Activité : installation et découverte de Kathara

Installation de Kathara

- Lien vers le Wiki : https://github.com/KatharaFramework/Kathara/wiki/Linux
- Présentation de Kathara : https://github.com/KatharaFramework/Kathara-Labs/blob/master/001-kathara-introduction.pdf

Cet atelier concerne l'installation de Kathara dans un environnement Linux Ubuntu ou Debian.

> **IMPORTANT**
> Vous allez installer Kathara sous le compte **btssio** et non pas sous le compte root.

Il est nécessaire d'installer l'émulateur de terminal X appelé **xterm**. Le programme xterm est un émulateur de terminal pour X Window utilisé par les programmes qui ne peuvent pas utiliser directement le système de fenêtrage.

```
$ sudo apt install xterm
```

Installation pour Ubuntu

- Ajouter le dépôt de Kathara :

```
$ sudo add-apt-repository ppa:katharaframework/kathara
```

- Mettre à jour la liste des paquets disponibles

```
$ sudo apt update
```

- Installer Kathara

```
$ sudo apt install kathara
```

Installation pour Debian

- Ajouter la clé publique du dépôt de Kathara

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 21805A48E6CBBA6B991ABE76646193862B759810
```

- Créer le fichier **kathara.list** dans le dossier **/etc/apt/sources.list.d** et y ajouter les lignes suivantes :

```
deb http://ppa.launchpad.net/katharaframework/kathara/ubuntu bionic main
deb-src http://ppa.launchpad.net/katharaframework/kathara/ubuntu bionic main
```

- Mettre à jour la liste des paquets disponibles

```
$ sudo apt update
```

- Installer Kathara

```
$ sudo apt install kathara
```

Tester le bon fonctionnement de Kathara

```
$ kathara check
```

Cela va prendre également un peu de temps pour télécharger les images Docker nécessaires...

- Vous pouvez ensuite visualiser un conteneur Docker (**kathara/quagga**) installé pour le fonctionnement de Kathara :

```
btssio@ubuntudocker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
kathara/quagga      latest              6b9b242d2656        10 months ago       698MB
btssio@ubuntudocker:~$
```

Le jeu de commandes de Kathara

Netkit fournit deux groupes de commandes :

- les **vcommandes**, préfixées par **'v'**, qui permettent de manipuler une seule VM à la fois ;
- les **lcommandes**, préfixées par **'l'** qui servent à manipuler des ensembles complexes de machines virtuelles en réseau. Dans le langage de Kathara, il s'agit des **Labs** (laboratoires).

Si vous souhaitez travailler avec une seule VM, utilisez les vcommandes. Sinon, pour travailler avec plusieurs VMs, il est préférable et bien plus pratique de créer un laboratoire (Lab) et d'utiliser alors les lcommandes.

Utilisation de machines autonomes

les v-commandes

Commande	Action
vstart	
vlist	→ donner la liste des VMs actives
vconfig	→ configurer à la volée une VM comme par exemple affecter une interface à la volée.
vclean	→ arrêter une VM et nettoyer les processus, configurations et fichiers temporaires créés

Gérer une VM

Lien : https://www.kathara.org/man-pages/kathara-vstart.1.html

Pour s'assurer du bon fonctionnement de Kathara, vous pouvez créer depuis un terminal une première machine virtuelle avec le nom **sta1**.
Création d'une VM avec vstart

```
btssio@ubuntudocker:~$ kathara vstart -n sta1 --eth 0:HubDCA --bridged
```

Explications :

- La commande vstart permet de lancer en interactif une VM ;
- Le nom de la VM **sta1** est le paramètre suivant précédé de -n
- **-eth** permet de définir le numéro de l'interface réseau **eth0** associée à au domaine de collision **HubDCA** (hub virtuel) ; Un domaine de collision correspond à un **concentrateur** pour Kathara. Il faudra définir manuellement une adresse IP avec ifconfig par exemple.
- –bridged permet de créer une 2ème interface eth1 qui relie la VM en NAT à l'hôte sur le bridge (pont) Docker.

Voici votre première VM Netkit avec la session root automatiquement ouverte :

Pour cette VM sta1, aucune adresse IP n'a été définie. C'est le bridge Docker créé par Kathara qui a fourni la configuration IP 172.20.0.2/16 dans le réseau 172.20.0.0/16. La passerelle est 172.20.0.1/16 et la VM accède à Internet

Pour visualiser le bridge créé par Kathara, tapez la commande suivante dans le terminal de votre serveur Debian/Ubuntu :

```
btssio@ubuntudocker:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:af:88:bf brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.199/24 brd 192.168.1.255 scope global dynamic enp0s3
    valid_lft 82586sec preferred_lft 82586sec
    inet6 fe80::a00:27ff:feaf:88bf/64 scope link
    valid_lft forever preferred_lft forever
3: br-8edf20a49895: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc no-queue state DOWN group default
    link/ether 02:42:64:3d:be:51 brd ff:ff:ff:ff:ff:ff
    inet 172.20.0.1/16 brd 172.20.255.255 scope global br-8edf20a49895
    valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:1d:98:0e:31 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
```

