

# TP 7 : Déploiement d'un cluster Kubernetes

Frédéric Fort - Université de Lille  
frederic.fort@univ-lille.fr

2025

## Introduction

Dans ce TP, vous allez continuer de vous familiariser avec Kubernetes en déployant un vrai cluster multi-machine. Vous utiliserez OpenTofu et Ansible pour le déploiement du cluster.

**Temps prévu pour ce TP :** 1 séance

Quelques liens utiles pour ce TP :

- La documentation officielle de Kubernetes
- La documentation OpenTofu du provider OpenStack
- La documentation d'installation de Docker

## Note pour le rendu

Pour faciliter la préparation et notation du rendu, il vous sera demandé d'utiliser un format unifié pour le rendu de ce TP. Créez un dossier `tp7` situé à la racine du dépôt de rendu. Les fichiers pour ce TP doivent tous se situer dans ce dossier.

Si vous avez rencontré des problèmes durant le TP, vous pouvez indiquer vos problèmes et présenter votre diagnostic ainsi que les opérations faites pour y remédier. Un processus de réflexion correct et approfondi sera noté favorablement.

## 1 Provisionnement avec OpenTofu

Dans un dossier `tofu`, créez un script `main.tf` permettant d'allouer 3 machines virtuelles : Une pour le plan de contrôle et deux nœuds. Les machines devront utiliser une image Debian. La VM plan de contrôle nécessitera au moins 2 CPU. Ajoutez des tags aux VM afin de marquer leur appartenance au cluster, ainsi que leur fonction au sein de celui-ci. Suivez les principes vu en cours et en TP.

## 2 Inventaire dynamique

Dans un dossier `ansible`, créez un fichier d’inventaire dynamique `openstack.yaml` afin de récupérer et grouper automatiquement les VM.

## 3 Installation de Kubernetes

Dans le dossier `ansible`, vous allez créer un playbook automatisant le déploiement d’un cluster Kubernetes sur vos machines. Le playbook devra bien entendu être idempotent.

Vous utiliserez les outils `kubeadm` et `kubectl` pour ce déploiement. Vous utiliserez la runtime `containerd` qui devra être installée depuis les dépôts de Docker pour des raisons de compatibilité. Les documentations disponibles ci-dessus pourront se révéler utiles.

Cet exercice est divisé en plusieurs étapes qui peuvent également être des plays individuels de votre playbook. Les plays sont en général à exécuter sur toutes les machines.

Dans le fichier `README.md` indiquez la commande à lancer afin d’exécuter le playbook.

### 3.1 Installation des paquets

Ajoutez les dépôts de Kubernetes et Docker afin d’installer Kubernetes et ses dépendances. Afin d’éviter les problèmes en cas de mis à jour, fixez également la version des paquets Kubernetes et `containerd` installés explicitement.

### 3.2 Configuration du système

Une fois Kubernetes installé, il est maintenant nécessaire d’effectuer la configuration de votre système.

- Les services `kubelet` et `containerd` doivent être (auto-)lancés ;
- Configurez les modules noyau à charger ;
- Configurez les paramètres noyau `sysctl` nécessaires ;
- Utilisez la commande `containerd config default` pour pré-générer un fichier de configuration et modifiez en particulier l’entrée `SystemdCgroup` afin qu’elle soit `true` ;
- Donnez les droits à l’utilisateur `debian` d’utiliser `kubectl`.

### 3.3 Lancement du cluster

Avec les configurations faites, le plan de contrôle peut lancer le cluster. Utilisez la commande `kubeadm` appropriée. Si vous utilisez *flannel*, pensez à spécifier `--pod-network-cidr=10.244.0.0/16`. Lancez votre plugin réseau ensuite.

### 3.4 Ajout des nœuds

Utilisez `kubeadm` pour générer une commande `join` depuis le plan de contrôle. Transmettez cette commande aux nœuds afin que ceux-ci rejoignent le cluster.

## 4 Validation

Afin de valider le bon déploiement du cluster, déployez un `ReplicaSet` avec plus de conteneurs que de nœuds. Dans le fichier `README.md` indiquez la commande et les fichiers nécessaires pour lancer les conteneurs, ainsi que la procédure pour valider que les conteneurs s'exécutent bien sur plusieurs machines.