TP1 : Installation et Configuration Kubernetes

Cluster local avec Kind et premiers déploiements

IUT Grand Ouest Normandie – BUT Informatique S5

Année 2024/2025

Table des matières

1 Prérequis et environnement

1.1 Configuration système recommandée

Machine physique : Recommandé pour des performances optimales
Ressources minimales :
— CPU : 4 cores
— RAM : 8 GB
— Disque : 40 GB disponibles
— OS : Linux (Ubuntu 22.04+, Debian 12+, Parrot OS)

1.2 Configuration VM Proxmox (si nécessaire)

Type CPU : Changer de `x86-64-v2-AES` à `host`

```
# Sur l'hote Proxmox, verifier la virtualisation imbriquee
# Intel :
cat /sys/module/kvm_intel/parameters/nested

# AMD :
cat /sys/module/kvm_amd/parameters/nested

# Si retourne N ou 0, activer :
# Intel :
echo "options kvm_intel nested=1" > /etc/modprobe.d/kvm-intel.conf
modprobe -r kvm_intel
modprobe kvm_intel

# AMD :
echo "options kvm_amd nested=1" > /etc/modprobe.d/kvm-amd.conf
modprobe -r kvm_amd
modprobe kvm_amd
```

Dans la VM, verifier les flags CPU :

```
grep -E 'vmx|svm' /proc/cpuinfo
ls -la /dev/kvm
```

1.3 Vérification Docker

```
# Verifier Docker
docker --version
docker info | grep -E "Cgroup|Kernel"

# Doit afficher :
# Cgroup Driver: systemd
# Cgroup Version: 2
```

2 Installation de Kind

2.1 Installation sur Linux

```
# Telecharger et installer Kind
curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.30.0/kind-linux-amd64
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind

# Verifier l'installation
```

```
kind version
```

2.2 Installation de kubectl

```
# Telecharger kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

# Installer
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# Verifier
kubectl version --client
```

3 Création du cluster Kind

3.1 Cluster simple (single-node)

```
# Creer un cluster basique
kind create cluster --name mon-cluster

# Verifier le cluster
kubectl cluster-info --context kind-mon-cluster
kubectl get nodes
```

3.2 Cluster avec support Ingress

Créer le fichier `kind-ingress-config.yaml` :

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
```

```
# Creer le cluster avec configuration
kind create cluster --config kind-ingress-config.yaml --name cluster-ingress

# Verifier
kubectl get nodes
docker ps | grep kind
```

3.3 Commandes Kind essentielles

```
# Lister les clusters
kind get clusters

# Obtenir le kubeconfig
kind get kubeconfig --name cluster-ingress

# Charger une image Docker dans le cluster
docker pull nginx:latest
kind load docker-image nginx:latest --name cluster-ingress

# Supprimer un cluster
kind delete cluster --name cluster-ingress
```

4 Installation Ingress NGINX

4.1 Déploiement du contrôleur

```
# Installer ingress-nginx
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/
    main/deploy/static/provider/kind/deploy.yaml

# Attendre que le controleur soit pret
kubectl wait --namespace ingress-nginx \
  --for=condition=ready pod \
  --selector=app.kubernetes.io/component=controller \
  --timeout=90s

# Verifier
kubectl get pods -n ingress-nginx
kubectl get svc -n ingress-nginx
```

4.2 Troubleshooting Ingress

Problème : Webhook validation timeout

```
# Verifier l'etat
kubectl get pods -n ingress-nginx

# Si jobs admission en erreur, creer le secret manuellement
kubectl create secret generic ingress-nginx-admission \
  --from-literal=cert=dummy \
  --from-literal=key=dummy \
  -n ingress-nginx

# Supprimer le webhook (environnement de test)
kubectl delete validatingwebhookconfiguration ingress-nginx-admission

# Redemarrer le controleur
kubectl rollout restart deployment/ingress-nginx-controller -n ingress-nginx

# Attendre
kubectl wait --namespace ingress-nginx \
  --for=condition=ready pod \
  --selector=app.kubernetes.io/component=controller \
  --timeout=180s
```

5 Premier déploiement : nginx

5.1 Manifeste Deployment

Créer `nginx-deployment.yaml` :

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.27
        ports:
        - containerPort: 80
```

5.2 Manifeste Service

Créer `nginx-service.yaml` :

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```

5.3 Manifeste Ingress

Créer `nginx-ingress.yaml` :

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: nginx.local
    http:
```

```
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

5.4 Déploiement et test

```
# Appliquer les manifestes
kubectl apply -f nginx-deployment.yaml
kubectl apply -f nginx-service.yaml
kubectl apply -f nginx-ingress.yaml

# Verifier
kubectl get pods -l app=nginx
kubectl get svc nginx-service
kubectl get ingress nginx-ingress

# Ajouter l'entree DNS
echo "127.0.0.1 nginx.local" | sudo tee -a /etc/hosts

# Tester
curl http://nginx.local
```

