

Access aux machines

Liens pratiques:

<https://ent.univ-rennes1.fr/f/intranet/p/rssEtuEsirlstic.u27l1n88/max/render.uP?pCp>

Portal VPN:

<https://istic-vpn.univ-rennes1.fr/sslvpn/portal.html#!/>

Depuis le réseau local de l'ISTIC

- via le navigateur()
 -
- En ligne de commande
ssh zprojet@[vm].istic.univ-rennes1.fr
- VMS:
 - ares1, ares2,, ares20

Compte utilisateur:

```
User : zprojet
Passwd: Zistic*!1
```

Passer en mode administrateur :

```
$ sudo su
```

Entrer votre mot de passe

En tant que administrateur, vous pouvez changer le mot passe :

```
# passwd
```

TP 2 : Virtualisation des réseaux avec KVM

Objectif :

- Créer un environnement virtualisé avec KVM et libvirt pour simuler des réseaux d'entreprise en utilisant des images Ubuntu Cloud et des outils comme `virsh` pour la gestion des VM et réseaux virtuels.

Durée :

- 1h30
-

1. Prérequis

- Avoir la virtualisation activée dans le BIOS (Intel VT-x ou AMD-V).
- Disposer d'une machine Linux (Ubuntu) avec les outils suivants installés :
 - `kvm` (Kernel-based Virtual Machine)
 - `libvirt` pour la gestion des machines virtuelles
 - `virsh` pour la gestion en ligne de commande
 - Une image Ubuntu Cloud pour déployer les VM.

Installation des outils :

1. Installer KVM et les outils nécessaires :

```
sudo apt update
sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients libguestfs-
tools bridge-utils virt-manager cloud-utils
```

2. Vérifier que KVM est bien installé :

```
sudo kvm-ok
```

La commande devrait retourner que l'hôte supporte la virtualisation.

3. Démarrer le service `libvirtd` :

```
sudo systemctl start libvirtd
sudo systemctl enable libvirtd
sudo systemctl status libvirtd
```

4. Télécharger l'image de base Ubuntu Cloud :

```
wget https://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-
amd64.img
```

4.1 Configurer l'image base

```
virt-custimize -a current/focal-server-cloudimg-amd64.img --root-password
password:<votre_mot_de_passe>
```

Attention! si la commande `virt-customize` n'exite pas, installer le packet `libguestfs-tools`

2. Création de machines virtuelles avec une image Ubuntu Cloud

Étape 1 : Préparation d'une image personnalisée avec un cloud-init

1. générer une clé ssh avec `ssh-keygen`
2. Créer un fichier de configuration cloud-init pour automatiser l'installation des VM :

```
cat > user-data <<EOF
#cloud-config
users:
  - name: student
    ssh-authorized-keys:
      - <votre_cle_ssh_publique>
    sudo: ['ALL=(ALL) NOPASSWD:ALL']
    groups: sudo
    shell: /bin/bash
EOF
```

3. Créer une image ISO de configuration cloud-init :

```
cloud-localds cloud-init.iso user-data
```

Étape 2 : Créer une machine virtuelle avec `virsh`

1. Utiliser `virt-install` pour créer une machine virtuelle avec l'image Ubuntu Cloud :

```
sudo virt-install --name=vm1 --vcpus=2 --memory=1024 \
--disk path=/var/lib/libvirt/images/vm1.qcow2,size=2 \
--disk path=cloud-init.iso,device=cdrom \
--import --os-variant=ubuntu20.04 --network network=default \
--graphics none --console pty,target_type=serial
```

Attention remplacer `/var/lib/libvirt/images/vm1.qcow2` par votre image customisée

2. La machine virtuelle démarrera automatiquement et utilisera l'image cloud pour se configurer.
3. Pour sortir de la console

```
ctrl+shift+5
```

4. Vérifier que la machine virtuelle fonctionne:

```
virsh list
```

5. Accéder à la console de la machine virtuelle :

```
virsh console <id_machine/nom_machine>
```

mettre root et le mot de passe root que vous avez défini initialement

6. Vérifier la configuration réseau de la machine (IP, MAC, GATEWAY) à l'aide de la commande `ip`
7. Connectez vous à la machine virtuelle avec `ssh`

```
ssh ./ssh/id_rsa student@<ip_vm>
```

Attention! vérifier que l'utilisateur courant a les droits de lecture sur le fichier de la cle privée.
Sinon donner les droits avec la commande `chown`

