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	
list	→ 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 pour eth0. Mais l'adresse IP 172.17.0.2/16 est définie pour **eth1** dans le réseau du bridge Docker. La passerelle est 172.20.0.1/16 (carte virtuelle Docker sur l'hôte) et la VM accède à Internet grâce au NAT défini sur cette interface Docker de l'hôte.

Visualisation de la VM créée avec list

```
btssio@ubuntudocker:~$ kathara list
TIMESTAMP: 2020-11-01 15:10:35.932897

╔══════════════════════╦═══════╦══════════════╦═════════╦═══════╦═══════════════════╦═══════╦═══════
═══╗
║ LAB HASH             ║ USER  ║ MACHINE NAME ║ STATUS  ║ CPU % ║ MEM USAGE / LIMIT ║ MEM % ║ NET I/O
   ║
╠══════════════════════╬═══════╬══════════════╬═════════╬═══════╬═══════════════════╬═══════╬═══════
═══╣
║ BquVk2860DTFrej8slHuVg ║ btssio ║ sta1         ║ running ║ 0.00% ║ 2.37 MB / 1.95 GB ║ 0.12% ║ 9.94 KB
/ 0 B ║
╚══════════════════════╩═══════╩══════════════╩═════════╩═══════╩═══════════════════╩═══════╩═══════
═══╝
btssio@ubuntudocker:~$
```

Vous pouvez visualiser :

- Les caractéristiques de la VM **sta1** : son **LAB HASH BquVk2860DTFrej8slHuVg** ainsi que les ressources consommées ;

La commande Docker montre le conteneur **katharabtssiosta1***BquVk2860DTFrej8slHuVg qui correspond à sta1 et l'image kathara/quagga qui a été utilisée. <code shell> btssio@ubuntudocker:~$ docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES c0e2f9e0e068 kathara/quagga "bash" 8 minutes ago Up 8 minutes katharabtssiosta1_BquVk2860DTFrej8slHuVg btssio@ubuntudocker:~$ </code> Arrêter une VM avec vclean Le container est alors supprimé : <code shell> btssio@ubuntudocker:~$ kathara vclean -n sta1 Deleting machines… |################################| 1/1 Deleting links… |################################| 1/1 INFO - Machine sta1 deleted successfully! 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/netkitbase, <code shell> btssio@ubuntudocker:~$ docker run -itd –name katharabtssio 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 katharabtssio kathara/netkitbase </code> * **A faire : enregistrer** le conteneur dans une nouvelle image: <code shell> btssio@ubuntudocker:~$ docker commit katharabtssio kathara/netkitbtssio </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. ====== Retour Accueil SISR3 ====== * SISR3***