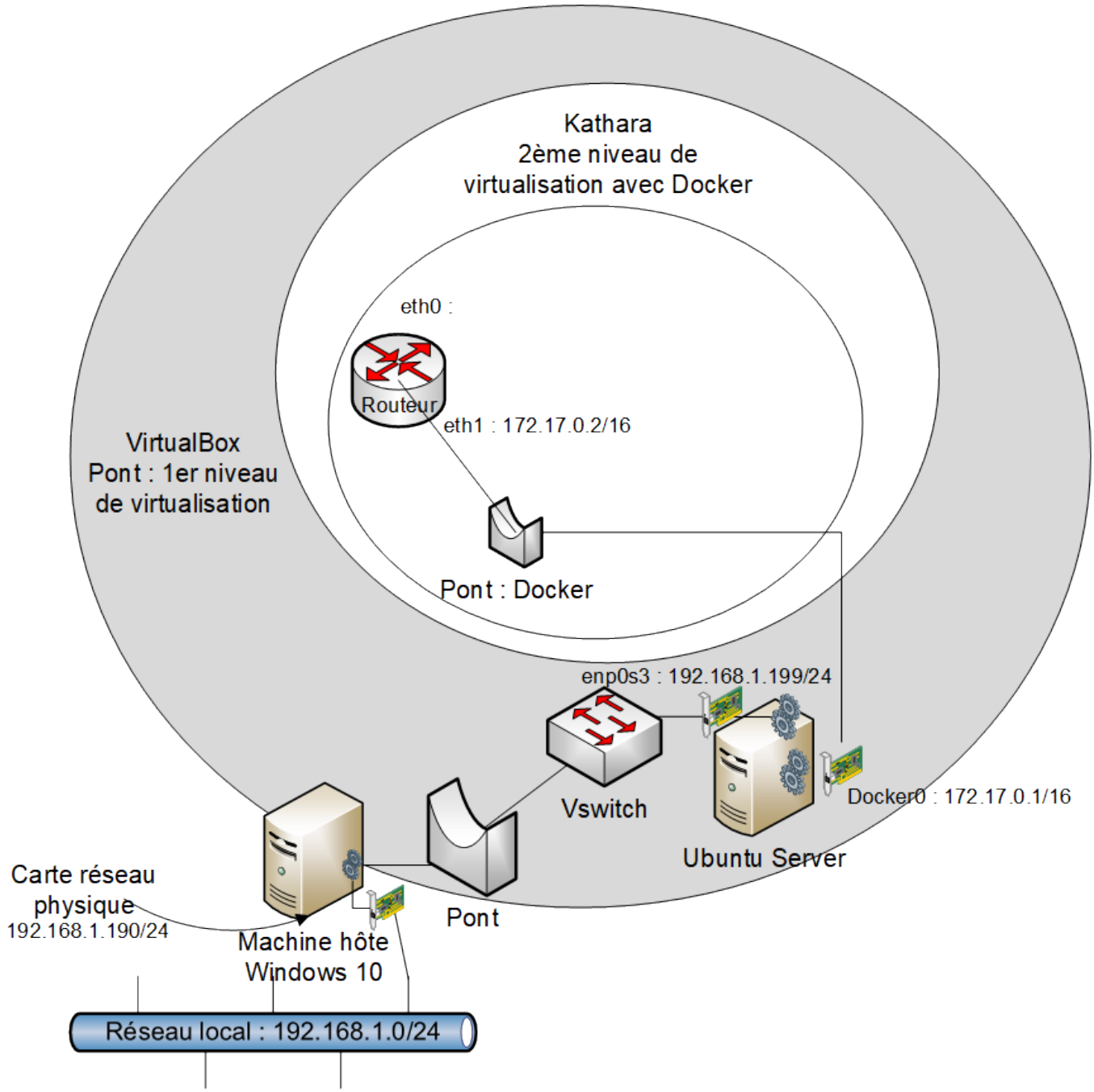
Activité : créer des infrastructures réseaux avec Kathara

