# *Provisionnement des VM*

**Important**: uniquement pour architecture x86_64

Il existe différentes solutions pour installer k8s. Celle que je vous propose ici est de partir de zéro, puis d'installer tous les composants permettant d'avoir un cluster constitué d'1 maître et de 2 nœuds. Pour se concentrer sur l'essentielle, je vous fournis des scripts qui automatisent l'installation.

## 1. Création d'un support USB bootable de Linux.

Le choix des versions de logiciel est très important. L'ensemble des scripts que je vous donne fonctionne uniquement avec kubuntu 24.04.1 LTS. Les principaux problèmes qu'on rencontre actuellement sont liés à:

- dépréciation de iptable, utilise nft_table
- changement du mode de boot pour ubuntu 24.04 et rocky linux 9.4
- secure boot

Si vous utilisez le disque de l'école, AVEC UN PC DE L'ÉCOLE, il est nécessaire de modifier le fichier

/etc/default/grub

Et d'ajouter la variable

GRUB_DISABLE_OS_PROBER=false

Pour que soit pris en compte un OS présent sur un disque externe. La mise à jour du grub et de l'UEFI se fait via la commande:

sudo update-grub

Sous windows , je n'ai pas testé, mais voir le lien:

https://answers.microsoft.com/en-us/windows/forum/all/how-do-i-add-a-boot-entry-for-a-partition/e5a3876e-f91b-46f5-99fb-821f2a7a2fe7

Pour éviter de perdre de nombreuses heures, il est important de suivre les règles suivantes :

- Si vous avez moins de 8 giga de ram sur votre PC, alors il est nécessaire d'emprunter un PC de l'école.
- Si vous avez 16G de RAM, vous devez boottez sur un disque SSD externe que je vous fournis.
  - En cas de probléme, désactiver le « secure boot » du BIOS.
  - Si vous travailler avec WINDOWS, Rechercher votre clé de récupération BitLocker avant TOUTE MANIPULATION.
- Si vous avez 32G de RAM, créer une VM de 24 Giga à partir de l'iso kubuntu 24.04.1 LTS avec un compte « etudiant » et mot de passe « etudiant ». Vous utilisez l'hyperviseur de votre choix.

Pour visionner correctement les pptx avec libre office, il est nécessaire d'installer les fontes présentes sous Windows (https://www.libreofficehelp.com/install-fonts-libreoffice-openoffice/ ou d'utiliser des fontes de substitutions (https://doc.ubuntu-fr.org/installer_de_nouvelles_polices_de_caractere#installation_pour_un_seul_utilisateur).

Vous trouverez le fichier «Fonts.zip» sur le site « partage.imt.fr » qui vous aidera dans cette tache.

## 2. Configuration de l'hôte.

Le cluster nécessite la création de 4 VM. Nous pourrions manuellement créer ces VM, mais cela est long avec une forte probabilité de ne pas être configurées de la même manière. Dans notre cas, les outils KVM, Vagrant et Ansible sont une réponse à notre besoin et nous allons les utiliser. D'autre existe, que vous verrez au cours de vos études.
Installons ces 3 outils.

1. Installation Kvm (voir https://www.linuxtechi.com/how-to-install-kvm-on-ubuntu-22-04/)

Il faut déjà vérifier que votre ordinateur supporte la virtualisation. La commande

```
$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
```

doit vous afficher un résultat. Si ce n'est pas le cas, allez voir dans le BIOS la configuration de votre ordinateur pour activer la virtualisation.

Installer le package de kvm

```
$ sudo apt install -y qemu-kvm virt-manager libvirt-daemon-system virtinst libvirt-clients bridge-utils
```

Il faut ensuite démarrer le daemon gérant la virtualisation

```
$ sudo systemctl enable --now libvirtd
$ sudo systemctl start libvirtd
```

Vérifier que le daemon gérant la virtualisation fonctionne bien

```
$ sudo systemctl status libvirtd
```

```
● libvirtd.service - Virtualization daemon
     Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-11-30 20:56:18 CET; 2min 38s ago
TriggeredBy: ● libvirtd-admin.socket
             ● libvirtd.socket
             ● libvirtd-ro.socket
       Docs: man:libvirtd(8)
             https://libvirt.org
   Main PID: 5203 (libvirtd)
      Tasks: 21 (limit: 32768)
     Memory: 10.3M
        CPU: 608ms
     CGroup: /system.slice/libvirtd.service
             ├─5203 /usr/sbin/libvirtd
             ├─5326 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --lea
             └─5327 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --lea
```

Pour créer des VM, l'utilisateur doit appartenir au groupe kvm et libvirt. Et libvirt doit accéder au répertoire etudiant.

```
$ sudo usermod -aG kvm $USER
$ sudo usermod -aG libvirt $USER
$ sudo usermod -aG $USER libvirt-qemu
```

Dans le fichier $HOME/.bashrc , rajoutez
export VIRSH_DEFAULT_CONNECT_URI=qemu:///system

export LIBVIRT_DEFAULT_URI=qemu:///system

Pour que l'appartenance aux groupes soit prise en compte, rebootez le PC..

