OPENWHISK

I. Installer les modules nécessaires à OpenWhisk (Kubernetes et Helm)

D'après la documentation, pour un développement local, "léger", Apache recommande d'utiliser Kubernetes activé dans Docker.

Pour se faire la documentation explique que pour déployer OpenWhisk sur Kubernetes un diagramme Helm est utilisé. Ce diagramme déploie le cœur de la plateforme OpenWhisk et éventuellement certains de ses événements (à la fois à un nœud et à plusieurs nœuds Kubernetes).

Vocabulaire

Kubernetes est une plateforme d'orchestration de conteneurs qui automatise le déploiement, la mise à l'échelle et la gestion des applications conteneurisées.
Helm est un gestionnaire de paquets pour Kubernetes qui simplifie la gestion des applications Kubernetes. Installer docker

1. Créer un cluster Kubernetes

La première étape consiste à créer un cluster Kubernetes capable de prendre en charge un déploiement OpenWhisk.
La manière la plus simple d'obtenir un petit cluster Kubernetes adapté au développement et aux tests est d'utiliser l'une des approches Docker-in-Docker pour exécuter Kubernetes directement au-dessus de Docker de notre machine de développement.
Pour notre système d'exploitation hôte, Linux, nous utiliserons kind. En effet, nous pouvons exécuter Kubernetes au-dessus de Docker sur Linux en utilisant le projet kind.

Vocabulaire

Kind est basé sur l'utilisation de la virtualisation Docker-in-Docker (DIND) et de kubeadm. Il peut être utilisé pour créer un cluster Kubernetes virtuel multi-nœuds qui convient au déploiement d'OpenWhisk pour le développement et les tests. L'utilisation de kind n'est appropriée qu'à des fins de développement et de test. Elle n'est pas recommandée pour les déploiements de production d'OpenWhisk.

a. Docker

Installer Docker

Nous avons suivi la documentation fournie par Docker. Nous avons choisi d'installer docker à l'aide du référentiel apt.
Pour cela, avant d'installer Docker Engine pour la première fois sur notre nouvelle machine hôte, nous avons configuré le dépôt Docker.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

```
# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Puis nous avons installé les paquets Docker :

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Enfin, nous avons vérifié que l'installation de Docker Engine a réussie en exécutant l'image hello-world :

```
sudo docker run hello-world
```

Rajouter son utilisateur dans le groupe Docker

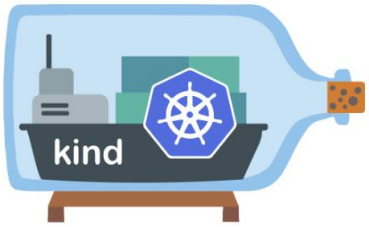
Comme le précisait la suite de la documentation Docker il est préférable d'ajouter son User au groupe Docker afin d'éviter par la suite du TP d'exécuter toutes les commandes en **root** ou en **sudo**.

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

Afin de vérifier que ces dernières commandes aient fonctionné nous avons rentré la commande suivante qui ne nous a pas répondu d'erreur d'autorisation.

```
docker ps -a
```

b. Kind



Docker installé, nous allons couvrir les opérations de base nécessaires à la création et au fonctionnement d'un cluster par défaut.
La documentation qui a été suivie est celle-ci.

Installer Kind

Nous avons téléchargé la dernière version v0.20.0 disponible lors de la rédaction de ce document en téléchargent les fichiers binaires (documentation).

```
[ $(uname -m) = aarch64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.20.0/kind-
linux-arm64
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind
```

Télécharger Kubectl

Avant de créer notre cluster, nous avons téléchargé kubectl pour interagir avec notre cluster en utilisant le fichier de configuration généré par kind.