```
    valid_lft forever preferred_lft forever
btssio@ubuntudocker:~$
```

> **INFORMATION**
> Le sous-réseau 172.0.0.0/8 est réservé pour l'utilisation de Kathara. Il ne faut donc pas l'utiliser pour définir des plans d'adressage de vos sous-réseaux. Lors de la création de VMs autonomes ou dans des labs comme vous le verrez ensuite, Kathara va créer autant de bridges que vous définissez de domaine de collision en leur associant un sous-réseau différent basé sur ce sous-réseau 172.0.0.0/8. Le premier de ces sous-réseaux est 172.19.0.0/16, le dernier est 172.255.0.0/16.

Visualisation de la VM créée avec vlist

```
btssio@ubuntudocker:~$ vlist
CONTAINER ID NAME             CPU % MEM USAGE / LIMIT MEM %  NET I/O BLOCK I/O PIDS
8ceba73bfb3f netkit_1000_sta1   0.00%  2.16MiB/1.419GiB  0.15%  6.58kB/0B  12.2MB/0B 2
NETWORK ID          NAME                DRIVER              SCOPE
f332c80a9dbd        bridge              bridge              local
c28efeb7848e        host                host                local
2877cae4fa72        netkit_1000_H       bridge              local
708458e85954        none                null                local
btssio@ubuntudocker:~$
```

Vous pouvez visualiser :

- Les caractéristiques de la VM **sta1** : son **ID Docker 8ceba73bfb3f** ainsi que les ressources consommées ;
- La liste des interface réseaux du serveur Debian/Ubuntu qui montre le bridge **netkit*1000*H** créé par Kathara et qui est associé au domaine de collision **HubDCA**.

Le commande Docker montre le conteneur **8ceba73bfb3f** qui correspond à **sta1** et l'image **kathara/netkit*base qui a été utilisée. <code shell> btssio@ubuntudocker:~$ docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 8ceba73bfb3f kathara/netkitbase "/bin/bash" 3 m… Up 3 m… netkit1000sta1 … </code> Cette autre commande permet également de visualiser le container Docker créé et vous pouvez le visualiser avec la commande suivante : <code shell> btssio@ubuntudocker:~$ docker container ls CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 8ceba73bfb3f kathara/netkitbase "/bin/bash" 4 m… Up 4 m… netkit1000sta1 btssio@ubuntudocker:~$ </code> Pour arrêter la VM avec une commande Kathara ; le container est alors supprimé :***

> **INFORMATION**
> La commande **vconfig** semble actuellement ne pas fonctionner convenablement

Arrêter une VM avec vclean Le container est alors supprimé : <code shell> btssio@ubuntudocker:~$ kathara vclean -n sta1 Any network still in use by another machine will not be deleted (and will raise an error instead) Containers will be deleted netkit*1000*sta1 netkit*1000*H btssio@ubuntudocker:~$ docker container ls CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES btssio@ubuntudocker:~$ </code> ==== Utilisation des labs ==== === Pourquoi utiliser les labs ? === Quand vous avez à créer et gérer une seule VM, vous utilisez les v-commandes. Pour des infrastructures utilisant plusieurs VMs, il est préférable de créer un laboratoire ou **lab** et de manipuler ce lab avec les **l-commandes**. Ces labs permettent de concevoir mais aussi de conserver une architecture réseau complexe ou que l'on souhaite pouvoir réutiliser et cela d'autant plus facilement qu'un lab se traduit par une arborescence de dossiers contenant des fichiers de configuration. Un lab occupe très peu de place et est facile à sauvegarder et à échanger. Vous trouverez sur Internet des ressources et des exemples de labs. **Voici des liens vers les labs proposés par l'équipe de Kathara :** * https://github.com/KatharaFramework/Kathara-Labs/wiki Sur ce site, vous trouverez de la documentation ainsi que des exemples de simulations qui peuvent être très complexes. Un Lab **minimaliste** consiste en une arborescence comprenant : * Obligatoirement un fichier de configuration (**lab.conf**) qui décrit les machines virtuelles qui seront lancées, leurs interfaces et les domaines de collision. * un **répertoire** par machine virtuelle qui sera lancée, dans lequel on peut stocker des fichiers. Pour l'instant, laissez ce répertoire vide. * Eventuellement pour les VM des fichiers **VMx.startup** et/ou un fichier **VMx.shutdown** qui indiquent les actions à réaliser lors du lancement ou de l'arrêt de la VMx (VMx correspond au nom de la VM Kathara). Cela permet de configurées automatiquement la VMx lors du lancement du lab à l'aide de scripts shell. Les scripts doivent être exécutables. **Les l-commandes** Les l-commandes sont utilisables dans le répertoire du lab qui doit contenir au minimum le fichier lab.conf, ex-ception faire de la commande **lwipe**. ^ Commande ^ Action ^ |lstart|→ pour lancer un lab.