Vérifiez que vous appartenez au groupe kvm et libvirt

```
etudiant@k8s:~$ id
uid=1000(etudiant) gid=1000(etudiant) groups=1000(etudiant),4(adm),24(cdrom),27(sudo),30(dip),46(
plugdev),114(lpadmin),125(libvirt),988(sambashare),993(kvm)
etudiant@k8s:~$
```

Si vous lancer virt-manager, vous serez alors capable de créer des VM. Faites un essai.

## 2. KVM

Avant d'aller plus loin, nous allons regarder et exécuter le code d'une VM basique pour comprendre le fonctionnement de la virtualisation matérielle.
Cloner le répertoire en local

```
$ git clone https://github.com/soulxu/kvmsample.git
```

Installer make et gcc, puis compiler et exécuter « kvmsample». Pouvez-vous m'expliquer le rôle de la fonction « kvm_cpu_thread »

## 3. Qemu

Pour voir la différence de performance entre une virtualisation complète et émulée, nous allons tester le lancement de Rocky Linux dans ces 2 modes sur x86 et ppc.
Chargez les images suivantes :

https://dl.rockylinux.org/vault/rocky/9.3/images/ppc64le/Rocky-9-GenericCloud.latest.ppc64le.qcow2

https://dl.rockylinux.org/vault/rocky/9.3/images/x86_64/Rocky-9-GenericCloud.latest.x86_64.qcow2

Pour voir les informations de boot sur le terminal QEMU, choisissez la fenêtre « serie » :
view → show Tabs →serial0

Pour émuler un ppc :

```
$ qemu-system-ppc64 -m 4096 -hda Rocky-9-GenericCloud.latest.ppc64le.qcow2
```

Pour émuler x86:

```
$ qemu-system-x86_64 -m 4096 -hda Rocky-9-GenericCloud.latest.x86_64.qcow2  -cpu SandyBridge-v2
```

Et pour une virtualisation complète

```
$ qemu-system-x86_64 -m 4096 -hda Rocky-9-GenericCloud.latest.x86_64.qcow2 -enable-kvm -cpu host
```

Vous pouvez constater la différence de vitesse ….

## 4. Création d'un réseau KVM

Avant d'aller plus loin, cloner le projet en local qui contient les fichiers utiles pour la suite du projet:

https://github.com/vjogit/tp-k8s.git

L'ensemble des VM appartienne au réseau 10.54.56/24. Ce réseau va être géré par un routeur virtuel de l'hote. Voici les instructions pour le créer grâce à l'utilitaire virsh

```
$ virsh net-define bridge-kvm.xml
$ virsh net-autostart bridge-kvm
$ virsh net-start bridge-kvm
```

Vérifier que tout fonctionne.

```
$ virsh net-list --all
```

```
etudiant@cours:/mnt/cours/k8s/cours-k8s/vm$ virsh net-list --all
 Name         State    Autostart   Persistent
--------------------------------------------
 bridge-kvm   active   yes         yes
 default      active   yes         yes

etudiant@cours:/mnt/cours/k8s/cours-k8s/vm$
```

## 5. Installation de Vagrant

Sur la page :

https://developer.hashicorp.com/vagrant/install

Téléchargez le fichier « vagrant_2.X.Y_linux_amd64.zip » et extrayez l'exécutable.
Vérifier que vous avez bien:

```
etudiant@k8s:~$ ./vagrant  --version
Vagrant 2.4.1
etudiant@k8s:~$
```

Pour utiliser libvirt avec Vagrant, il est nécessaire d'installer le plugin adéquat :

```
$ sudo apt install libvirt-dev
$ ~/vagrant plugin install vagrant-libvirt
```

## 6. Installation de Ansible

Ansible est écrit en Python, et s'installe via pipx.

```
$ sudo apt install pipx
```

