TP1 – Partie 3

III. Gestion des conteneurs

Pour les scripts suivants, vous allez utiliser des conteneurs LXC. Dans cette partie, vous allez écrire des scripts bash qui permettront la mise en place de différents conteneurs. Vous avez le choix du type de conteneur que vous allez utiliser : Debian, **Ubuntu**, Alpine.

Les conteneurs seront montés selon une architecture NAT sur l'interface réseau de votre machine.

I. Table des matières

1. Scripts de création / destruction de l'environnement

Écrivez les 4 scripts qui assurent la mise en place l'environnement d'exécution des conteneurs :

a. inst_cont_env.sh : installera l'ensemble des packages nécessaires à l'utilisation de conteneurs et à la création de bridge.

Dans un premier temps j'ai mis en place la bridge sur la machine Ubuntu :

```
#! /bin/bash
apt-get update
if [ $( dpkg --list | grep -c lxc-templates) -eq 0 ]
then
        echo "Intallation du paquet LXC nécessaire."
        apt install lxc lxc-templates -y
else
        #Verification de l'installation
        if [ $(lxc-checkconfig | grep -c disabled) -eq 0 ]
        then

        else
                echo "Problème de configurations sur LXC. Exit erreur 1."
                exit 1
        fi
fi
exit 0
```

b. set_net.sh : assurera la création du bridge permettant la connexion des futurs conteneurs, vous êtes libre du type de bridge que vous allez utiliser

```
#! /bin/bash

#Mise en place du bridge
if [ $# -ne 2 ]
then
    echo "Il manque des arguments. Exit erreur 1. "
    exit 1
else
    ip_bridge=$2
    nom_bridge=$3
    if [ $(cat /etc/netplan/00-installer-config.yaml | grep -c bridges) -eq 0 ]
    then
        #On suppose que le fichier n'a jamais été modifié donc on le sauvegarde avant
modifications
        cp /etc/netplan/00-installer-config.yaml /etc/netplan/00-installer-config.yaml.copy
        echo "  bridges:" >> /etc/netplan/00-installer-config.yaml
    fi
    if [ $(cat /etc/netplan/00-installer-config.yaml | grep -c $nom_bridge) -eq 0 ]
    then
        echo "   $nom_bridge:" >> /etc/netplan/00-installer-config.yaml
        echo "      dhcp4: false" >> /etc/netplan/00-installer-config.yaml
        echo "      addresses: [$ip_bridge]" >> /etc/netplan/00-installer-config.yaml
        echo "      nameservers:" >> /etc/netplan/00-installer-config.yaml
        echo "        addresses: [8.8.8.8] " >> /etc/netplan/00-installer-config.yaml

        netplan apply
    else
        echo "Un bridge porte déjà le nom '$nom_bridge'. Impossible de créer le bridge. Exit
erreur 2."
        exit 2
    fi
fi
```

```
exit 0
```

c. restore_net.sh : assurera la destruction du bridge créé par le script set_net.sh, ainsi que la remise en place de la configuration d'origine.

```
#! /bin/bash
#on récupère le nom du bridge a effacer
nom_bridge=$(brctl show | awk '{if($1 != "bridge") print $1;}')

#On supprime le bridge
ip link set $nom_bridge down
brctl delbr $nom_bridge

#On met a jour le fichier netplan grâce à notre copie
cp /etc/netplan/00-installer-config.yaml.copy /etc/netplan/00-installer-
config.yaml
netplan apply

if [ $(ip a | grep -c $nom_bridge) -eq 0 ]
then
    echo "Le bridge $nom_bridge a été supprimé. "
else
    "Problème lors de la suppression. Exit erreur 1."
fi

exit 0
```

d. del_cont_env.sh : assurera l'effacement des packages installés par le script inst_cont_env.sh

```
#! /bin/bash
if [ $(which lxc | grep -c lxc) -eq 0 ]
then
    echo "Rien a désinstaller. LXC n est pas present sur votre machine."
else
    apt-get --purge remove lxc
    apt-get --purge remove lxc-templates
fi
exit 0
```

2. Scripts de création / destruction de conteneur

Les questions b, c et d sont assez similaires même si les scripts semblent longs.

a. creat_container.sh : assure la création d'un conteneur LXC. Ce script prendra en paramètre l'ensemble des paramètres du conteneur : template, adresse réseau, configuration système (mémoire, CPU …), mot de passe root … Ce script assurera la modification du fichier de configuration du conteneur.