Les VM autonomes avec Kathara

Créer un routeur

Voici le réseau à obtenir :



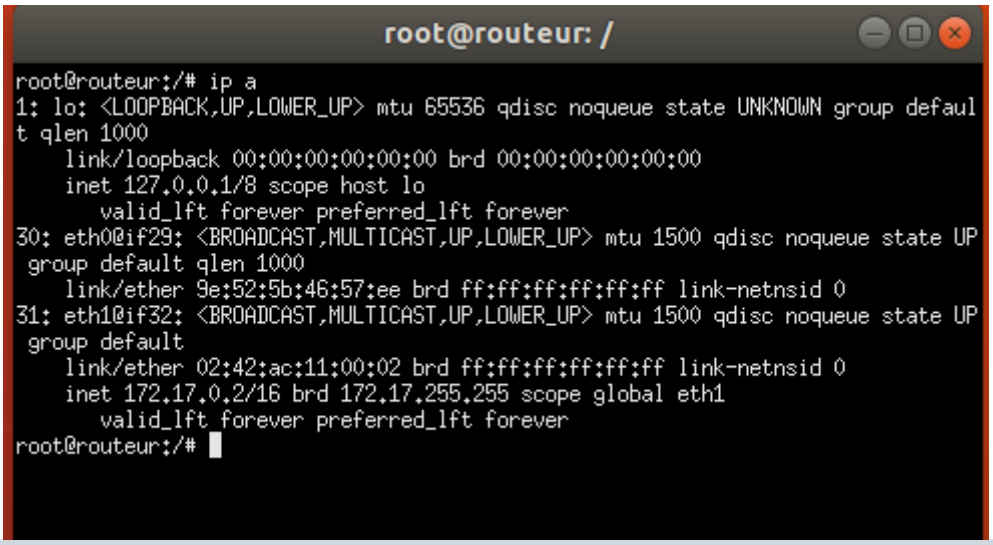
Le **routeur** va être créé en tant que **machine virtuelle**, avec une interface réseau **eth0** pour l'instant sans adressage IP et une interface réseau **eth1** connectée au **bridge Docker** (configuration par Docker).

INFORMATION
Votre VM est créée pour être un routeur. De ce fait, ce routeur est représenté dans le schéma avec le symbole du routeur plutôt qu'avec le symbole d'un ordinateur.

Dans le terminal, tapez la commande suivante :

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

Prenez la bonne habitude de donner des **noms significatifs** aux VMs comme aux domaines de collision (ici HubDCA).



INFORMATION
La VM routeur est créée avec une adressez IP fournie par le bridge de Docker dans le **sous-réseau 172.17.0.0/16**.

- La commande **vclean** permet de supprimer une VM autonome ;
- La commande **wipe** permet de supprimer toutes les VM créées ainsi que les bridges qui ne sont plus utilisés.

Supprimez toutes les VMs créées avec wipe et vérifiez que les interfaces des bridges Docker soit bien également supprimées avec les commandes ifconfig ou ip.

- Exemple :

```
btssio@ubuntudocker:~$ lwipe
btssio@ubuntudocker:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group …
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel …
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue …
```

Il ne devrait plus rester que 3 interfaces : **lo, enps03 et docker0**.