| |lclean|→ arrêter les VMs et nettoyer les processus, configurations et fi-chiers temporaires créés| |linfo|→ information sur le laboratoire.| |ltest|→ vérification du bon fonctionnement du laboratoire.| |lwipe|→ arrêter les VMs des labs de Kathara et arrêter une VM et nettoyer les processus, configurations et fichiers temporaires créés.| Dans l'activité sur les infrastructures réseaux avec Kathara sera abordé la création des labs : * la création de l'arborescence des dossiers du lab, * le contenu du fichier lab.conf, * la création des fichiers qui permettent de personnaliser le fonctionnement des VMs, * l'utilisation des l-commandes, * l'ipmasquerade pour configurer une VM comme routeur NAT. ===== Etendre les fonctionnalités de l'image Kathara ===== Lorsque Kathara crée une VM, avec une v-commande ou une l-commande, c'est un conteneur qui est créé par Docker à partir de l'image **kathara/netkit_base**. L'équipe de développement de Kathara a intégré à cette image un certain nombre de paquets logiciels de telle sorte que les VMs créées puissent les utiliser. Cependant, si dans la réalisation de votre maquette vous avez besoin d'un paquet logiciel qui n'est pas présent dans les VMs, vous pouvez **l'ajouter** à l'image **kathara/netkit_base** afin que toutes les VMs qui seront ensuite créées puissent l'utiliser. La démarche qu'il faut suivre est la suivante : *

Créer un conteneur avec image kathara/netkit*base, <code shell> btssio@ubuntudocker:~$ docker run -itd –name kathara*btssio kathara/netkit_base </code>

INFORMATION
Le conteneur est créé en lui associant un nom à votre convenance, ce qui sera plus facile pour l'identifier.

* **Se connecter** à une console du conteneur : <code shell> btssio@ubuntudocker:~$ docker exec -it kathara_btssio bash root@93e38cf2c65:/# </code> * **Mettre à jour** le conteneur : <code shell> root@93e38cf2c65:/#apt-get update && apt-get upgrade </code> * **Installer** les paquets logiciels voulus ; pour cet exemple la bibliothèque scapy pour python : <code shell> root@93e38cf2c65:/#apt-get install scapy </code> * **Quitter** le conteneur : <code shell> root@93e38cf2c65:/#apt-get clean && exit </code> * **Enregistrer** le conteneur modifié dans l'image **actuelle** ou comme une **nouvelle image**. La deuxième solution sera utilisée car elle permet de garder l'image de base et dans ce cas il faut faudra indiquer le nom de cette nouvelle image, soit pour toutes les VMs, soit uniquement pour celles qui en ont besoin. * **Pour information : enregistrer** le conteneur dans l'image actuelle : <code shell> btssio@ubuntudocker:~$ docker commit kathara*btssio kathara/netkit*base </code> * **A faire : enregistrer** le conteneur dans une nouvelle image: <code shell> btssio@ubuntudocker:~$ docker commit kathara*btssio kathara/netkit*btssio </code> * **Utiliser** la nouvelle image pour tous les conteneurs Kathara: Il est nécessaire de modifier la directive IMAGE*NAME de Kathara en modifiant à ligne 15 du fichier* ***/opt/Kathara/bin/python/netkitcommons.py : <code shell> btssio@ubuntudocker:~$ nano /opt/Kathara/bin/python/netkit_commons.py </code> * Utiliser la nouvelle image pour une seule VM d'un lab : Il suffit d'indiquer dans le fichier lab.conf un paramètre supplémentaire précisant l'image à utiliser : <code shell> Routeur[image]=netkit_btssio </code> Vous disposez maintenant de deux images pour les VMs de Kathara : * L'image de base kathara/netkitbase, * Et une image personnalisée kathara/netkit_btssio. ===== ne plus utiliser===== * Téléchargez les fichiers de Kathara et l'interface graphique GUI dans le dossier /opt : <code shell> $ cd /opt $ sudo git clone –recursive https://github.com/KatharaFramework/Kathara.git </code> * Exportez les variables d'environnement : <code shell> btssio@ubuntudocker:~$ export NETKITHOME=/opt/Kathara/bin btssio@ubuntudocker:~$ export PATH=$PATH:$NETKITHOME </code>***

ATTENTION
Le dossier **/opt/Kathara** s'écrit avec un **K majuscule** !

Vous devrez réaliser cela à chaque fois que vous lancerez la VM Ubuntu ou que vous ouvrirez une nouvelle ses-sion avec le compte BTSSIO. Pour éviter cela, ajouter ces deux lignes à la fin du fichier **.bashrc** qui est dans votre répertoire /home/btssio/. Puis fermez et réouvrez votre session. <code shell> btssio@ubuntudocker:~$ nano .bashrc </code> * **Ajoutez** les variables d'environnement : <code shell> ... export NETKIT*HOME=/opt/Kathara/bin export PATH=$PATH:$NETKIT*HOME </code> * **Fermer** puis **ouvrez à nouveau** votre session BTSSIO : * **Lancez** l'installation de Kathara : <code shell> btssio@ubuntudocker:~$ $NETKIT_HOME/install </code> ====== Retour Accueil SISR3 ====== * SISR3