Pour information voici le lien avec tous les paramètres configurables dans LXC.

Je n'ai pas réussi à faire fonctionner la limitation de la mémoire, du CPU et la création d'un mot de passe root. Voici ce que j'avais essayé en sachant que l'appelle du script de création se faisait ainsi « `sudo ./start_container.sh ubuntu test2 10.0.0.1/24 1G 0.1 mdp` »:

```
nano  nano /var/lib/lxc/$nom_conteneur/config
#!/bin/bash
echo "root:mdp" | chpasswd
```

et

```
nano
# System configuration
lxc.cgroup.cpuset.cpus = $cpu
lxc.cgroup.memory.limit = $memoire

#root password
lxc.hook.pre-start = /usr/local/bin/chroot_script.sh
```

Ainsi, j'ai changé légèrement la consigne puisque ce script permet désormais seulement de créer un conteneur où il nous est permis de choisir le bridge et le template.

Attention, problème à régler, parfois l'adresse MAC générée aléatoirement ne fonctionne pas. Il faut alors la changer à la main.

```
#! /bin/bash

if [ $# -ne 3 ]
then
    echo "Le nombre d argument n est pas bon. "
    exit 1
else
    template=$1
    nom_conteneur=$2
    bridge=$3

    if [ $(lxc-ls -f | grep -c $nom_conteneur) -eq 0 ]
    then
        #On crée en fonction du choix de l utilisateur (ubuntu, debian ou alpine)
        if [ $template = "ubuntu" ]
        then
            echo "Creation d un conteneur LXC Ubuntu ... "
            lxc-create -n $nom_conteneur -t ubuntu

        elif [ $template = "debian" ]
```

```
        then
            echo "Creation d un conteneur LXC Debian ... "
            lxc-create -n $nom_conteneur -t download -- -d debian -r bullseye -a amd64

        elif [ $template = "alpine" ]
        then
            echo "Creation d un conteneur LXC Alpine ... "
            lxc-create -n $nom_conteneur -t download -- -d alpine -r 3.17 -a amd64

        else
            echo "Ce script ne permet pas la création de votre template."
            exit 2
        fi

        # Fonction pour générer un octet aléatoire (2 chiffres hexadécimaux)
        generate_octet() {
        echo $(openssl rand -hex 1)
        }

        # Fonction pour générer une adresse MAC aléatoire
        generate_mac_address() {
        local mac=""
        for ((i=0; i<6; i++)); do
            octet=$(generate_octet)
            mac+=":${octet}"
        done
        mac="${mac:1}" # Supprimer le premier ":" superflu
        echo "$mac"
        }

        # Génération d'une adresse MAC aléatoire car doit être unique
        random_mac=$(generate_mac_address)

        #Configuration du conteneur
        #copie du fichier de configuration
        cp /var/lib/lxc/$nom_conteneur/config /var/lib/lxc/$nom_conteneur/config.copy
        rm /var/lib/lxc/$nom_conteneur/config

        echo "# Common configuration"> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.include = /usr/share/lxc/config/common.conf">>
/var/lib/lxc/$nom_conteneur/config
        echo """>> /var/lib/lxc/$nom_conteneur/config
        echo "# Container specific configuration">> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.rootfs.path = dir:/var/lib/lxc/$nom_conteneur/rootfs">>
/var/lib/lxc/$nom_conteneur/config
        echo "lxc.uts.name = $nom_conteneur">> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.arch = amd64" >> /var/lib/lxc/$nom_conteneur/config
        echo """>> /var/lib/lxc/$nom_conteneur/config
        echo "# Network configuration">> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.net.0.type = veth">> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.net.0.link = $bridge">> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.net.0.flags = up">> /var/lib/lxc/$nom_conteneur/config
        echo "lxc.net.0.hwaddr = $random_mac">> /var/lib/lxc/$nom_conteneur/config

    else
        echo "Le nom $nom_conteneur désigne déjà un conteneur."
        exit 3
    fi
fi

exit 0
```