La résolution de nom DNS

Les VM créés sont correctement configurées avec la passerelle par défaut mais la résolution de noms ne fonctionne pas :

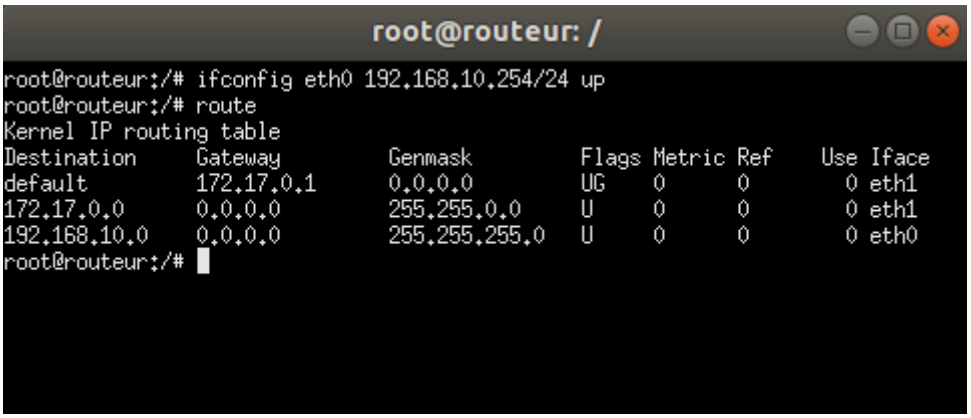
```
root@routeur:~#   ping www.google.fr
```

Modifiez le contenu du fichier **/etc/resolv.conf** qui est présent dans les VM comme la VM routeur pour ajouter un DNS public comme celui de Google (8.8.8.8):

```
nameserver 8.8.8.8
```

Restez positionné dans VM routeur et configurez son adresse IP pour l'interface **eth0**:

```
root@routeur:/# ifconfig eth0 192.168.10.254/24 up
```



Utilisation des labs

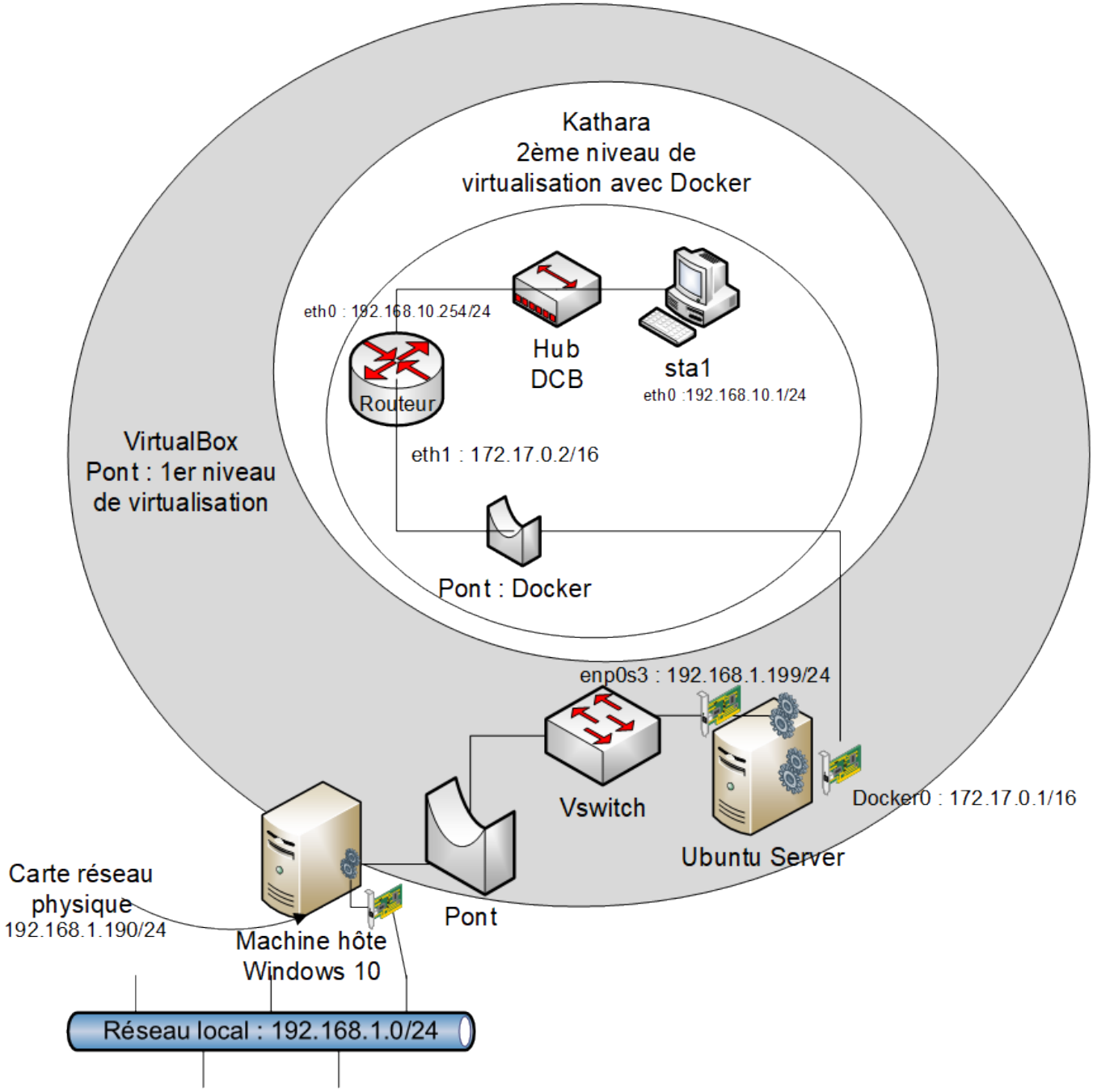
Quand vous avez à créer et gérer une seule VM, vous utilisez les v-commandes. Pour des infrastructures utili-sant plusieurs VMs, il est préférable de créer un laboratoire ou **lab** et de manipuler ce lab avec les **l-commandes**. Ces labs permettent de concevoir mais aussi de conserver une architecture 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.