```
curl -LO "https://dl.k8s.io/release/v1.27.3/bin/linux/arm64/kubectl"
curl -LO "https://dl.k8s.io/release/v1.27.3/bin/linux/arm64/kubectl.sha256"
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```

```
ubuntu@ip-10-0-0-111:~$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:33191
CoreDNS is running at https://127.0.0.1:33191/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Pour interagir avec un cluster spécifique, il suffira de spécifier le nom du cluster en tant que contexte dans kubectl

```
kubectl cluster-info --context kind-whisk-cluster
```

Configuration du cluster (kind-cluster.yaml)

Ce fichier définit la structure de votre cluster Kubernetes au sein de Kind, en spécifiant les nœuds et leurs rôles.

En dehors du cluster Kubernetes nous utiliserons la redirection de port configurée par les extraPortMappings pour permettre à la propriété apihost d'OpenWhisk d'être définie sur localhost:31001. Le script suppose donc que le port 31001 est disponible sur notre machine et qu'il peut être utilisé par OpenWhisk.

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
  extraPortMappings:
    - hostPort: 31001
      containerPort: 31001
- role: worker
```

Configurer OpenWhisk (mycluster.yaml)

Le fichier mycluster.yaml contiendra nos configurations spécifiques pour OpenWhisk lorsque nous utiliserons Helm pour l'installer dans notre cluster Kubernetes. Il contient les paramètres personnalisés nécessaires pour qu'OpenWhisk fonctionne correctement dans notre environnement. Nous utiliserons donc ce fichier plus tard.

```
mycluster.yaml
whisk:
  ingress:
    type: NodePort
    apiHostName: localhost
    apiHostPort: 31001
    useInternally: false

nginx:
  httpsNodePort: 31001

# disable affinity
affinity:
  enabled: false
toleration:
  enabled: false
invoker:
  options: "-Dwhisk.kubernetes.user-pod-node-affinity.enabled=false"
  # must use KCF as kind uses containerd as its container runtime
  containerFactory:
    impl: "kubernetes"
```

Créer le cluster (start-kind.yaml)

Le script `start-cluster.yaml` nous permet de lancer un cluster Kind. La commande présente dans le script permet de créer le cluster avec une option de configuration. Cette option permet de passer un fichier yaml définit dans `mycluster.yaml`.

```
#!/bin/bash

# Boot cluster
kind create cluster --config kind-cluster.yaml --name whisk-cluster --wait 10m || exit
1

echo "Kubernetes cluster is deployed and reachable"
kubectl describe nodes
```

```
ubuntu@ip-10-0-0-111:~$ ./start-kind.yaml
Creating cluster "whisk-cluster" ...
 ✓ Ensuring node image (kindest/node:v1.27.3)
 ✓ Preparing nodes
 ✓ Writing configuration
 ✓ Starting control-plane
 ✓ Installing CNI
 ✓ Installing StorageClass
 ✓ Waiting ≤ 10m0s for control-plane = Ready
 • Ready after 15s
Set kubectl context to "kind-whisk-cluster"
You can now use your cluster with:

kubectl cluster-info --context kind-whisk-cluster

Not sure what to do next?  Check out https://kind.sigs.k8s.io/docs/user/quick-start/
Kubernetes cluster is deployed and reachable
./start-kind.yaml: line 7: kubectl: command not found
```

Permet d'afficher les clusters :

```
ubuntu@ip-10-0-0-111:~$ kind get clusters
whisk-cluster
```

2. Helm

Helm est un outil qui simplifie le déploiement et la gestion des applications sur les clusters Kubernetes. La chart OpenWhisk Helm nécessite Helm 3.

a. Télécharger Helm

Nous avions la version v1.27.3 de Kubernetes. D'après la documentation Helm nous devions installer la version 3.12.x de Helm. Nous avons donc récupéré la version v3.12.3.

