Il fonctionne partout
 - embarqué
- Communauté
 - Nombreuses et active
 - Soutenu par Google et d'autres entreprises

Contraintes de K8S

- Culture Agile / DevOps dans l'entreprise
- Compétences linux + réseau + cgroups solides pour déboguer
- FIXME: quels sont les contraintes de K8S
- K8S est assez jeune et il y a encore des changements dans son fonctionnement (ex: run => create, version des modules, etc.)

Fonctionnement

- En mode cluster : controller + worker nodes
 - cloud privé on-premise / bare-metal
 - cloud public : Google Cloud, Amazon Web Services, Microsoft Azure, VMWare vSphere
 - cloud hybride : cloud public + cloud privé simultanément
- En mode démo/développement/test: noeud simple
 - En local avec minikube
 - Utilise une VM (via virtualbox, hyper-v ou libvirt)

Références

- [Xavki: Kubernetes - 1. Introduction](#)
- [Kubernetes Documentation: The Kubernetes API](#)

Concepts et primitives

Cluster

- Ensemble des machines physiques et virtuelles (appelés Nodes)
- Utilisées par Kubernetes
- Pour y faire fonctionner les applications

Nodes

- Machine physique ou virtuelle,
 - Node = nom "moderne" pour serveur
- Support physique du Cluster
- Soit noeud de gestion (controler), soit noeud d'exécution (worker)

Pod

- Ensemble cohérent de containers
- Groupe d'un ou plusieurs containers déployés ensemble

Problèmes :

- par défaut le réseau est interne au cluster, personne ne peut leur parler
- les pods et les containers sont dynamiques et leurs IP changent

Note : "cosse de petit pois" en français

Service

- Un service expose un (ou plusieurs) pod sur le réseau
 - publique (vis à vis d'internet)
 - ou en interne sur notre cluster, pour la communication entre les Pods
- Il fournit aux pods "exposés" une même adresse statique
 - Service = IP + Port fixe
- Différents types
 - ClusterIP

- NodePort
- LoadBalancer
- Permet de faire nativement de la répartition de charge (load balancing)

Volume

- Lieux d'échanges entre les pods
- Intérieur de pods = non-persistant
- Extérieur de pods = persistant

Déploiement

- Décrit les contraintes de fonctionnement des pods
- Objet de gestion des déploiements
- Création / suppression de pods
- Scaling de pods
- Gestion des parametres pour la montée en charge (ou la réduction)

Namespace

- Cluster virtuel
- Permet de segmenter un sous-ensemble de services (pods, volumes, etc.)
- Permet de restreindre les droits, augmenter la cohérence, etc.

Plein d'autres primitives

- Comme avec Git : de très nombreuses options
- Pas besoin de connaitre toutes : 3-4 suffisent pour l'utilisation quotidienne.

References

- <https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/>
- <https://kubernetes.io/docs/concepts/workloads/pods/pod/#termination-of-pods>
- <https://www.youtube.com/watch?v=Ew7QigU8JMQ>

Infrastructure as code

Syntaxe

Du YAML

Note

La taille moyenne d'un fichier de config 50-150 lignes

La base

```
metadata:
  labels:
    app: monapp-api
spec:
  containers:
    - name: "php"
      image: "mon-registre.com/monapp-php:v42"
      ports:
        - containerPort: 8080
```

Pour un déploiement de pod

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: monapp-api
spec:
  template:
    metadata:
      labels:
        app: "monapp"
        type: "api"
    spec:
      containers:
        - name: "php"
          image: "mon-registre.com/monapp-php:v42"
          env:
            - name: APP_ENV
              value: "prod"
# ...
```


Pour un service

```
apiVersion: apps/v1
kind: Service
metadata:
  name: "monapp-api"
spec:
  type: LoadBalancer
  ports:
    - name: nginx-http
      protocol: TCP
      port: 80
      targetPort: 8080
  selector:
    app: "monapp"
    type: "api"
# ...
```

Pour un scaling automatique

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: "monapp-api"
  minReplicas: 2
  maxReplicas: 8
  metrics:
    - type: Resource
      resource:
        name: cpu
        targetAverageUtilization: 80
# ...
```

CLI

Kubectl

- Application en ligne de commande (wrapper)
- Permet d'interagir avec l'API de Kubernetes

L'outil kubectl

Outil en ligne de commande qui :

- communique avec le serveur d'API sur le master node
- envoie les fichiers YAML au serveur
- kubernetes applique la configuration

Exemples

Pour appliquer notre configuration

```
$ kubectl apply -f toto.yaml
$ kubectl apply -f toto.yaml -f titi.yaml
$ kubectl apply -f dossier-projet/
```

Pour vérifier la configuration

```
$ kubectl get pods
NAME                                READY STATUS RESTARTS AGE
service-...-api-764b8d9594-7m2n2  2/2   Running 0      3d
service-...-api-764b8d9594-884sl  2/2   Running 0      3d
```

```
$ kubectl get svc
NAME         TYPE        CUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
kubernetes  ClusterIP   100.64.0.1   <none>       443/TCP    72d
service-...-api LoadBalancer 100.64.14.251 aa4a30a359c... 80:30939/TCP 72d
```

```
$ kubectl get hpa
NAME           REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
service-...-api Deployment/...-api  12%/80%  2        4        2        18d
```


Quelle version choisir ?

- Kubernetes fonctionne partout
- Il y a de nombreuses façons de faire fonctionner Kubernetes
- Avec plus ou moins d'intégration pour certains fournisseurs cloud (AWS, GCE, etc.)
 - ex: les Volumes ou External Load Balancers ne fonctionnent qu'avec les fournisseurs cloud compatibles

Cloud publics managés

Les opérateurs cloud proposent de plus en plus des versions pré-installées et gérées de Kubernetes:

Opérateur	Nom du service
Amazon Web Services	Elastic Kubernetes Service (EKS)
Microsoft Azure	Azure Kubernetes Service (AKS)
Google Cloud Platform	Google Kubernetes Engine (GKE)
Digital Ocean	DigitalOcean Kubernetes Services (DOKS)
OVH	Kubernetes Services
Linode	...

Installation manuelle

- Kops
 - gere des clusters via les API de AWS, DigitalOcean, etc.
 - seulement pour Mac / Linux (pour Windows il faut une VM linux)
- [Kubeadm](#) - Outil de déploiement par défaut de K8S
- [K3s](#) - Kubernetes léger développé par Rancher Labs

- [k0s](#) - Kubernetes ultraléger développé par Mirantis
- [Metal3](#) -
- [Lokomotive](#) -
- [Typhoon](#) -
- [K8S Pharos](#) -
- [Tinkerbelle](#) -

Installation mono-noeud (dev/test local)

- minikube
 - démarrage rapide d'une machine avec un cluster kubernetes
- docker client
 - les versions récentes intègrent K8S
- kind (Kubernetes IN Docker)

References

- <https://stackoverflow.com/questions/42456159/minikube-volumes>

Google - Kubernetes Engine (GKE)

Pour utiliser Kubernetes sur Google Cloud, vous aurez besoin de la ligne de commande gcloud.

```
$ gcloud components update
$ gcloud config set project ...
$ gcloud container clusters create ...
$ gcloud compute instances list
$ gcloud compute firewall-rules create demo-k8s --allow tcp:31481
```

Amazon - Elastic Kubernetes Service (AKS)

Azure - Kubernetes Services (AKS)

Se connecter sur AKS

```
az login
```

Créer un groupe de ressources

```
az group create \  
  --name MyResourceGroup \  
  --location francecentral
```

Créer un cluster (la version gratuite est limitée à 2 noeuds)

```
az aks create \  
  -g MyResourceGroup \  
  -n MyManagedCluster \  
  --generate-ssh-keys --node-count 2
```

Attendre 3-5 minutes... que les nodes soient créés

Se connecter au cluster (pour .kube/config)

```
az aks get-credentials \  
  --resource-group MyResourceGroup \  
  --name MyManagedCluster
```

Vérifier que les nodes sont prêts

```
kubectl get nodes
```

Précautions d'emploi

Load Balancers

- <https://docs.microsoft.com/fr-fr/azure/aks/load-balancer-standard>

Pour un exemple complet

- <https://itnext.io/running-your-microservices-securely-on-aks-417a110b2e76?sk=40002aac0f7d5af48fc781c844cfb9ba>

- <https://medium.com/microsoftazure/secure-your-microservices-on-aks-part-2-5496bf2ba00c>
- <https://medium.com/microsoftazure/secure-your-microservices-on-aks-part-3-the-network-dfde7d26af8c>

Digital Ocean - Kubernetes Services (DOKS)

OVH - Managed Kubernetes

Kubeadm

Kubeadm est un outil conçu pour fournir `kubeadm init` et `kubeadm join`. Il se veut un accélérateur de bonnes pratiques pour la création de clusters Kubernetes.

- Kubeadm réalise les actions nécessaires à la mise en place d'un cluster minimum viable et opérationnel.
- De par sa conception, il ne se soucie que du bootstrapping, et non des machines d'approvisionnement.
- De même, l'installation de divers addons pratiques, comme le tableau de bord Kubernetes, les solutions de surveillance et les addons spécifiques au cloud, n'est pas concernée.

Références

- <https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/>

Rancher

Distribution Kubernetes :

- on-Premise
- multi-orchestrateurs
- multi-clouds

Références

- <https://rancher.com/docs/rancher/v2.x/en/installation/>

Openshift

Distribution Kubernetes développée par Redhat

- Pour les cloud on-premise,
- Conforme à la norme de sécurité financière PCI DSS

Références

- [Wikipedia: OpenShift](#)
- [Wikipedia: Norme de sécurité de l'industrie des cartes de paiement](#)

K3s

« Kubernetes pour l'IoT »

Développé par Rancher Labs

Références

- <https://k3s.io/>

Docker client

La récente version de Docker Desktop (Version Edge sous Windows/macOS 18.06.0-ce-mac70 CE) comprend un serveur et un client Kubernetes autonomes, ainsi qu'une intégration Docker CLI.