b. start_container.sh : assure le démarrage d'un ou de plusieurs conteneurs dont les identifiants sont passés en paramètres. Ce script permettra l'utilisation de jokers : cnt* désignera l'ensemble des conteneurs dont le nom commence par cnt.

```
#! /bin/bash
conteneurs_dossier="/var/lib/lxc/"
nbr_total_conteneurs=$(ls -l $conteneurs_dossier | grep -c  '^d')
if [ "$#" -eq 0 ] || [ "$#" -gt "$nbr_total_conteneurs" ];
then
    echo "Il faut au moins le nom d'un conteneur à allumer OU il est possible que
vous ayez fourni trop de noms (pas autant de disponnible). "
    exit 1
else
    #Afin de faciliter le traitement on met tous les noms passés en paramètre
dans un tableau (tab)
    tab=()
    for nom in "$@";
    do
        #On vérifie si un des noms passés contient une "*"
        case "$nom" in
            *"*"*)
                # Si oui, on extrait la partie avant l'étoile
                avant_etoile=$(echo "$nom" | grep -o '^[^*]*')

                # On récupère les conteneurs qui commencent par les caractères se
trouvant avant l'étoile
                conteneurs=$(ls "$conteneurs_dossier" | grep -E "^$avant_etoile")

                # On vérifie qu'au moins un conteneur corresponde au motif
spécifié
                if [ -n "$conteneurs" ]; then
                    # On les rajoute à tab pour les allumer
                    tab+=($conteneurs)
                else
                    echo "Aucun conteneur ne correspond au motif $nom."
                fi
                ;;
            *)
                if [ -d "$conteneurs_dossier/$nom" ]; then
                    tab+=("$nom")
                else
                    echo "Le nom $nom n'est pas présent dans le dossier
$conteneur_dir, il ne sera donc pas allumé."
                fi
                ;;
        esac

    done
    echo "On allume le ou les conteneur(s) suivant(s) : ${tab[@]}"

    #On allume les conteneurs présents dans tab
    for conteneur in "${tab[@]}"; do
        if lxc-info -n "$conteneur" | grep -q "RUNNING"; then
            echo "Le conteneur $conteneur est déjà en cours d'exécution."
        else
```

```
            if lxc-start -n "$conteneur"; then
                echo "$conteneur est maintenant allumé."
            else
                echo ""
                echo "Erreur : Le conteneur $conteneur a rencontré une erreur
lors du démarrage."
                echo ""
            fi
        fi
    done
fi
```

c. stop_container.sh : assure l'arrêt d'un ou de plusieurs conteneurs dont les identifiants sont passés en paramètres. Ce script permettra l'utilisation de jokers : cnt* désignera l'ensemble des conteneurs dont le nom commence par cnt.