Helm Version	Supported Kubernetes Versions
3.13.x	1.28.x - 1.25.x
3.12.x	1.27.x - 1.24.x
3.11.x	1.26.x - 1.23.x
3.10.x	1.25.x - 1.22.x
3.9.x	1.24.x - 1.21.x

D'après la documentation :

```
wget https://get.helm.sh/helm-v3.12.3-linux-arm64.tar.gz
tar -zxvf helm-v3.12.3-linux-arm64.tar.gz
```

```
sudo mv linux-arm64/helm /usr/local/bin/helm
help helm
```

II. Déployer OpenWhisk

Maintenant que nous avons notre cluster Kubernetes et que nous avons installé le CLI Helm 3, nous pouvons déployer OpenWhisk.

1. Configuration initiale du cluster

Nous sommes dans une configuration multi-nœud. Nous pouvons le remarquer avec le résultat de la commande « `get nodes` » qui retourne 2 workers et un master :

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ kubectl get nodes
NAME                           STATUS   ROLES           AGE   VERSION
whisk-cluster-control-plane    Ready    control-plane   11m   v1.27.3
whisk-cluster-worker           Ready    <none>          11m   v1.27.3
whisk-cluster-worker2          Ready    <none>          11m   v1.27.3
```

Pour installer OpenWhisk sur un groupe de serveurs avec plusieurs nœuds, nous devons organiser ces nœuds pour que certaines tâches ne perturbent pas le bon fonctionnement d'OpenWhisk. Voici ce que nous devons faire :

```
kubectl label node whisk-cluster-worker openwhisk-role=invoker
kubectl label node whisk-cluster-worker2 openwhisk-role=invoker
```

2. Personnaliser le déploiement

Nous avions précédemment créé ce fichier (**`mycluster.yaml`**).

3. Déployer OpenWhisk avec Helm

Nous avons utilisé **`owdev`** comme nom de version et openwhisk comme espace de noms dans lequel les ressources de la charte seront déployées.

Le projet OpenWhisk maintient un dépôt Helm à l'adresse https://openwhisk.apache.org/charts.
Nous avons installé les versions officielles d'OpenWhisk à partir de ce dépôt :

```
helm repo add openwhisk https://openwhisk.apache.org/charts
helm repo update
helm install owdev openwhisk/openwhisk -n openwhisk --create-namespace -f kind-
cluster.yaml
```

Retour :

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ helm install owdev openwhisk/openwhisk -n openwhisk --create-namespace -f kind-cluster.yaml
NAME: owdev
LAST DEPLOYED: Wed Dec 20 08:37:53 2023
NAMESPACE: openwhisk
STATUS: deployed
REVISION: 1
NOTES:
Apache OpenWhisk
Copyright 2016-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

To configure your wsk cli to connect to it, set the apihost property
using the command below:

  $ wsk property set --apihost :31001

Your release is named owdev.

To learn more about the release, try:

  $ helm status owdev
  $ helm get owdev

Once the 'owdev-install-packages' Pod is in the Completed state, your OpenWhisk deployment is ready to be used.

Once the deployment is ready, you can verify it using:

  $ helm test owdev --cleanup
```

Vérifications :

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ helm list -A
NAME    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
owdev   openwhisk       1               2023-12-20 08:37:53.872159843 +0000 UTC deployed        openwhisk-1.0.0
```

4. Configurer le CLI wsk

a. Télécharger wsk CLI

Voici les commandes nécessaires :

```
wget https://github.com/apache/openwhisk-cli/releases/download/1.2.0/OpenWhisk_CLI-1.2.0-linux-arm64.tgz
tar -zxvf OpenWhisk_CLI-1.2.0-linux-arm64.tgz
sudo mv wsk /usr/local/bin/wsk
wsk --help
```

b. Configurer wsk CLI sur Linux

La suite de la documentation nous donne les commandes nécessaires :

```
wsk property set --apihost localhost:31001
wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP
```

Cette clef est pour s'authentifier en invité.

Résultats :