Le serveur Kubernetes :

- Il s'exécute localement dans votre instance Docker,
- Il n'est pas configurable,
- Il ne possède qu'un seul noeud

Pour l'activer, aller dans `Préférences > Kubernetes` :

- Activer le service
- Appliquer

Références

- [Docker Documentation: Deploy on Kubernetes \(Mac\)](#)
- [Docker Documentation: Deploy on Kubernetes \(Windows\)](#)

Minikube

Intérêt

- Un système permettant de faciliter l'installation locale de Kubernetes
- Lance un cluster avec 1 seul nœud dans une VM Linux
- Pour tester Kubernetes ou pour le développement
- Pas pour la production, pas de load-balancer, pas de haute disponibilité
- Sert à apprendre, faire des tests ou des démos

Fonctionnement de Minikube

- Fonctionne sur Linux, Windows et MacOS
- Nécessite un outil de virtualisation, au choix :

Système d'exploitation	Hyperviseurs supportés
macOS	VirtualBox, VMware Fusion, HyperKit
Linux	VirtualBox, KVM
Windows	VirtualBox, Hyper-V

Installation

Voir <https://kubernetes.io/fr/docs/tasks/tools/install-minikube/>

Sous Linux (ici Debian/Ubuntu)

```
$ apt-get install virtualbox
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
$ chmod +x minikube
$ sudo cp minikube /usr/local/bin && rm minikube
```

Sous Microsoft Windows

- Installation
 - depuis <https://github.com/kubernetes/minikube>
 - depuis chocolatey

```
$ choco install minikube kubernetes-cli
```

Sous MacOS

Avec Homebrew:

```
$ brew install minikube
```

Sans Homebrew

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-darwin-amd64
$ chmod +x minikube
$ sudo mv minikube /usr/local/bin
```

Démarrage rapide

Pour démarrer le cluster :

```
$ minikube start
```

Kind

« Kubernetes IN Docker »

Intérêt

- Un systeme permettant de faciliter l'installation locale de Kubernetes
- Lance un cluster avec autant de noeuds que vous voulez dans des Containers Docker
- Pour tester Kubernetes ou pour le developpement
- Pas pour la production, pas de load-balancer, pas de haute disponibilité
- Sert à apprendre, faire des tests ou des démos

Configuration pour un cluster multi-noeuds

Références

- <https://kind.sigs.k8s.io/docs/user/quick-start/>

040 k3d

Setup de kubectl

- Linux: <https://storage.googleapis.com/kubernetes-release/release/v1.11.0/bin/linux/amd64/kubectl>
- MacOS: <https://storage.googleapis.com/kubernetes-release/release/v1.11.0/bin/darwin/amd64/kubectl>
- Windows: <https://storage.googleapis.com/kubernetes-release/release/v1.11.0/bin/windows/amd64/kubectl.exe>
- Or use a packaged version for your OS: see <https://kubernetes.io/docs/tasks/tools/install-kubectl/>

```
$ choco install kubernetes-cli
```


Configuration et contextes

Une fois le kubecfg de votre opérateur récupéré

Vérifier la configuration

Regarder la configuration dans `~/.kube/config`

```
$ cat ~/.kube/config  
... (fixme)
```

Vérifier le contexte actuel

Contextes d'utilisation

Dans le cas où vous utilisez plusieurs clusters

```
$ kubectl config get-contexts  
$ kubectl config set-context  
$ kubectl config current-context  
$ kubectl config ...
```

Et surtout

```
$ kubectl config use-context <...>
```

Vous pouvez aussi installer l'outil `kubectx` qui facilite le travail

Fusionner la configuration d'un nouveau cluster avec l'ancienne

```
# Merge the two config files together into a new config file  
$ cp ~/.kube/config ~/.kube/config.bak  
  
# Replace your old config with the new merged config  
$ KUBECONFIG=~/.kube/config:~/.kube/config.kubectx kubectl config view --flatten > /tmp/  
config  
  
# (optional) Delete the backup once you confirm everything worked ok  
$ mv /tmp/config ~/.kube/config  
$ rm ~/.kube/config.bak
```

Premiers pas

Test du cluster

On va lancer un echo server, pour vérifier que tout se passe bien

```
$ kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --port=8080
```

```
$ kubectl expose pod hello-minikube --type=NodePort
```

```
$ minikube service hello-minikube --url
```

Ouvrir un navigateur sur l'URL qui s'affiche sur la dernière commande

```
$ curl http://...url..
```

Important

Pour obtenir l'url du service, l'outil minikube (fourni uniquement pour le développement) utilise les informations que vous pouvez trouver par vous même avec la commande `kubectl get nodes -o wide` et `kubectl get services`

Obtenir de l'information

 **Documentation officielle Kubernetes**

 **Comment voir plus d'informations dans les listes (-o wide)**

 **Comment voir les détails d'informations dans les listes (-o yaml)**

 **La commande describe**

 **La commande explain**

Cluster

Voir l'état du cluster

```
kubectl cluster-info
```

Namespaces

Lister les namespaces

```
$ kubectl get namespace
NAME          STATUS AGE
default       Active 1d
kube-node-lease Active 1d
kube-public   Active 1d
kube-system   Active 1d
```

Créer un namespace

```
$ kubectl create namespace demo
```

Utiliser un namespace

Soit utiliser ce namespace au cas par cas

```
$ kubectl create .... --namespace demo
```

Soit définir ce namespace par défaut pour toutes les commandes qui suivent

```
$ kubens demo
```

Références

- [Kubernetes Documentation: Namespaces](#)

Nodes

Rôle des noeuds

Parmi ces nodes, un (ou plusieurs) master node, pour les fonctionnalités internes de Kubernetes :

- Controler Manager
- API server
- Scheduler
- Docker / ContainerD / CRI-O / Podman

Sur les autres nodes, worker nodes, qui servent à faire fonctionner les applications :

- Proxy
- Kubelet
- Docker / ContainerD / CRI-O / Podman

Voir les noeuds du cluster

```
$ kubectl get nodes
```


Les pods

- Ces containers
 - sont toujours démarrés, arrêtés ou répliqués en groupe
 - partagent le même network namespace (meme IP et port)
 - peuvent communiquer ensemble en utilisant localhost
 - peuvent partager des données via des Shared Volumes

Plein d'options

- Replicas (ex: 3 répliques)
- Sécurité (ex: pas en root)
- Liveness probe
 - ex: est-ce que mes pods sont capables de répondre?
 - ex: est-ce qu'on doit les tuer pour en relancer?
- Readiness probe
- Volumes (espace de stockage qui peut être attaché au pod)
 - ex: idem que docker
- Requests
 - ex: combien de mémoire on a le droit de consommer
 - ex: combien de CPU on a le droit de consommer
- Limits
 - ex: tuer mon application si elle consomme trop de RAM
- Envvars
 - ex: injecter de la configuration
- Stratégie de MAJ
 - ex: est-ce que je tue d'abord mes pods et ensuite je déploie ? (downtime)
 - ex: mise à jour progressive ? (j'enleve petit à petit des pods)
 - ex: lancer des scripts au lancement, à l'arrêt, etc.
- Hook start/stop

Note

C'est la primitive sur laquelle vous allez passer le plus de temps :

Vous allez passer des jours, des semaines à définir la configuration de vos pods 😊

- "frontiere d'une application"

Créer un pod

```
kubectl create -f pod-...yaml
```

Obtenir l'état des pods en cours

```
$ kubectl get pod
```

Décrire (de façon détaillée) un pod

```
$ kubectl describe pod <pod>
```

Exposer le port d'un Pod (= crée un nouveau service)

```
$ kubectl expose pod <pod> --port=<port> --name=frontend
```

Redirige le port exposé du pod vers la machine locale

```
$ kubectl port-forward <pod> <localport>[:<remoteport>] ...
```

Attache le pod localement

```
$ kubectl attach <pod> [-i] [-c <container>]
```

Executer une commande dans le pod

```
kubectl exec <pod> -- command
```

Ajouter un nouveau label à un pod

```
kubectl label pods <pod> mylabel=awesome
```

Lancer un shell dans un pod (pratique pour déboguer)

```
kubectl run -it busybox2 --image=busybox --restart=Never --rm -- sh
```



Démo 020 - Créer un Pod

Services


Créer le service en exposant un pod

Créer le service d'après un YAML

```
$ kubectl create -f mywebapp-service.yaml  
service "mywebapp-service" created
```

Récupérer l'adresse externe du service (et attendre un peu que le pod soit complètement démarré)

```
$ kubectl get services  
NAME          CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE  
kubernetes    10.3.240.1   <none>        443/TCP   3h  
mywebapp-service 10.3.252.113 104.192.250.211 8080/TCP 1m
```

 **Démo 030 - Créer un Service**

 **a propos du load-balancing (service extérieur)**

Architecture de K8S

Kubelet

- Responsable de lancer les Pods
- Connecté au Master Node pour obtenir les informations

Kube-Proxy

- Gère IPTables

Déploiements

- Permet de faire des déploiement & des updates
- Définit l'état final de l'application
 - K8S s'occupe de faire correspondre l'état du cluster à l'état souhaité
- Facilite l'utilisation des replication controller / replication set
- Apport plus de possibilités
 - Créer un déploiement
 - Update un déploiement
 - Rolling updates (zero downtimes)
 - Rollback a une version précédente
 - Pause / Resume un déploiement (ex: 20%)

Configuration

Commandes

Obtenir les infos sur les Deployment

```
$ kubectl get deployments
```

Obtenir des infos sur les ReplicaSets

```
$ kubectl get rs
```

```
$ kubectl get pods --show-labels
```

Obtenir l'état du déploiement

```
kubectl rollout status <deployment>
```

Changer l'image

```
kubectl set image <deployment> k8s-demo=k8s-demo:2
```

Editer l'objet

```
kubecttl edit <deployment>
```

Démo



Démo de déploiement

Volumes et données

Basics deployments

Basics labels

Basics node architecture

Basics pods state lifecycle

Basics secrets

Basics services

Sécurité sous Kubernetes

Gestion des données

Passer des informations sensible ou de configuration au Pod Trop de dev. mettent des détails de config ou des passwords dans un container et le publient sur dockerhub !!!

Mieux: Utiliser des Secret et ConfigMaps qui seront montés dans le container en tant que volumes au démarrage.

Gestion des processus

Sur le node01:

```
$ ps aux |grep -B1 leprocess
```

=> le process est lancé en root !

ConfigMap

Créer la configmap

```
$ mkdir config-mywebapp  
$ cd config-mywebapp
```

```
$ vi machin1.config  
$ vi machin2.config
```

On crée la configmap d'après le dossier config-webtestapp

```
$ kubectl create configmap myconfigmap --from-file=config-webtestapp
```

Pour vérifier le contenu de la configmap

```
$ kubectl get configmaps myconfigmap -o yaml  
apiVersion: v1  
data:  
  machin1.config: |  
    contenu du fichier  
  ...  
  machin2.config: |  
    contenu du fichier  
  ...
```

Utiliser la configmap

vi machin.yml

```
spec:  
  template:  
    spec:  
      containers:  
        # ...  
        - image: gcr.io/projectID/tomcat7:1  
          name: tomcat  
          volumeMounts:  
            - mountPath: /usr/local/tomcat/webapps  
              name: app-volume  
            + - mountPath: /config/machin  
              + name: config-volume  
          volumes:  
            - name: app-volume  
              emptyDir: { }  
            + - name: config-volume
```

```
+ configMap:
+   name: myconfigmap
```

Verifier

Se connecter sur le pod et aller voir dans le dossier

```
$ kubectl exec -it mywebapp-deploy-...-... bash
root@mywebapp-deploy-...-...:/usr/local/tomcat#
root@mywebapp-deploy-...-...:/usr/local/tomcat# ls /config/machin
machin1.config machin2.config
root@mywebapp-deploy-...-...:/usr/local/tomcat#
```

Astuce

Si les pods sont bloqués dans l'état `ContainerCreating` state, vérifier que le ConfigMap / Secret associé est bien disponible.

Secrets

Note

Demonstration de création des différents types de secrets

Note

Comparaison avec les ConfigMap lors de l'utilisation dans l'environnement ou en tant que volumes

- `configMapRef => secretRef`
- `configMapKeyRef => secretKeyRef`

Helm

Helm is the package manager for Kubernetes, it makes deploying complex application workloads simple, helps organize the update process.

Références

- <https://github.com/cdwv/awesome-helm>
- <https://github.com/KarolNet/awesome-helm>
- <https://hub.helm.sh/charts/stable/chaoskube>

Créer des charts Helm

Repository Helm

Construire et déployer

Fission

<https://github.com/fission/fission>

<https://docs.fission.io/installation/kubernetessetup/>

<https://docs.fission.io/installation/>

010 auto discovery

030 configmap

040 daemon sets

050 ext dns

Requêtes et limites

Enjeux et vocabulaire

Qu'est-ce qui se passe si j'ai une pénurie de ressource ?

Attention, il ne s'agit pas d'un minimum/maximum !

Request

Utilisé au moment de la mise en place des Pods :

- Requests n'est utilisé que pour le placement et crée une carte théorique du cluster.
- Kubernetes recherchera un nœud qui a à la fois assez de CPU et de mémoire en fonction de la configuration des Request.
- Kubernetes s'assure que la somme des ressources demandées pour un noeud est égale ou inférieure à la capacité du noeud. Ce n'est pas un minimum. Notre conteneur pourrait en fait en utiliser moins. C'est un indice de ce dont il a besoin.

Limits

Appliqué au moment de l'exécution :

- Si un conteneur dépasse les limites, Kubernetes essaiera de l'arrêter.
- Pour le CPU, il limitera simplement l'utilisation de sorte qu'un conteneur ne pourra généralement pas dépasser sa capacité limite ; il ne sera pas tué, il ne pourra simplement pas utiliser plus de CPU.
- Si un conteneur dépasse ses limites de mémoire, il pourra être arrêté.

Mise en oeuvre

pod--webapp.yml

```
apiVersion: v1
kind: Pod
metadata:
  # ...
spec:
```

```
containers:
- image: gcr.io/projectID/webapps:2
  name: webapps
resources:
  requests:
    memory: "128Mi"
    cpu: "10m"
  limits:
    cpu: "2"
    memory: "192Mi"
# ...
```

Important

Que veut dire limite à `cpu: "2"` quand on utilise une infrastructure Cloud ?

Il ne s'agit pas de CPU réels ! On sait pas réellement sur quel machine hardware le programme fonctionne in fine ?

Le quantité de CPU est exprimé en milliers de milicpu (ou milicore). Un processus ne pourra pas consommer plus que le temps CPU qui lui est accordé.

Kubernetes mesure le temps consommé sur le CPU toutes les 100ms. La quantité de temps réelle qu'un container peut utiliser le CPU est donc le pro-rata. est donc

On appelle ça du Throttling.

- 100ms de temps CPU => `cpu: 1` ou `cpu: 1000m`
- 10ms de temps CPU => `cpu: 0.1` ou `cpu: 100m`
- 20ms de temps CPU => `cpu: 0.2` ou `cpu: 200m`

Auto-tuning de programmes

Attention aux programmes qui font de l'auto-tuning (ex: pour lancer des threads) !

Ils vont voir le "vrai" nombre de CPU, sans avoir l'information sur les quotas, donc ils vont se planter.

Voir plutot dans `/sys/fs/cgroup/cpu/`

```
$ cat /sys/fs/cgroup/cpu/cpu.cfs_quota_us
200000
$ cat /sys/fs/cgroup/cpu/cpu.cfs_period_us
100000
$ expr 20000 / 100000
2 // nombre CPU réellement disponibles
```

Métriques CPU

Pour vérifier la consommation réelle avec kubectl

```
$ kubectl top pod cpu-demo --namespace=cpu-example
NAME          CPU(cores)   MEMORY(bytes)
cpu-demo      974m        <something>
```

Sinon dans /sys/fs/cgroup/cpu

```
$ cat /sys/fs/cgroup/cpu/cpu.stat

user 1637

system 88

nr_periods 520

nr_throttled 364 <-- nombre de fois que la tâche a été freinée

throttled_time 72988838516 <-- temps (en ns) durant lequel la tâche a été freinée
```

Métriques mémoire

```
$ cat /sys/fs/cgroup/memory/memory.limit_in_bytes
402653184 <-- 384MB maximum
```

Comment collecter les métriques

Voir Prometheus

En cas de dépassement

Voir exemple sur : <https://kubernetes.io/fr/docs/tasks/configure-pod-container/>

References

- <https://medium.com/@betz.mark/understanding-resource-limits-in-kubernetes-cpu-time-9eff74d3161b>
- <https://kubernetes.io/fr/docs/tasks/configure-pod-container/>
- <https://vincentlauzon.com/2019/04/02/requests-vs-limits-in-kubernetes/>
- <https://kubernetes.io/docs/tasks/configure-pod-container/assign-cpu-resource/>

- <https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/>

Quotas

Références

- <https://kubernetes.io/docs/concepts/policy/resource-quotas/>

Redimensionnement

Redimensionnement horizontal

- Lancer plusieurs fois l'application
- Pour les application sans état (stateless) seulement
 - pas d'écriture de fichiers
 - pas de données stockées
 - sinon probleme de synchro
- La plupart des applications peut etre rendues stateless
 - Faire la gestion des sessions hors du container (ex: memcache, redis, ...)
 - Pas de fichiers en local (ex: volumes, nfs, s3, ...)
- Sinon penser au redimensionnement vertical (+ CPU, +memoire, +disk)
- Si possible, utiliser les pratiques 12factor

Replication Controler

- S'occupe du scaling horizontal
- Garantit un nombre de réplique de Pod
- Remplace les Pod s'ils tombent en panne, s'arretent ou sont supprimés
- Recommandé même pour apporter des garanties sur un seul Pod

```
$ kubectl create -f mywebapp-deploy-3.yaml
deployment "mywebapp-deploy" created

$ kubectl create -f mywebapp-service.yaml
service "mywebapp-service" create

$ # On vérifie les pods lancés
$ kubectl get pods
NAME                                READY STATUS  RESTARTS AGE
mywebapp-deploy-2243...-...  2/2    Running  0       39s
```

Replication Set

- Replication Controler Next Gen

- Permet le filtrage avancé
 - ex: environment = "dev" OU "qa" (et pas seulement égalité)
- Utilisé par l'objet Deployment

Manuellement

On va faire passer le nombre de pods à 2

```
$ kubectl scale --replicas=2 rc/mywebapp-rc
replicationcontroller "mywebapp-rc" scaled
```

Vérifier maintenant le nombre de pods

```
$ kubectl get pods
NAME                                READY STATUS RESTARTS AGE
mywebapp-deploy-2243...-8dfc9 2/2   Running 0      10m
mywebapp-deploy-2243...-jshbm 2/2   Running 0      17s
```

Impact sur les visiteurs

On change le Session Affinity : ClientIP à None pour ne pas attacher un pod spécifique à un visiteur (=sticky)

```
$ vi mywebapp-service.yaml
(...)
spec:
  ports:
    - port: 8080
  type: LoadBalancer
  sessionAffinity: None
(...)
```

On applique les changements

```
$ kubectl apply -f mywebapp-service.yaml
service "mywebapp-service" configured
```

On vérifie qu'avec plusieurs requetes le nom du service change

```
$ wget -qO- http://35.184.185.4:8080/webTestApp/front?action=status | grep name
name:1@mywebapp-deploy-...-jshbm uptime:789 ...
$ wget -qO- http://35.184.185.4:8080/webTestApp/front?action=status | grep name
name:1@mywebapp-deploy-...-8dfc9 uptime:789 ...
$ wget -qO- http://35.184.185.4:8080/webTestApp/front?action=status | grep name
name:1@mywebapp-deploy-...-jshbm uptime:789 ...
```

Astuce

La quantité de pod peut être redimensionnée automatiquement, en se basant sur des mesures classiques (cpu, usage, etc.) ou sur-mesure en utilisant le `HorizontalPodAutoscaler` .

Astuce

La quantité de noeuds peut également être redimensionnée automatiquement, en utilisant `ClusterAutoscaling`

Redimensionnement automatique

Horizontal (HPA)