Puis

```
$ pipx install --include-deps ansible
$ pipx ensurepath
```

Fermer la fenêtre shell et ré-ouvrer une nouvelle. Vous devez avoir

```
etudiant@k8s:~$ ansible --version
ansible [core 2.17.4]
  config file = None
  configured module search path = ['/home/etudiant/.ansible/plugins/modules', '/usr/share/ansible/plug
ns/modules']
  ansible python module location = /home/etudiant/.local/share/pipx/venvs/ansible/lib/python3.12/site-
ackages/ansible
  ansible collection location = /home/etudiant/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/etudiant/.local/bin/ansible
  python version = 3.12.3 (main, Jul 31 2024, 17:43:48) [GCC 13.2.0] (/home/etudiant/.local/share/pipx
venvs/ansible/bin/python)
  jinja version = 3.1.4
  libyaml = True
etudiant@k8s:~$
```

Puis ajouter le package python :

**$ sudo apt install python3-lxml**

3. Clef SSH (voir http://www.linuxproblem.org/art_9.html)

Les VM Vagrant possèdent par défaut un compte (log:«vagrant», pwd: «vagrant») et une clef ssh permettant de se connecter sans mot de passe. Par facilité, nous en utiliserons une spécifique au compte «etudiant».
Dans un premier temps, créons la clef publique et la clef privée dans le répertoire ~/.ssh

**$ ssh-keygen -t rsa**

Plus tard, nous utiliserons le fichier id_rsa.pub dans les scripts de création des VM.

4. Création des VM avec Vagrant (voir https://blog.stephane-robert.info/docs/infra-as-code/provisionnement/vagrant/introduction/)

Aller dans le répertoire cours-k8s/vm.vagrant et exécuter la commande:

**$ ~/vagrant up**

Vagrant exécute les commandes présentent dans le fichier Vagranfile:

- recherche l'image de Rocky 9 pour vagrant et libvirt en local, ou sur le site download.rockylinux.org si pas présente en local
- vérifie la checksum
- 'commande' à kvm la création d'une VM avec une adresse ip

De plus, kvm crée si nécessaire le réseau «bridge-kvm» d'ip 10.54.56/24 sur lequel sera connecté toutes les VM du cluster.

Faite un essai pour vérifier que les Vm sont bien configurées

**$ vagrant ssh master-vagrant**
**$ ping google.fr**

Dans le cas classique,chaque Vm possède 2 cartes:
eth0 :192.168.121.XXX utilisé par vagrant
eth1 : 10.54.56.100 pour k8s

Avec cette configuration, k8s va utiliser la première carte qu'il trouve, c'est-à-dire le réseau en 192. Ceci pose problème, car nous voulons un réseau spécifique k8s en 10.54.56.0/24.

Il est possible de contourner ce problème (voir le fichier Vagrant.ok), mais le plus simple est d'utiliser un autre outil.

Détruisez les VM qui ne nous seront plus utile

```
$ vagrant destroy
```

## 5. Création des VM avec Ansible

Ansible est plus complet que Vagrant, dans le sens ou nous pouvons provisionner des VM et les configurer (comme Puppet) de manière plus flexible. Pour nous faciliter la tâche, je vais prendre cette fois-ci l'image contenant l'outil cloud-init pour configurer :

- l'ip de la machine et une plage d'ip de 10.54.56.33 à 10.54.56.53 pour le master
- un utilisateur (logging: «etudiant», pwd: «etudiant»)
- une clef ssh.

Je vous ai écrit les scripts de base pour la création, l'arrêt et la suppression d'une VM.

Allez dans le répertoire vm.ansible, créer le réseau bridge-kvm s'il n'existe pas comme en 2.2 et exécuter la commande

```
$ virsh net-define bridge-kvm.xml
$ virsh net-autostart bridge-kvm
$ virsh net-start bridge-kvm
$ ansible-playbook creation.yaml
```

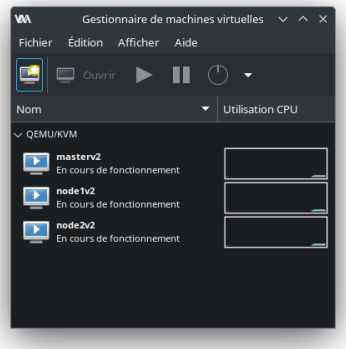
Pour information, les commandes détruisant une interface réseau sont

```
$ virsh net-destroy bridge-kvm
$ virsh  net-undefine bridge-kvm
```

Vérifier que les machines sont bien configurées

```
$ ssh root@10.54.56.100
$ ping google.fr
```

La commande « virsh-manager » doit vous donner la sortie suivante :




Voilà, cette première partie est terminée.