```
ubuntu@ip-10-0-0-111:~$ wsk property set --apihost localhost:31001
ok: whisk API host set to localhost:31001
ubuntu@ip-10-0-0-111:~$ wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccO
Fqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP
ok: whisk auth set. Run 'wsk property get --auth' to see the new value.
```

5. Tester

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ helm test owdev -n openwhisk
NAME: owdev
LAST DEPLOYED: Wed Dec 20 09:02:34 2023
NAMESPACE: openwhisk
STATUS: deployed
REVISION: 1
TEST SUITE:     owdev-tests-package-checker
Last Started:   Wed Dec 20 09:26:44 2023
Last Completed: Wed Dec 20 09:26:44 2023
Phase:          Failed
NOTES:
Apache OpenWhisk
Copyright 2016-2020 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

To configure your wsk cli to connect to it, set the apihost property
using the command below:

  $ wsk property set --apihost :31001

Your release is named owdev.

To learn more about the release, try:

  $ helm status owdev
  $ helm get owdev

Once the 'owdev-install-packages' Pod is in the Completed state, your OpenWhisk deployment is ready to be used.

Once the deployment is ready, you can verify it using:

  $ helm test owdev --cleanup
Error: warning: Hook test openwhisk/templates/tests/package-checker-pod.yaml failed: 1 error occurred:
        * object is being deleted: pods "owdev-tests-package-checker" already exists
```

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ kubectl get pods -n openwhisk
NAME                                READY   STATUS             RESTARTS       AGE
owdev-alarmprovider-7bc8fdcd4-5rqbm   0/1     Init:0/1           0              37m
owdev-apigateway-69ff54548-76glg      0/1     CrashLoopBackOff   12 (69s ago)   37m
owdev-controller-0                  0/1     Init:0/2           0              37m
owdev-couchdb-6f586c4757-6v84f      1/1     Running            0              37m
owdev-gen-certs-jfw8b               0/1     ImagePullBackOff   0              37m
owdev-init-couchdb-tdchc            0/1     ImagePullBackOff   0              37m
owdev-install-packages-6zb8z        0/1     Init:0/1           0              37m
owdev-invoker-0                     0/1     Init:0/1           0              37m
owdev-kafka-0                       0/1     Init:0/1           0              37m
owdev-kafkaprovider-76555797f8-dg45p   0/1     Init:0/1           0              37m
owdev-nginx-6df4684f69-nbb9r        0/1     Init:0/1           0              37m
owdev-redis-68cd47d75-zbm78         1/1     Running            0              37m
owdev-wskadmin                      0/1     ImagePullBackOff   0              37m
owdev-zookeeper-0                   0/1     ImagePullBackOff   0              37m
```

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ kubectl logs owdev-apigateway-69ff54548-76glg -n openwhisk
exec /usr/bin/dumb-init: exec format error
```

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ helm list -A
NAME    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
owdev   openwhisk       1               2023-12-20 09:02:34.084238265 +0000 UTC deployed        openwhisk-1.0.0
```

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ wsk list -v
REQUEST:
[GET]   https://localhost:31001/api/v1/namespaces/_/actions?limit=0&skip=0
Req Headers
{
  "Authorization": [
    "Basic MjNiYzQ2YjEtNzFmNi00ZWQ1LThjNTQtODE2YWE0ZjhjNTAyOjEyM3pPM3haQ0xyTU42djJCS0sxZFhZRnBYbFBrY2NPRnFtMTJDZEFzTWdSVTRWck5aOWx5R1ZDR3VNREdJd1A="
  ],
  "User-Agent": [
    "OpenWhisk-CLI/1.0 (2021-04-01T23:49:54.523+0000) linux arm64"
  ]
}
error: Unable to obtain the list of entities for namespace 'default': Get "https://localhost:31001/api/v1/namespaces/_/actions?limit=0&skip=0": EOF
```

```
ubuntu@ip-10-0-0-111:~/my_kind_files$ wsk property get
whisk API host          localhost:31001
whisk auth              23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP
whisk namespace         _
client cert
Client key
whisk API version       v1
whisk CLI version       2021-04-01T23:49:54.523+0000
whisk API build         Unknown
whisk API build number  Unknown
error: Unable to obtain API build information: Get "https://localhost:31001/api/v1": EOF
```