```
```diff tab="5-hpa.yml" metadata: name: hpa-lab namespace: lab spec: +
scaleTargetRef: + apiVersion: extensions/v1beta + kind: Deployment + name: lab-
java-deployment + minReplicas: 1 + maxReplicas: 3 metrics: - type: Object object:
target: kind: Service name: lab-java-service metricName: endpoint_hello
targetValue: 100000m
```

```
```shell-session
$ kubectl apply -f 5-hpa.yml
horizontalpodautoscaler.autoscaling "hpa-lab" created
$ kubectl get horizontalpodautoscaler hpa-lab -w
NAME      REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
hpa-lab   Deployment/lab-java-deployment  599m/100  1        3        1        32s
hpa-lab   Deployment/lab-java-deployment  600m/100  1        3        1        35s
hpa-lab   Deployment/lab-java-deployment  599m/100  1        3        1        55s
...
```

Redimensionnement automatique Vertical (VPA)

Redimensionnement automatique du Cluster (CA)

La procédure dépend de l'hébergeur sous-jacent. D'après le github de Kubernetes

In general, to support Cluster Autoscaler in a new environment, you'll need to implement its cloud provider interface for that environment. This is described in more detail in #953.

On peut (presque) contourner avec :

- <https://github.com/elsonrodriguez/harbormaster>
- <https://www.nivenly.com/meanwhile-in-the-kubicorn-repository/>

Références

- <https://medium.com/kubecost/understanding-kubernetes-cluster-autoscaling-675099a1db92>
- <https://dzone.com/articles/kubernetes-autoscaling-101-cluster-autoscaler-hori-1>

Readiness

Expliquer à K8S à partir de quand l'application est prête

Deux tests :

- livenessProbe : savoir que le container est en vie
- readinessProbe: savoir que le container est bien démarré (avant d'ouvrir les flux, notamment dans le cas d'un HPA)

Pour vérifier en HTTP

```
spec:
  containers:
    - name: lab
      image: glenux/whatever:v10
      + livenessProbe:
      +   httpGet:
      +     path: /hello
      +     port: 8080
      + readinessProbe:
      +   httpGet:
      +     path: /hello
      +     port: 8080
      +   initialDelaySeconds: 5
      +   periodSeconds: 2
      ports:
        - containerPort: 8080
      # ...
```

Probleme : si le code HTTP < 400, le container est tué et redémarré.

Important

Eviter de tester l'ensemble des routes ici !

Pour vérifier (simplement)

```
spec:
  containers:
    - name: lab
      image: glenux/whatever:v10
      + livenessProbe:
      +   tcpSocket:
```

```
+   port: 8080
+   readinessProbe:
+     tcpSocket:
+       port: 8080
+     initialDelaySeconds: 5
+     periodSeconds: 2
  ports:
    - containerPort: 8080
  # ...
```

Evenemnts

```
$ kubectl get events
```

On peut voir les différents elements de kubernetes :

- default-scheduler => choisir un noeud
- kubelet => pull, creation, start
- deployment-controler => scale

Basics healthchecks

Basics readiness

Basics scaling pods

Basics webui

070 interpod anti affinity

080 monitoring

Ingress

- TODO https://codeburst.io/kubernetes-ingress-simply-visually-explained-d9cad44e4419?sk=e8ca596700f5b58c7ab0d85d4dab6386&source=friends_link&gi=dfb5bd211b5
- TODO [Contour - Ingress with Envoy | octetz](#)
- TODO <https://blog.stephane-robert.info/post/kubernetes-ingress-nginx-baremetal/>

Egress

090 node affinity

100 operators

Volumes avancés



- Volumes avancés avec les "StorageClass"
- Les "StatefulSets"

110 pod presets

Authentication et autorisation

- Les identités dans K8S
- Les méthodes d'authentification
- ServiceAccounts et tokens
- Les modèles d'autorisation
- Administration RBAC (Role-Based Access Control)

120 stateful sets

Kubernetes Dashboard

Ancienne méthode

Récupérer les infos de la master URL

```
kubectl cluster-info ksmall
Kubernetes master is running at ...
GLBCDefaultBackend is running at ...
(..)
kubernetes-dashboard is running at https:// ...
```

Pour obtenir les infos d'authentification paramétrées par votre hébergeur cloud :
(ici GCP) :

```
gcloud container clusters describe ksmall
clusterIpv4Cidr: 10.40.0.0/14
endpoint: 35.184.22.61
(...)
masterAuth:
  password: ...
  username: ...
(...)
```

Enfin ouvrir l'url en ajoutant /ui à la master Url : <https://35.184.22.61/ui>

Nouvelle méthode

```
$ kubectl proxy --port=8081
Starting to serve on 127.0.0.1:8081
```


Prometheus

Base de donnée pour Time-Series.

Spécialisé dans la collecte de métriques. S'integre tres bien à K8S

Il requete en HTTP des composant de kubernetes pour récupérer les métriques

Plein de sondes qui vont chercher plein d'infos dans le cluster

Permet de faire du Grafana, de l'alerte, etc.

Les informations utiles

- container_cpu_cfs_throttled_periods_total
- container_cpu_cfs_throttled_seconds_total
- container_memory_failcnt

=> permet de voir la pression sur les containers.

- soit parce qu'on a pas payé assez cher la machine :-)
- soit parce qu'on a mis une limite au hasard au début :-)

Coté applicatif

Implémenter `/metrics` :

```
endpoint_hello_total{status="get", } 1606.0
```

Ajouter dnas vos metadata:

```
kind: Service
apiVersion: v1
metadata:
  name: lab-java-service
+ annotations:
+   prometheus.io/scope: "true"
spec:
  selector:
    app: lab-java
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

Reading multiple logs (from replicas)

Kubetail

<https://github.com/johanhaleby/kubetail>

Stern

<https://github.com/wercker/stern>

130 taints tolerations

140 volumes

Human coders

Ref. <https://www.humancoders.com/formations/kubernetes-avance#outline>

Jour 1 : gérer ses apps Kubernetes efficacement Packager son app

- La pipeline de déploiement Skaffold
- Les registres privés
- Le système de charts Helm en détails

Mise en place d'une solution GitOps

- Concepts de GitOps et installation d'ArgoCD
- Releases Helm et manifestes Kubernetes intégrés comme apps dans ArgoCD
- Registre avancé avec Harbor

Mise en pratique (à titre indicatif, peut varier en fonction de chaque groupe) : -
Skaffold et la pipeline de développement - Les registres privés avec Harbor -
Création d'une chart Helm - Mise à jour des ressources de notre cluster via un
dépôt Git et l'installation d'ArgoCD Jour 2 : des déploiements plus complexes et un
cluster mieux supervisé La supervision du cluster

- Prometheus, kube-metrics et Grafana
- Elasticsearch (EFK) et la centralisation des logs dans les pods

Le réseau avancé

- Istio et les services meshes pour l'introspection réseau, les politiques de routage et les stratégies de déploiement avancées
- les NetworkPolicies
- les différents plugins de réseau (CNI)

La sécurité dans Kubernetes

- La gestion de droits RBAC et l'ajout de comptes dans Kubernetes via certificats
- Planification des pods avec les Labels et les Affinités, les Taints et Tolerations
- Les règles de sécurité des pods avec Gatekeeper et les Linux capabilities
- Actualité de la sécurité sur Kubernetes

Les clusters multi-nodes stockage

- Découverte de Rancher et des clusters multi-nodes
- Du stockage distribué on premise avec Rook et Ceph

Mise en pratique (à titre indicatif, peut varier en fonction de chaque groupe) : -
Centralisation des logs avec EFK - Monitoring Kubernetes avec Prometheus et
Grafana - Création d'un cluster multi-nœuds avec Rancher - Mise en place d'une
solution de volumes distribués avec Rook - Stratégies de déploiement avancées
avec Istio - Ajout de NetworkPolicies au cluster - Paramétrage des droits RBAC et
ajout d'users

M2i

Jour 1

Rappels sur les fondamentaux de Kubernetes

Rappel des ressources Kubernetes
Dernières nouveautés Kubernetes
Tour d'horizon de l'écosystème Kubernetes

Gestion des volumes avancés

Volumes avancés avec les "StorageClass"
Les "StatefulSets"

Exemples de travaux pratiques (à titre indicatif)

Volumes avancés
"StatefulSets"

Authentification et autorisation

Les identités dans K8S
Les méthodes d'authentification
ServiceAccounts et tokens
Les modèles d'autorisation
Administration RBAC (Role-Based Access Control)

Exemple de travaux pratiques (à titre indicatif)

Gestion de l'authentification et des autorisations

Maîtrise des capacités

Les capacités du cluster
Les "LimitRanges"
Les "ResourceQuotas"

Exemple de travaux pratiques (à titre indicatif)

Quotas et limitations des ressources

Jour 2 Monitoring

Principes sur le monitoring
Prometheus
Grafana

Exemples de travaux pratiques (à titre indicatif)

Déploiement et configuration du monitoring
Création de dashboard

Gestion des logs

Production des logs applicatifs
Les différentes solutions
Le modèle EFK (Elasticsearch, Fluentd et Kibana)

Exemple de travaux pratiques (à titre indicatif)

Déploiement et configuration d'EFK

Audit

Production des logs d'audit Kubernetes
Analyse des logs

Exemples de travaux pratiques (à titre indicatif)

Mise en place de l'auditing au sein du cluster
Visualisation des logs Kubernetes

Architecture avancée

Présentation des concepts d'architecture avancée de Kubernetes
Disponibilité des composants Kubernetes
Bonnes pratiques
Optimiser sa gestion du cluster
Cycle de vie du cluster
Mettre à jour son cluster Kubernetes

Exemple de travaux pratiques (à titre indicatif)

Architecture avancée et cycle de vie du cluster Kubernetes

Packaging applicatif avec Helm

Présentation des fonctionnalités de packaging de Helm
Organisation des manifests Kubernetes en charts

Exemple de travaux pratiques (à titre indicatif)

Création d'un package applicatif avec Helm

Registre avancé avec Harbor

Présentation des fonctionnalités du registre Harbor
Organisation des objets (conteneurs, charts...) dans Harbor
Fonctionnalité de scan de sécurité des images Docker

Exemple de travaux pratiques (à titre indicatif)

Déploiement et utilisation de Harbor

010 high availability

Installation / Deployment

- TODO <https://enix.io/fr/blog/deployer-kubernetes-1-13-sur-openstack-grace-a-terraform/>
- TODO <https://dzone.com/articles/advanced-kubernetes-deployment-strategies>

010 master services

010 namespaces

010 networking

010 node maintenance

010 quotas limits

RBAC

- TODO [Kubernetes - Le controle d'accès basé sur les rôles \(RBAC\) - Partie 1](#)
- TODO <https://www.youtube.com/watch?v=TZ73EBP2a9Q>
- TODO <https://stackoverflow.com/questions/52268462/limit-access-to-kubernetes-secret-by-rbac>
- TODO <https://stackoverflow.com/questions/52268462/limit-access-to-kubernetes-secret-by-rbac>
- TODO <https://www.cncf.io/wp-content/uploads/2020/08/RBAC-Online-Talk.pdf>

010 tls on elb

010 user management

Zzz fixme

Security

Comment appliquer le principe de moindre privilege ?