Étape 3 : Créer plusieurs VM (dans la limite des ressources) en clonant l'image

1. Une fois la première VM prête, vous pouvez la cloner pour obtenir plusieurs instances :

```
sudo virt-clone --original vm1 --name vm2 --file  
/var/lib/libvirt/images/vm2.qcow2
```

3. Configuration de réseaux internes avec virsh

Étape 4 : Créer des réseaux virtuels internes

1. Créer un réseau virtuel avec un fichier XML pour définir le réseau interne :

```
sudo nano internal-network.xml
```

Exemple de configuration :

```
<network>  
  <name>internal</name>  
  <bridge name='virbr1' />  
  <ip address='192.168.50.1' netmask='255.255.255.0'>  
    <dhcp>  
      <range start='192.168.50.100' end='192.168.50.200' />  
    </dhcp>  
  </ip>  
</network>
```

2. Charger et démarrer le réseau :

```
sudo virsh net-define internal-network.xml  
sudo virsh net-start internal  
sudo virsh net-autostart internal
```

3. Attacher les machines virtuelles à ce réseau via `virsh` ou `virt-manager`.

`sudo virsh net-list --all` Pour attacher une machine virtuelle à un réseau spécifique avec `virsh`, vous devez suivre les étapes suivantes :

3.1 Lister les réseaux disponibles Utilisez la commande suivante pour lister tous les réseaux virtuels disponibles sur l'hôte KVM :

```
sudo virsh net-list --all
```

Cela affichera tous les réseaux définis, y compris ceux qui sont actifs et inactifs.

Exemple de sortie :

Name	State	Autostart	Persistent

default	active	yes	yes
internal	active	yes	yes

3.2 Attacher une machine virtuelle à un réseau avec `virsh`

- a) : Arrêter la machine virtuelle Avant de modifier la configuration réseau de la machine virtuelle, vous devez l'arrêter si elle est en cours d'exécution :

```
sudo virsh shutdown <nom_de_la_vm>
```

Vérifiez que la machine est bien arrêtée :

```
sudo virsh list --all
```

- b) : Détacher l'interface réseau actuelle (optionnel) Si vous devez d'abord détacher une interface réseau existante, utilisez la commande suivante :

```
sudo virsh detach-interface <nom_de_la_vm> network --current
```

- c) : Attacher une nouvelle interface réseau Attachez l'interface réseau de la machine virtuelle au réseau de votre choix, par exemple `internal` :

```
sudo virsh attach-interface --domain <nom_de_la_vm> --type network --source internal --model virtio --config
```

- `--domain <nom_de_la_vm>` : le nom de la machine virtuelle.
- `--type network` : spécifie que l'interface sera attachée à un réseau défini par libvirt.
- `--source internal` : spécifie le nom du réseau auquel la VM sera connectée (ex. `internal`).
- `--model virtio` : spécifie le modèle de l'interface réseau (généralement `virtio` pour les meilleures performances).
- `--config` : pour rendre la modification permanente (même après le redémarrage de la VM).
- d) : Démarrer la machine virtuelle Une fois l'interface attachée, redémarrez la machine virtuelle :

```
sudo virsh start <nom_de_la_vm>
```

3.2 Vérification Pour vérifier que l'interface est bien attachée au réseau souhaité, utilisez la commande suivante :

```
sudo virsh domiflist <nom_de_la_vm>
```

Cela affichera les interfaces réseau de la VM et leurs sources.

Exemple de sortie :

Interface	Type	Source	Model	MAC
vnet0	network	internal	virtio	52:54:00:4d:57:bc

Étape 5 : Configurer un routeur virtuel

1. Configurer une des VM comme routeur virtuel :

- Activer le routage IP sur cette machine :

```
sudo sysctl -w net.ipv4.ip_forward=1
```

2. Configurer le NAT sur le routeur en utilisant `iptables` :

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

3. Sur les autres machines virtuelles, configurer le routeur comme passerelle par défaut :

```
sudo ip route add default via 192.168.50.1
```

4. Test et validation

Étape 6 : Tester la connectivité

1. Depuis une VM, tester la connectivité avec d'autres machines virtuelles du même réseau :

```
ping 192.168.50.x
```

2. Tester l'accès à Internet depuis une machine derrière le routeur virtuel.

5. Conclusion et rendu

- Documentez la création des machines virtuelles et des réseaux virtuels.
- Décrivez les tests réalisés pour valider la connectivité des sous-réseaux.
- Proposez des pistes d'amélioration ou d'ajouts, comme la segmentation de réseaux ou l'ajout de services réseau (DNS, DHCP).