Hormis les « echo » il n'y a réellement que la partie violette qui diffère du scripte b.

```
#! /bin/bash
conteneurs_dossier="/var/lib/lxc/"
nbr_total_conteneurs=$(ls -l $conteneurs_dossier | grep -c  '^d')
if [ "$#" -eq 0 ] || [ "$#" -gt "$nbr_total_conteneurs" ];
then
    echo "Il faut au moins le nom d'un conteneur à etteindre OU il est possible
que vous ayez fourni trop de noms (pas autant de disponnible). "
    exit 1
else
    #Afin de faciliter le traitement on met tous les noms passés en paramètre
dans un tableau (tab)
    tab=()
    for nom in "$@";
    do
        #On vérifie si un des noms passés contient une "*"
        case "$nom" in
            *"*"*)
                # Si oui, on extrait la partie avant l'étoile
                avant_etoile=$(echo "$nom" | grep -o '^[^*]*')

                # On récupère les conteneurs qui commencent par les caractères se
trouvant avant l'étoile
                conteneurs=$(ls "$conteneurs_dossier" | grep -E "^$avant_etoile")

                # On vérifie qu'au moins un conteneur corresponde au motif
spécifié
                if [ -n "$conteneurs" ]; then
                    # On les rajoute à tab pour les allumer
                    tab+=($conteneurs)
                else
                    echo "Aucun conteneur ne correspond au motif $nom."
                fi
                ;;
            *)
                if [ -d "$conteneurs_dossier/$nom" ]; then
                    tab+=("$nom")
                else
```

```
                    echo "Le nom $nom n'est pas présent dans le dossier
$conteneur_dir, il ne sera donc pas éteint."
                fi
                ;;
        esac

    done
    echo "On éteint le ou les conteneur(s) suivant(s) : ${tab[@]}"

    #On etteint les conteneurs présents dans tab s'ils sont en "RUNNING"
    for conteneur in "${tab[@]}"; do
        if lxc-info -n "$conteneur" | grep -q "RUNNING"; then
            if lxc-stop -n "$conteneur"; then
                echo "$conteneur est maintenant éteint."
            else
                echo ""
                echo "Erreur : Le conteneur $conteneur à rencontré une erreur
lors de son arrêt."
                echo ""
            fi
        else
            echo "Le conteneur $conteneur est déjà en éteint."
        fi
    done
fi
```

d. del_container.sh : assure la destruction d'un ou de plusieurs conteneurs dont les identifiants sont passés en paramètres. Ce script permettra l'utilisation de jokers : cnt* désignera l'ensemble des conteneurs dont le nom commence par cnt.

Hormis les « echo » il n'y a réellement que la partie violette qui diffère du scripte b.

```
#! /bin/bash
conteneurs_dossier="/var/lib/lxc/"
nbr_total_conteneurs=$(ls -l $conteneurs_dossier | grep -c  '^d')
if [ "$#" -eq 0 ] || [ "$#" -gt "$nbr_total_conteneurs" ];
then
    echo "Il faut au moins le nom d'un conteneur à supprimer OU il est possible
que vous ayez fourni trop de noms (pas autant de disponnible). "
    exit 1
else
    #Afin de faciliter le traitement on met tous les noms passés en paramètre
dans un tableau (tab)
    tab=()
    for nom in "$@";
    do
        #On vérifie si un des noms passés contient une "*"
        case "$nom" in
            *"*"*)
                # Si oui, on extrait la partie avant l'étoile
                avant_etoile=$(echo "$nom" | grep -o '^[^*]*')

                # On récupère les conteneurs qui commencent par les caractères se
trouvant avant l'étoile
```

```
                conteneurs=$(ls "$conteneurs_dossier" | grep -E "^$avant_etoile")

                # On vérifie qu'au moins un conteneur corresponde au motif
spécifié
                if [ -n "$conteneurs" ]; then
                    # On les rajoute à tab pour les allumer
                    tab+=($conteneurs)
                else
                    echo "Aucun conteneur ne correspond au motif $nom."
                fi
                ;;
            *)
                if [ -d "$conteneurs_dossier/$nom" ]; then
                    tab+=("$nom")
                else
                    echo "Le nom $nom n'est pas présent dans le dossier
$conteneur_dir, il ne sera donc pas supprimé."
                fi
                ;;
        esac

    done
    echo "On supprime le ou les conteneur(s) suivant(s) : ${tab[@]}"


    for conteneur in "${tab[@]}"; do
        #On etteint les conteneurs présents dans tab s'ils sont en "RUNNING" car
sinon ne fonctionne pas
        if lxc-info -n "$conteneur" | grep -q "RUNNING"; then
            ./stop_container.sh
        fi
        if lxc-destroy -n "$conteneur"; then
            echo "$conteneur est maintenant supprimé."
        else
            echo ""
            echo "Erreur : Le conteneur $conteneur a rencontré une erreur lors de
sa suppression."
            echo ""
        fi
    done
fi
```