Security Context

Un Security Context définit les paramètres de privilèges et de contrôle d'accès pour un Pod ou un Container, cad :

- User ID
- Linux Capabilities
- SELinux labels
- AllowPrivilegeEscalation

Mise en place

```
spec:
+ # Pod Security Context
+ securityContext:
+   runAsNonRoot: true
+   runAsUser: 1234
+   fsGroup: 2000
  containers:
    - name: lab
      image: blabla
+   # Container Security Context
+   securityContext:
+     allowPrivilegeEscalation: false
    ports:
      - containerPort: 8080
```

Comment passer à l'échelle ?

Comment mettre en place des SecurityContext automatisé ?

Dans OpenShift :

- SCCs : permettent d'appliquer un contexte de sécurité par défaut sur les PODs.
- Plusieurs sont fournis par défaut (dont "Restricted")

Dans K8s :

- PSP : Pod Security Policy est une ressource de niveau cluster qui contrôle les aspects les plus critiques de la sécurité des spécifications du pod.
- RTFM : à vous de les contruire :-)

En résumé

- Important de comprendre les SecurityContext, travailler avec les OPS sur la mise en oeuvre des PSP (ou utiliser OpenShift)
- SELinux : éviter `run setenforce 0`
- Utiliser des namespaces dédiés
- Utiliser des ServiceAccount : des comptes techniques qui permettront de jouer avec les RBAC
- Quelle sécurité pour les flux applicatifs ? TLS de bout en bout ?

Arborescence de vos projets

```
.cloud/  
  docker/  
    nginx/Dockerfile  
    php/Dockerfile  
  jenkins/  
    Jenkinsfile  
  kubernetes/  
    deployment.yaml  
  terraform/  
    vars/  
    rds.tf
```

Integration continue

For development

Kubefwd

- <https://github.com/txn2/kubefwd>

Telepresence

- <https://telepresence.io>

FIXME: faire comme si le container local était sur le cluster ?

Squash

- <https://github.com/solo-io/squash>

Tools

Kube-Score

- <https://github.com/zegl/kube-score>

Kubesecc

- kubesecc.io

Helm Unittest

- <https://github.com/lrills/helm-unittest>

Helm Diff

- <https://github.com/databus23/helm-diff>

Dépannage

Other tools

- github.com/GoogleContainerTool
- cloud.google.com/cloud-code
- github.com/cloudnativelabs/kube-shell
- `kubectl diff`
- `krew`
- github.com/aylei/kubectl-debug
- [kubernetes-sigs/kustomize](https://kubernetes-sigs.github.io/kustomize/)

Pod Disruption Budget (aka PDB)

En cas de "disruption" "volontaire" cela permet de maintenir le minimum d'instances

Cas d'usage :

- opération de maintenance par des Ops (kubectl drain sur les machines)

Permet d'apporter une garantie globale sur l'infrastructure (meme pendant que les Pods déménagent d'un Node à l'autre)

```
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: lab-java-pdb
spec:
  minAvailable: 1
  selector:
    matchLabels:
      app: lab-java
```


K8s school

INTRODUCTION AUX MICRO-SERVICES ● Les bonnes pratiques: la méthodologie des “12 facteurs” ● Application monolithique versus Micro-services ● Faire évoluer une application vers les micro-services

RAPPELS SUR LES CONTENEURS ● Vue d’ensemble de Docker ● Vue d’ensemble des conteneurs ● Installer et exécuter des images Docker ● Interagir avec des conteneurs ● Créer ses propres images ● Différence entre les dépôts privés et publics

KUBERNETES: LES BASES ● Créer un cluster Kubernetes: sur votre poste de travail, dans votre datacenter ou dans le cloud. ● Architecture et composants de Kubernetes (côté Control Plane et Node) ● Cycle de vie d’une requête kubectl ● Déployer une application sur plusieurs machines ● Explorer une application ● Passage à l’échelle

KUBERNETES: LES CONCEPTS FONDAMENTAUX ● Vue d’ensemble des Pods ● Interagir avec les Pods ● Configuration et sécurité d’une application (ConfigMaps et Secrets) ● Vue d’ensemble des Services (ClusterIP, NodePort, LoadBalancer, Headless) ● Créer ses propres services pour exposer ses applications ● Exposer une application sur le réseau ● Organiser ses Pods avec les Labels

DEPLOYER SES MICRO-SERVICES ● Stratégies de déploiement en mode “Cloud-Native” ● Stratégies de calcul intensif (Jobs) ● Cas pratique: déploiements avec kubectl et yaml ● Stratégies de passage à l’échelle (Replicasets et Daemonsets) ● Cas pratique: utilisation des réplicas ● Cas pratique: installation d’un gestionnaire de journaux distribués ● Stratégie de mise à jour logicielle (Deployments) ● Cas pratique: Rolling update ● Gérer simplement ses mises à jour applicatives ● Techniques avancées : déploiement blue/green, canary

CONCEPTS AVANCÉS ● Stockages volatiles et persistants (PersistentVolume/ PersistentVolumeClaim) ● Techniques de supervision avancées: Prometheus ● Déploiement des bases de données distribuées (StatefulSet) ● Cas pratique: installation de MongoDB et Redis en mode distribué

EN OPTION ● Les Services Mesh: fonctionnement et cas pratique avec Istio (½ jour) ● Ingress: fonctionnement et cas pratique avec Traefik (½ jour) ● Accompagnement et conseil sur des cas pratiques proposés par les stagiaires (½ à 1 jour) ● Services de gestion de conteneurs du Cloud public ou Multi-Cloud: les exemples de Google Kubernetes Engine et de Rancher (½ journée)

M2i

Programme Fondamentaux Historique

- Rappel des concepts du Cloud
- Comment comprendre les conteneurs par rapport à la virtualisation ?
- D'où vient le concept de conteneurs ?
- L'historique des conteneurs
- L'arrivée de Docker
- Le monde Windows
- Les orchestrateurs de conteneurs

Principes de fonctionnement

- Notions d'isolation
 - Cgroup
 - Namespaces
- Les conteneurs LXC
- Les conteneurs Docker
- Union File System et modèle en couches
- La couche Copy-On-Write (COW)

Technologies

- Composants de base d'une infrastructure Docker
- Définitions et terminologie Docker
- La notion d'OS minimaux
- Notion de Stateless / Stateful
- Comment gérer ses données ?
- Le cas du data-Only-Container
- Fonctionnement du réseau sur un hôte
- Fonction du réseau entre conteneurs

Container as a Service (CaaS) et orchestration Fondamentaux

- Comment lier des conteneurs ?
- Utilisation de Docker Compose
- Création d'une infrastructure composée de plusieurs conteneurs
- Mise en pratique

Technologies de conteneurs et CaaS

- Notions de base et définitions
 - Images
 - Couches
 - Conteneurs
 - Hub
 - Registry...

Modèle en couche "layering" et impacts
Gestion des déploiements
Présentation des solutions de clustering et d'orchestration
Kubernetes
Swarm
Mesos...

Bénéfices des conteneurs et du CaaS

Les bénéfices liés à la technologie
Les bénéfices pour les développeurs
Les bénéfices pour les administrateurs
Les bénéfices dans l'usage du Cloud et dans l'hybridation
Les bénéfices financiers
L'apport des conteneurs dans la démarche DevOps

Limites des conteneurs et du CaaS

Ces technologies sont-elles matures pour la production ?
La sécurité est-elle suffisante ?
Existe-t-il un risque de verrouillage ? L'interopérabilité est-elle réelle ?
Les communications réseaux entre conteneurs sont-elles optimales ?
Comment gérer les données avec Docker ?
Stateless vs Stateful
Le CaaS est-il préférable au PaaS ?

Kubernetes : les bases Fondamentaux

Historique
Google et Kubernetes
Les autres contributeurs : Red Hat...