6 Accès via IP LAN

6.1 Configuration réseau

```
# Trouver votre IP LAN
hostname -I | awk '{print $1}'

# Verifier que kind ecoute sur toutes les interfaces
sudo ss -tulpn | grep :80
# Devrait afficher : 0.0.0.0:80
```

6.2 Accès depuis d'autres machines

Sur les machines clientes (par exemple IP serveur : 192.168.56.11) :

```
# Ajouter l'entree DNS
echo "192.168.56.11 nginx.local" | sudo tee -a /etc/hosts

# Tester
curl http://nginx.local

# Ou avec header Host
curl -H "Host: nginx.local" http://192.168.56.11
```

7 Commandes kubectl essentielles

7.1 Gestion des ressources

```
# Lister les pods
kubectl get pods
kubectl get pods -o wide
kubectl get pods --all-namespaces

# Lister les deployments
kubectl get deployments

# Lister les services
kubectl get services

# Lister les ingress
kubectl get ingress

# Tout voir
kubectl get all
kubectl get all -A
```

7.2 Informations détaillées

```
# Describe (details complets)
kubectl describe pod <pod-name>
kubectl describe deployment <deployment-name>
kubectl describe service <service-name>

# Logs
kubectl logs <pod-name>
kubectl logs <pod-name> -f  # follow
kubectl logs deployment/<deployment-name>

# Events
kubectl get events --sort-by='.lastTimestamp'
```

7.3 Manipulation des ressources

```
# Appliquer un manifest
kubectl apply -f fichier.yaml
kubectl apply -f repertoire/

# Supprimer des ressources
kubectl delete -f fichier.yaml
kubectl delete pod <pod-name>
kubectl delete deployment <deployment-name>

# Modifier une ressource
kubectl edit deployment <deployment-name>

# Scaler un deployment
kubectl scale deployment <deployment-name> --replicas=3
```

7.4 Exec et port-forward

```
# Executer une commande dans un pod
kubectl exec <pod-name> -- ls /
kubectl exec -it <pod-name> -- /bin/bash

```

```
# Port-forward
kubectl port-forward pod/<pod-name> 8080:80
kubectl port-forward service/<service-name> 8080:80
```

8 Contextes et namespaces

8.1 Gestion des contextes

```
# Lister les contextes
kubectl config get-contexts

# Changer de contexte
kubectl config use-context kind-cluster-ingress

# Voir le contexte actuel
kubectl config current-context
```

8.2 Gestion des namespaces

```
# Lister les namespaces
kubectl get namespaces

# Creer un namespace
kubectl create namespace dev

# Definir un namespace par defaut
kubectl config set-context --current --namespace=dev

# Deployer dans un namespace specifique
kubectl apply -f fichier.yaml -n dev
kubectl get pods -n dev
```

9 Nettoyage

```
# Supprimer les ressources
kubectl delete -f nginx-ingress.yaml
kubectl delete -f nginx-service.yaml
kubectl delete -f nginx-deployment.yaml

# Supprimer le cluster
kind delete cluster --name cluster-ingress

# Retirer l'entree /etc/hosts
sudo sed -i '/nginx.local/d' /etc/hosts
```

10 Troubleshooting

10.1 Problèmes courants

ImagePullBackOff :

```
# Charger l'image dans Kind
docker pull nginx:1.27
kind load docker-image nginx:1.27 --name cluster-ingress
```

CrashLoopBackOff :

```
# Voir les logs
kubectl logs <pod-name>
kubectl logs <pod-name> --previous

# Voir les events
kubectl describe pod <pod-name>
```

Pods Pending :

```
# Verifier les ressources des nodes
kubectl describe nodes

# Verifier les events
kubectl get events --sort-by='.lastTimestamp'
```

Service ne répond pas :

```
# Verifier les endpoints
kubectl get endpoints <service-name>

# Verifier les labels
kubectl get pods --show-labels
kubectl describe service <service-name>
```

11 Cheat Sheet rapide

```
# CLUSTER
kind create cluster --name mon-cluster
kind delete cluster --name mon-cluster
kind get clusters

# DEPLOIEMENT
kubectl apply -f fichier.yaml
kubectl delete -f fichier.yaml

# VISUALISATION
kubectl get pods
kubectl get all
kubectl describe pod <nom>
kubectl logs <pod-name>

# MANIPULATION
kubectl scale deployment <nom> --replicas=3
kubectl exec -it <pod> -- /bin/bash
kubectl port-forward svc/<service> 8080:80

# DEBUG
kubectl get events --sort-by='.lastTimestamp'
kubectl logs <pod> --previous
kubectl describe pod <pod>

# CONTEXTE
kubectl config get-contexts
kubectl config use-context <context>
kubectl config set-context --current --namespace=<ns>
```

12 Ressources utiles

— Documentation Kind : `https://kind.sigs.k8s.io/`
— Documentation Kubernetes : `https://kubernetes.io/docs/`
— Ingress NGINX : `https://kubernetes.github.io/ingress-nginx/`
— kubectl Cheat Sheet : `https://kubernetes.io/docs/reference/kubectl/cheatsheet/`