La terminologie

Notion de pods
Notion de Replica Set et Replica Controller
Notion de services
Notion de volumes
Notion de ConfigMaps et secrets

Découverte de Kubernetes

Installation de Kubernetes sur un Cloud public : l'exemple de Google Container Service
Installation locale avec Minikube
Dashboard, CLI et API
Proxy et DNS
Démarrer ses premiers conteneurs

Kubernetes : mettre en oeuvre Les commandes usuelles

- Namespaces
- Contextes
- Visualiser les Kubernetes API Objects
- Gérer les objets Kubernetes
 - Création
 - Mise à jour
 - Suppression...
- Associer des labels aux objets
- Troubleshooting

Les pods

- Créer un pod et un pod manifest
- Gérer les pods (lister, supprimer...)
- Accéder à un pod
 - Port forwarding
 - Logs
 - Exec
 - Copier des fichiers
- Healthchecks
- Gestion des données persistantes et des volumes

Label et annotation

- Appliquer et modifier des labels
- Label selector
- Annotations
- Cleanup

Service Discovery

- Service DNS
- Intégration Cloud
- Kube-proxy
- Gestion du réseau au sein du cluster

Replica Set

- Replica Set et pods
- Créer un Replica Set
- Identifier un Replica Set dans un pod
- Rechercher un ensemble de pods pour un Replica Set
- Mettre à l'échelle les Replica Sets
 - kubectl scale
 - kubectl apply
 - autoscaling

ConfigMaps et secrets

- Créer et utiliser des ConfigMaps
- Créer et utiliser des secrets

- Contraintes de nommage
- Gérer les ConfigMaps et les secrets
 - Lister
 - Créer
 - Mettre à jour

Kubernetes : déployer des applications d'entreprise Son premier déploiement

- Créer, gérer, mettre à jour des déploiements
- Mettre à l'échelle des déploiements
- Stratégies de déploiement
 - Recreate
 - Rollingupdate
 - Rollout
- Supprimer un déploiement

Déployer des applications d'entreprise

- Déployer une application Web basée sur un service Web, une base de données, une base in-memory
- Configurer les composants
- Créer le service Kubernetes pour cette application
- Déployer le service applicatif
- Gérer le cluster

Certification (en option)

- Prévoir l'achat de la certification en supplément
- L'examen (en français) sera passé le dernier jour, à l'issue de la formation et s'effectuera en ligne
- Il s'agit d'un QCM dont la durée moyenne est d'1h30 et dont le score obtenu attestera d'un niveau de compétence
- La certification n'est plus éligible au CPF depuis le 31/12/2021, mais permettra néanmoins de valider vos acquis

Modalités d'évaluation des acquis

- En cours de formation, par des études de cas ou des travaux pratiques
- Et, en fin de formation, par un questionnaire d'auto-évaluation ou une certification (M2i ou éditeur)

Plb

Ref. <https://www.plb.fr/formation/open-source/formation-kubernetes,10-700808.php>

Introduction à Kubernetes

Présentation Kubernetes, origine du projet Fonctionnalités de base :
automatisation des déploiements et de la maintenance des applications en
containers Les différents containers supportés, plate-formes utilisant Kubernetes
Les composants essentiels de Kubernetes Quelques définitions importantes : pods,
labels, controllers, services Architecture de Kubernetes

Kubernetes Master : stockage des configurations par etcd, interfaçage par l'API
server Noeuds Kubernetes : hébergement des containers Utiliser Kubelet pour la
supervision des noeuds Installation et configuration de Kubernetes

Présentation des différentes solutions d'installation possibles, comment choisir ?
Installation de base les outils : kubectl, minikube, kubeadm Configuration de pods
et containers : assignation de mémoire, espace de stockage, processeurs,
affectation de pods à des noeuds... Configuration d'applications et exécution
Administration de Kubernetes

Utiliser les outils de supervision, analyse des logs, debugging Comment utiliser
kubectl exec pour accéder en shell à un container ? Analyser l'état des noeuds
avec Node Problem Detector Mise en oeuvre de StackDriver L'outil Rancher
Sécurité de Kubernetes

Présentation des différents points à sécuriser Accès à l'API Kubernetes
Limitations des ressources Contrôle des accès réseau Restrictions des accès à
etcd

Audit

<https://github.com/vchinnipilli/kubestriker>

Extra kubeadm

Extra tls cert manager

Intro build container image

Intro build container

Intro cluster setup

Intro run first app

Kops

Kops Project URL <https://github.com/kubernetes/kops>

Free DNS Service Sign up at <http://freedns.afraid.org/>

Choose for subdomain hosting

Enter the AWS nameservers given to you in route53 as nameservers for the subdomain

<http://www.dot.tk/> provides a free .tk domain name you can use and you can point it to the amazon AWS nameservers

Namecheap.com often has promotions for tld's like .co for just a couple of bucks

Cluster Commands `kops create cluster --name=kubernetes.newtech.academy --state=s3://kops-state-b429b --zones=eu-west-1a --node-count=2 --node-size=t2.micro --master-size=t2.micro --dns-zone=kubernetes.newtech.academy`

`kops update cluster kubernetes.newtech.academy --yes --state=s3://kops-state-b429b`

`kops delete cluster --name kubernetes.newtech.academy --state=s3://kops-state-b429b`

`kops delete cluster --name kubernetes.newtech.academy --state=s3://kops-state-b429b --yes`

Other

Kubernetes from scratch You can setup your cluster manually from scratch

If you're planning to deploy on AWS / Google / Azure, use the tools that are fit for these platforms

If you have an unsupported cloud platform, and you still want Kubernetes, you can install it manually

CoreOS + Kubernetes: <https://coreos.com/kubernetes/docs/latest/getting-started.html>

Docker You can download Docker Engine for:

Windows: <https://docs.docker.com/engine/installation/windows/>

MacOS: <https://docs.docker.com/engine/installation/mac/>

Linux: <https://docs.docker.com/engine/installation/linux/>

DevOps box Virtualbox: <http://www.virtualbox.org>

Vagrant: <http://www.vagrantup.com>

DevOps box: <https://github.com/wardviaene/devops-box>

Launch commands (in terminal / cmd / powershell):

```
cd devops-box/
```

```
vagrant up
```

Launch commands for a plain ubuntu box:

```
mkdir ubuntu
```

```
vagrant init ubuntu/xenial64
```

```
vagrant up
```

Cheatsheet: Docker commands Build image: `docker build .`

Build & Tag: `docker build -t wardviaene/k8s-demo:latest .`

Tag image: `docker tag imageid wardviaene/k8s-demo`

Push image: `docker push wardviaene/k8s-demo`

List images: `docker images`

List all containers: `docker ps -a`

Cheatsheet: Kubernetes commands `kubectl get pod`: Get information about all running pods

`kubectl describe pod` : Describe one pod

`kubectl expose pod --port=444 --name=frontend`: Expose the port of a pod (creates a new service)

`kubectl port-forward 8080`: Port forward the exposed pod port to your local machine

`kubectl attach -i`: Attach to the pod

`kubectl exec -- command`: Execute a command on the pod

`kubectl label pods mylabel=awesome`: Add a new label to a pod

`kubectl run -i --tty busybox --image=busybox --restart=Never -- sh`: Run a shell in a pod - very useful for debugging

`kubectl get deployments`: Get information on current deployments

`kubectl get rs`: Get information about the replica sets

`kubectl get pods --show-labels`: get pods, and also show labels attached to those pods

`kubectl rollout status deployment/helloworld-deployment`: Get deployment status

`kubectl set image deployment/helloworld-deployment k8s-demo=k8s-demo:2`: Run k8s-demo with the image label version 2

`kubectl edit deployment/helloworld-deployment`: Edit the deployment object

`kubectl rollout status deployment/helloworld-deployment`: Get the status of the rollout

`kubectl rollout history deployment/helloworld-deployment`: Get the rollout history

`kubectl rollout undo deployment/helloworld-deployment`: Rollback to previous version

`kubectl rollout undo deployment/helloworld-deployment --to-revision=n`: Rollback to any version version

AWS Commands `aws ec2 create-volume --size 10 --region us-east-1 --availability-zone us-east-1a --volume-type gp2`

Certificates Creating a new key for a new user: `openssl genrsa -out myuser.pem 2048`

Creating a certificate request: `openssl req -new -key myuser.pem -out myuser-csr.pem -subj "/CN=myuser/O=myteam/"`

Creating a certificate: `openssl x509 -req -in myuser-csr.pem -CA /path/to/kubernetes/ca.crt -CAkey /path/to/kubernetes/ca.key -CAcreateserial -out myuser.crt -days 10000`

Abbreviations used Resource type: Abbreviated alias

configmaps: cm

customresourcedefinition: crd

daemonsets: ds

deployments deploy

horizontalpodautoscalers: hpa

ingresses ing

limitranges limits

namespaces: ns

nodes: no

persistentvolumeclaims: pvc

persistentvolumes: pv

Pods: po

replicasets: rs

replicationcontrollers: rc

resourcequotas: quota

serviceaccounts: sa

services: svc

Overview

- <https://www.mcorbin.fr/posts/2021-12-04-kubernetes-pourquoi/>
- Kubernetes Overview Diagrams <https://brennarm.github.io/posts/kubernetes-overview-diagrams.html>

[Ask HN: Is it still worth learning Kubernetes in 2022? | Hacker News](#)

[Understanding Kubernetes in a visual way - 08 - Secrets | Kubernetes en français](#)

<https://vishnuch.tech/kubernetes-cheatsheet>

Pimp k9s

Pimp tubekit

Policies

<https://kyverno.io/>

<https://marcusnoble.co.uk/2022-01-20-restricting-cluster-admin-permissions/>

Questions

A faire

- Déployer un ingress
 - configurer un ingress dans minikube
 - configurer un ingress dans azure
- Utiliser helm pour installer wordpress
 - Possible dans minikube ?
- Mettre en place un service-mesh
- Utiliser devspace dans un vrai projet
- Mettre en place un service minikube et faire un tunnel

Open Questions

- Ou est la documentation pour les fichiers YAML de Kubernetes ?
- Comment obtenir la liste de toutes les ressources disponibles dans K8S ?
- Quelle différence entre service.type NodePort et ClusterIP ?
- Quelles sont les limitations du NodePort ?
- C'est quoi un service-mesh ? (ex: Istio, Linkerd) ?
- Quelle différence entre control plane et master / control
 - quand utiliser quel mot ?
- Comment utiliser/mettre en place des ingress ?
- Quels sont les différences entre les types de ingress ?
- Comment faire des selector ET ?
- Comment faire des selector OU ?
- Comment installer Ambassador ?
- Comment installer HAProxy Ingress Controller ?
- Comment installer Nginx Ingress Controller ?
- Comment installer Traefik Ingress Controller ?
- Comment définis-t-on des Storage Class (FIXME) ?

Questions [Resolved]

- C'est quoi Ambassador ?
 - Ambassador API Gateway is an Envoy based ingress controller with community or commercial support from Datawire.
 - Alternatives: Gloo, Voyager, HAProxy Ingress, Nginx Ingress Controller, Contour Ingress Traffic (Istio)
 - REF: <https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/>
- Est-ce que le nodeport fait du load-balancing ?
 - Oui, il balance (?) les connexions entre les pods

- REF: <https://www.ovh.com/blog/getting-external-traffic-into-kubernetes-clusterip-nodeport-loadbalancer-and-ingress/>
- Peut-t-on utiliser plusieurs ingress Controller ?
 - Oui. Il faudra cependant annoter chaque Ingress avec une `ingress.class` qui indique quel ingress controller va le gérer.
 - Exemple:

```
metadata:
  name: foo
  annotations:
    kubernetes.io/ingress.class: "nginx"
```
- REF: <https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md#multiple-ingress-controllers>

A (re)lire

- <https://learnk8s.io/troubleshooting-deployments>
- <https://medium.com/@hengfeng/if-you-cannot-get-external-ip-open-another-terminal-and-run-minikube-tunnel-4dd33385567b>

A faire

- Déployer un ingress
 - configurer un ingress dans minikube
 - configurer un ingress dans azure <https://www.dadall.info/article664/welcome-to-nginx-puis-wordpress>
- Utiliser helm pour installer wordpress
 - Possible dans minikube ?
- Mettre en place un service-mesh
- Utiliser devspace dans un vrai projet
- Mettre en place un service minikube et faire un tunnel
- Installer un storage avec Rook

Open Questions

- Comment définir une updateStrategy (FIXME: onDelete, RollingUpdate, etc.) ?
- Fonctionnement des Headless Services ?
- Comment fonctionnent les CSI (Cloud Storage Interface)
 - <https://softwareengineeringdaily.com/2019/01/11/why-is-storage-on-kubernetes-is-so-hard/>
- Comment fonctionnent les StatefulSet ?
- C'est quoi les feature gates ?
- En quoi les feature gates impactent apiserver, controller-manager, scheduler ?
 - <https://github.com/stefanprodan/podinfo/blob/6c8a85a5ab953874c7c83d50317359a0e5a352a9/docs/4-statefulsets.md>
- Comment faire un persistentVolume on-premises ?
- Comment peut-on voir les différents services qui fonctionnent sur chaque node ?
- Quelles sont les limites d'EmptyDir ?
 - https://www.alibabacloud.com/blog/kubernetes-volume-basics-emptydir-and-persistentvolume_594834
- Quelles sont les limites de hostPath ?
- Quelle différence entre control plane et master / control
 - quand utiliser quel mot ?
- Comment utiliser/mettre en place des ingress ?
- Quels sont les différences entre les types de ingress ?
- Comment faire des selector ET ?
- Comment faire des selector OU ?
- Comment installer Ambassador ?
- Comment installer HAProxy Ingress Controller ?
- Comment installer Nginx Ingress Controller ?
- Comment installer Traefix Ingress Controller ?

- Comment faire du Ingress sur de l'UDP ?
- Comment le Ingress ouvre son port vers l'extérieur ?
- Comment configurer un ExternalName ?
- Quelle différence entre un Ingress Controller et un API Gateway en tant que Kubernetes Service ?
- Comment déployer de nouveaux Kubernetes Services ?
- Comment utilise-t-on les StorageClass une fois définis ?
 - https://docs.okd.io/latest/install_config/storage_examples/gluster_dynamic_example.html
 - https://docs.openshift.com/container-platform/3.4/install_config/storage_examples/gluster_dynamic_example.html
- Pourquoi utiliser un Deployment plutôt qu'un ReplicationController ?
 - <https://medium.com/stakater/k8s-deployments-vs-statefulsets-vs-daemonsets-60582f0c62d4>

Questions [PENDING]

- C'est quoi un StatefulSet ?
 - <https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/>
 - <https://medium.com/stakater/k8s-deployments-vs-statefulsets-vs-daemonsets-60582f0c62d4>
- C'est quoi un DaemonSet ?
- Qu'est ce qu'un NetworkPolicy ?
- Comment fonctionnent les NetworkPolicy pour contrôler le trafic ?
 - <https://cloudnativelabs.github.io/post/2017-04-18-kubernetes-networking/>
- Quels outils pour sécuriser Kubernetes ?
 - <https://sysdig.com/blog/kubernetes-security-harden-kube-system/>
- Commence configurer docker pour utiliser un autre registry par défaut ?
 - `eval $(minikube docker-env)`

Questions [RESOLVED]

- Comment créer un helm chart ?
 - <https://medium.com/google-cloud/kubernetes-and-helm-create-your-own-helm-chart-5f54aed894c2>
- Comment définis-t-on des Storage Class (FIXME) ?
 - <https://kubernetes.io/docs/concepts/storage/persistent-volumes/>
 - <https://kubernetes.io/docs/concepts/storage/storage-classes/#glusterfs>
- Un pod peut-il parler à un autre pod sans service ? All pods can communicate with all other pods without NAT All nodes running pods can communicate with all pods (and vice-versa) without NAT IP that a pod sees itself as is the same IP that other pods see it as
- Peut-il y avoir plusieurs containers à l'interieur d'un Pod ?
- Comment faire un Pod multi-container
 - REF <https://www.bmc.com/blogs/kubernetes-pods/>
- Qu'est ce que les containers partagent au sein d'un Pod ?
 - Une IP unique, le réseau, le stockage, les autres spec assignées au Pod
- Comment est architecturé K8S, quels sont les services sur chaque node ?
 - kube-apiserver, etc, kube-controller-manager, kube-scheduler, kube-dns kubelet, kube-proxy,
 - REF: <https://sysdig.com/blog/kubernetes-security-harden-kube-system/>
- A quoi sert k8s apiserver ?
 - kube-apiserver : Le point de communication central du cluster. Fournit des terminaux REST pour interagir avec les autres entités du cluster et stocke l'état distribué dans le backend etcd.
- Quelles sont les limitations du NodePort ?
 - NodePort perce un trou dans la sécurité de votre cluster, car il ne peut pas être contrôlé par un NetworkPolicy
 - REF: <https://sysdig.com/blog/kubernetes-security-harden-kube-system/>

- Kubernetes NodePort ne peut pas exposer les ports standard à faible numérotation comme 80 et 443, ou même 8080 et 8443.
- A quoi sert k8s kubelet ?
 - L'agent de cluster qui s'exécute sur chaque nœud Kubernetes. Le kubelet lance les pods à l'aide du moteur de conteneur disponible (Docker, rkt, etc.) et vérifie périodiquement l'état des pods.
- A quoi sert k8s Kube-Controller-Manager ?
 - Surveille les pods nouvellement créés qui n'ont pas de nœud assigné, et sélectionne un nœud pour qu'ils fonctionnent dessus. A partir de la version 1.6, vous pouvez brancher votre propre programmeur Kubernetes personnalisé.
- Comment assigner un Pod à un node particulier ?
 - Il faut utiliser un `spec.nodeSelector` avec les propriétés du nœud visé dans la configuration du Pod
 - REF: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/?source=post_page-----46bd3ac6059d-----#nodeselector
- C'est quoi Ambassador ?
 - Ambassador API Gateway is an Envoy based ingress controller with community or commercial support from Datawire.
 - Alternatives: Gloo, Voyager, HAProxy Ingress, Nginx Ingress Controller, Control Ingress Traffic (Istio)
 - REF: <https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/>
- Est-ce que le nodeport fait du load-balancing ?
 - Oui, il balance (?) les connexions entre les pods
 - REF: <https://www.ovh.com/blog/getting-external-traffic-into-kubernetes-clusterip-nodeport-loadbalancer-and-ingress/>
- Peut-t-on utiliser plusieurs ingress Controller ?
 - Oui. Il faudra cependant annoter chaque Ingress avec une `ingress.class` qui indique quel ingress controller va le gérer.
 - Exemple: `metadata:`

```

name: foo

```

```
annotations:
```

```
kubernetes.io/ingress.class: "nginx"
```

- REF: <https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-guide/multiple-ingress.md#multiple-ingress-controllers>
- C'est quoi un service-mesh ? (ex: Istio, Linkerd) ?
 - REF: <http://www.reseaux-telecoms.net/actualites/lire-tout-savoir-sur-istio-et-le-service-mesh-kubernetes-27680.html>
 - REF: <https://platform9.com/blog/kubernetes-service-mesh-a-comparison-of-istio-linkerd-and-consul/>
 - REF: <https://blog.octo.com/vision-docto-sur-le-service-mesh-radiographique-du-service-mesh/>
 - REF: <https://www.zdnet.fr/actualites/au-dela-de-kubernetes-istio-le-service-reseau-en-mode-mesh-39877213.htm>
 - REF: <https://www.objectif-libre.com/fr/blog/2019/09/05/service-mesh-decouverte-et-mise-en-oeuvre/>
 - REF: <https://glasnostic.com/blog/kubernetes-service-mesh-what-is-istio>
 - REF: <https://dzone.com/articles/introduction-to-service-meshes-on-kubernetes-and-p>

A (re)lire

- <https://learnk8s.io/troubleshooting-deployments>
- <https://medium.com/@hengfeng/if-you-cannot-get-external-ip-open-another-terminal-and-run-minikube-tunnel-4dd33385567b>
- <https://blog.getambassador.io/kubernetes-ingress-nodeport-load-balancers-and-ingress-controllers-6e29f1c44f2d>
- <https://blog.jdxcode.com/posts/2019-12-06-pithy-guide-to-kubernetes-part-1/>
- <https://blog.jdxcode.com/posts/2019-12-07-pithy-guide-to-kubernetes-part-2-golang/>
- <https://blog.wescale.fr/2017/09/04/kubernetes-utiliser-traefik-comme-loadbalancer/>
- <https://schoolofdevops.github.io/ultimate-kubernetes-bootcamp/ingress/>

Kubernetes

NEW RESOURCES

<https://blog.zwindler.fr/2021/02/15/mettre-a-jour-le-ca-de-kubernetes-the-hard-way/> <https://blog.eleven-labs.com/fr/k9s/>

Examples of Kube Code

<https://github.com/joatmon08/kubernetes-reference>

?

<https://techbeacon.com/enterprise-it/47-advanced-tutorials-mastering-kubernetes>
<https://blog.zwindler.fr/2020/08/31/gerez-vos-secrets-kubernetes-dans-vault/>
<https://github.com/kodekloudhub/certified-kubernetes-administrator-course>
<https://unofficial-kubernetes.readthedocs.io/en/latest/concepts/storage/volumes/>

https://kubectl.docs.kubernetes.io/pages/app_management/apply.html <https://kube.news/>

Beginner guides & bootcamps

- <https://kubernetesbootcamp.github.io/kubernetes-bootcamp/>
- <https://www.katacoda.com/courses/kubernetes>
- <https://github.com/dennyzhang/challenges-kubernetes>
 - <https://kubernetes.dennyzhang.com/>
- <https://kubedex.com/kubernetes-courses/>
- <https://github.com/wx-chevalier/Awesome-Lists/blob/master/Infrastructure/Virtualization/Orchestration/Kubernetes/Kubernetes-List.md>
- <https://github.com/wx-chevalier/Awesome-Lists/blob/master/Infrastructure/Virtualization/Orchestration/Kubernetes/Kubernetes-OpenSource-List.md>
- <https://github.com/wx-chevalier/Awesome-Lists/blob/master/Infrastructure/Virtualization/Orchestration/Kubernetes/Kubernetes-Practices-List.md>
- <https://ston3o.me/auto-hebergement/installation-simple-cluster-kubernetes/#>

Books

- <https://zhiweiyin318.github.io/k8s-notes/> Container networking: from Docker to Kubernetes

Beginner courses

- <https://github.com/up1/course-kubernetes-in-practice>
- <https://kubernetes.io/docs/tutorials/hello-minikube/#create-a-minikube-cluster>
- <https://www.ibm.com/cloud/garage/content/course/kubernetes-101/0>
- <https://www.bogotobogo.com/DevOps/DevOps-Kubernetes-1-Running-Kubernetes-Locally-via-Minikube-Copied.php>
- <https://blog.alterway.fr/kubernetes-101-lancez-votre-premier-template-k8s.html>
- <https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16>
- <http://www.liksi.fr/2019/05/20/kubernetes-101/>
- <https://www.stavros.io/posts/kubernetes-101/>

Advanced courses

- <https://github.com/contino/kubernetes-201>
- <https://fr.slideshare.net/ChristianDick/kubernetes-201>

Focused Articles

- <https://zero-to-jupyterhub.readthedocs.io/en/latest/create-k8s-cluster.html>
- <https://zero-to-jupyterhub.readthedocs.io/en/latest/amazon/step-zero-aws.html>
- https://netapp-trident.readthedocs.io/en/stable-v19.01/dag/kubernetes/kubernetes_cluster_architecture_considerations.html

Cheat Sheet

- <https://cheatsheet.dennyzhang.com/cheatsheet-kubernetes-A4>
- <https://cheatsheet.dennyzhang.com/kubernetes-yaml-templates>
- <https://github.com/dennyzhang/cheatsheet-kubernetes-A4>

- <https://github.com/the-arcade/kubernetes-practice>
- <https://kubernetes.io/fr/docs/reference/kubectl/cheatsheet/>

Awesome

- <https://github.com/debianmaster/kubernetes-awesome>
- <https://github.com/zepouet/awesome-k8s>
- <https://github.com/ramitsurana/awesome-kubernetes/wiki/Awesome---Kubernetes>
- <https://github.com/ramitsurana/awesome-kubernetes>
- <https://github.com/tmrts/awesome-kubernetes>
- <https://github.com/jk8s/awesome-kubernetes>
- <https://github.com/coreos/awesome-kubernetes-extensions>
- <https://github.com/lukecav/awesome-kubernetes>
- <https://kgoralski.gitbook.io/wiki/kubernetes>

Free Courses

- <https://eu.udacity.com/course/scalable-microservices-with-kubernetes--ud615>
- <https://www.udemy.com/learn-devops-the-complete-kubernetes-course/> \$12.99 (acheté)

Unsorted Courses

- <https://www.edx.org/course/introduction-to-kubernetes>
- <https://digitaldefynd.com/best-kubernetes-tutorial-training-course/>
- <https://github.com/nareshganesan/kubernetes-practice>
- <https://training.linuxfoundation.org/training/kubernetes-fundamentals/> \$299
- <https://github.com/wardviaene/kubernetes-course>

Docker?

- <https://www.udemy.com/docker-and-kubernetes-the-complete-guide/> \$12.99

Workshops / Full tutorials

- <https://github.com/kubernetes/examples>
- <https://github.com/kelseyhightower/kubernetes-the-hard-way>
- <https://github.com/aws-samples/aws-workshop-for-kubernetes>
- <https://gist.github.com/janitham/d47c23cfa55331fe62a8c33f573cf013>
- <https://github.com/lalalawoo/Kubernetes-in-Practice>
- <https://github.com/saurjain108/Kubernetes-practice>
- <https://github.com/aokabin/kubernetes-practice/blob/master/kubernetes-clusters/>
- <https://github.com/shawnkoon/kubernetes-practice>

Interview questions

Video

- READ [Kubernetes et les containers pour votre application web : step by step](#)
- READ [Kubernetes 101](#)
- <https://www.youtube.com/watch?v=ULUQ8ZjLFeI>
- https://www.youtube.com/watch?v=ntuxW_sL4jI
- https://www.youtube.com/watch?v=Iscs6_Qz8Jw
- https://www.youtube.com/watch?v=Z_sNyT0hcVw
- <https://www.youtube.com/watch?v=uyiDNcSmwFw>
- <https://www.youtube.com/watch?v=UBkp48NXyW0>
- <https://www.udemy.com/helm-best-practices-2019/>
- <https://www.udemy.com/kubernetes-by-example/>
- <https://www.udemy.com/kubernetes-getting-started/>
- <https://www.udemy.com/just-enough-kubernetes/>
- <https://www.udemy.com/just-enough-istio-to-be-dangerous/>
- <https://www.udemy.com/containers-101/>

Outils tiers

- <https://draft.sh/>

Tags / Topics

- audit
-
- spring-boot
- istio
- flagger
- prometheus

namespaces operators <https://github.com/operator-framework/awesome-operators>
lessons-learned <https://github.com/cuongnv23/awesome-k8s-lessons-learned> tips
<https://github.com/mhausenblas/kubectl-in-action> cluster <https://github.com/topics/kubernetes-cluster> machine-learning <https://github.com/CogronicLabs/awesome-AI-kubernetes> best-practices <https://github.com/feiskyer/kubernetes-handbook/blob/master/en/SUMMARY.md> <https://github.com/ahmetb/kubernetes-network-policy-recipes> <https://github.com/Azure/k8s-best-practices> <https://github.com/freache/kubernetes-security-best-practice> <https://github.com/RyanAoh/awesome-k8s-practice> <https://github.com/zevarito/Kubernetes/blob/master/docs/user-guide/config-best-practices.md> embedded <https://github.com/rancher/k8s>
<https://github.com/rootsongjc/awesome-cloud-native>

Infrastructures / test

- <https://console.cloud.google.com/kubernetes/list?project=angular-theorem-245820>

Question

Pourquoi devoir switcher entre namespace et clusters ?

C'est quoi un plugin pour kubectk ?

Comment gérer les packages/plugin pour kubectl ?

<https://www.techrepublic.com/article/how-to-install-kubernetes-on-centos-8/>
https://www.youtube.com/watch?v=Qvj_ndNLPeE

https://www.youtube.com/channel/UCs_AZuYXi6NA9tkdbhjItHQ/playlists https://www.youtube.com/watch?v=brqAMyayjrI&list=PL34sAs7_26wMMJ_cAGTNhFs1m6U34V5ox <https://www.youtube.com/watch?>

v=deFfAUZpoxs&list=PL34sAs7_26wP009Cl03TZbtRFZ2DMJovl https://www.youtube.com/watch?v=Iscs6_Qz8Jw <https://www.youtube.com/watch?v=UXQ8LSVme1g&list=PLXOSxCLWFrNGlnu9KAn4ncFuUjdL1nWIT> https://www.youtube.com/watch?v=pFWVMbRGdXU&list=PLuZ_sYdawLiVpaW1w-oi97F_HnLj55dvh&index=4 <https://www.youtube.com/watch?v=NChhdOZV4sY> <https://kubernetes.io/docs/tasks/configure-pod-container/translate-compose-kubernetes/#install-kompose>

<https://github.com/kelseyhightower/kubernetes-the-hard-way/>

<https://www.mankier.com/1/kubeadm-init>

Pour les volumes: <https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#create-a-persistentvolume>

<https://learnk8s.io/troubleshooting-deployments>

Service mesh

- <https://itnext.io/kubernetes-istio-simply-visually-explained-58a7d158b83f>
- https://www.reddit.com/r/devops/comments/k8fubi/istio_service_mesh_a_beginners_guide/

Tips performance

```
sudo sysctl net/netfilter/nf_conntrack_max=131072
```

Tips

- TODO <https://enix.io/fr/blog/kubernetes-tip-and-tricks-la-commande-wait/>
- TODO <https://blog.zwindler.fr/2021/09/06/kubectl-tips-and-tricks-n4/>
- TODO <https://blog.zwindler.fr/2020/06/22/kubectl-tips-and-tricks-3/>
- TODO <https://blog.zwindler.fr/2020/03/23/supprimer-un-namespace-bloque-a-terminating/>
- TODO <https://blog.zwindler.fr/2020/01/20/kubectl-tips-and-tricks-n2/>
- TODO <https://blog.zwindler.fr/2019/10/30/kubectl-tips-tricks-1/>

Kubernetes tools

<https://github.com/juicedata/juicefs>

<https://sandstorm.github.io/sku/#/>

<https://k8slens.dev/>

<https://github.com/derailed/k9s>

TODO

- https://www.cncf.io/wp-content/uploads/2020/08/Getting-Started-with-Containers-and-Kubernetes_-March-2020-CNCF-Webinar.pdf

Debugging tips

PV / PVC is stuck !

<https://veducate.co.uk/kubernetes-pvc-terminating/>