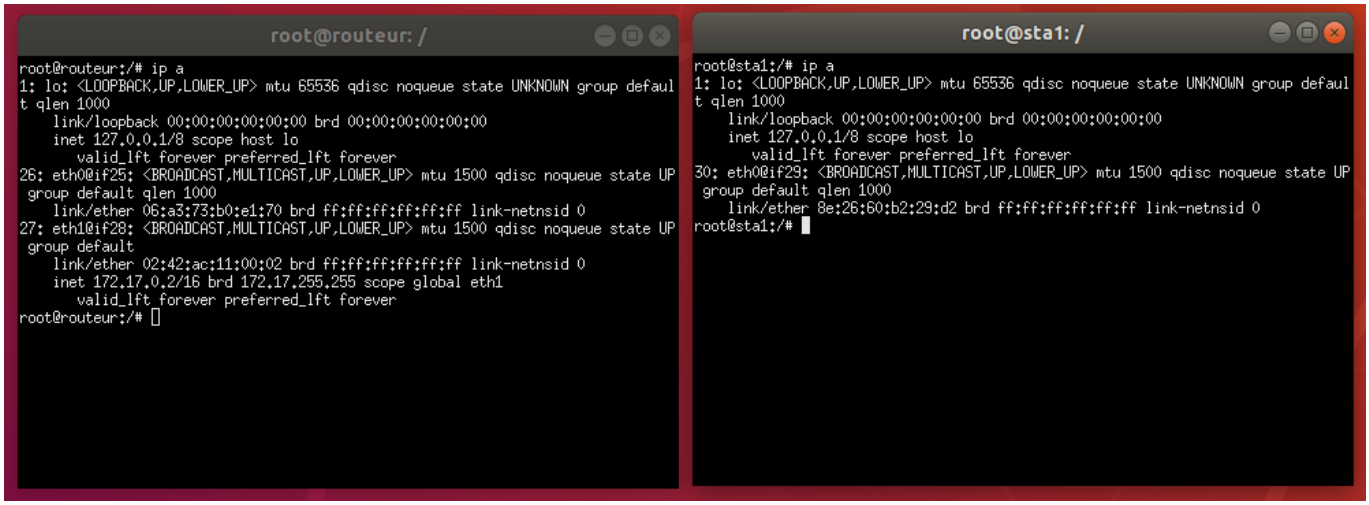
Un premier lab à réaliser



- dans le dossier Documents de btssio, créez le sous-dossier lab1 ;
- dans le sous-dossier **lab1** , créez les sous-dossiers **routeur** et **sta1** ;
- créez un fichier **lab.conf** avec le contenu suivant ; l'interface **eth0** du routeur est indiqué **sans nom de domaine de collision** afin qu'elle soit configurée automatiquement par le bridge de Docker. Cette interface doit permettre de sortir sur Internet.

```
routeur[0]=HubDCB
routeur[bridged]=true
sta1[0]=HubDCB
```

- lancez le Terminal et positionnez-vous dans le dossier **lab1** ;
- tapez la commande **kathara lstart** pour lancer le **lab** et créer les deux VMs.



Pour la VM routeur, les deux interfaces eth0 et eth1 sont chacune sur deux bridges Docker différents. L'interface eth0 de la VM sta1 est sur le même bridge Docker que l'interface eth0 de la VM routeur.

Pour configurer correctement les 2 interfaces concernées, les fichiers **routeur.startup** et **sta1.startup** doivent être créés.

- **créez** dans le dossier **lab1** le fichier **routeur.startup** avec le contenu suivant pour définir l'adresse IP de eth0 :

```
ifconfig eth0 192.168.10.254/24 up
```

- **créez** dans le dossier **lab1** le fichier **sta1.startup** avec le contenu suivant pour définir l'adresse IP de eth0 et définir sa passerelle:

```
ifconfig eth0 192.168.10.1/24 up
route add default gw 192.168.10.254
```

- **Arrêtez** le lab et **relancez-le** :

```
btssio@ubuntudocker:~/Documents/lab1$ kathara wipe
btssio@ubuntudocker:~/Documents/lab1$ kathara lstart
```

Dans chaque VM, modifiez le fichier **/etc/resolv.conf** pour ajouter une ligne (nameserver 8.8.8.8) afin d'avoir une résolution de nom.

Les adresses IP des interfaces des 2 VMS reliées au domaine de collision HubDCB ont été configurées par les fichiers **.startup**.

La **VM routeur** peut accéder à Internet. La **VM sta1** ne peut pas sortir sur Internet. En effet il faut activer **l'ipmasquerade** sur le routeur pour avoir du NAT.

Ipmasquerade

Si vous avez bien réalisé cet atelier, **sta1** doit pouvoir communiquer avec **Routeur** mais pas avec l'**hôte Ubuntu** (ou Debian) et n'a pas accès à votre réseau local ni à Internet.

Visualisons les routes de tous ces hôtes :

Pour sta1 et à partir de cet hôte :

```
root@sta1:~#   route -n
```

Une ligne indiquant un '**UG**' confirme qu'il y a bien une route UP (U) et par défaut (G). Cette route par défaut est bien l'adresse IP de l'interface eth1 du routeur.

```
root@sta1:/# ip route
default via 192.168.10.254 dev eth0
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.1
root@sta1:/#
```

Pour routeur et à partir de cet hôte :

```
root@routeur:~#   route –n
```

routeur peut envoyer les paquets via son interface **eth0** au réseau 192.168.10.0/24 (celui de **sta1**) et via son interface **eth1** vers le réseau 172.17.0.0/16 qui est celui du bridge Docker.

```
root@routeur:/# ip route
default via 172.17.0.1 dev eth1
172.17.0.0/16 dev eth1 proto kernel scope link src 172.17.0.2
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.254
root@routeur:/#
```

Pour l'hôte Server et à partir de cet hôte :

```
btssio@ubuntudocker:~/Documents/lab1$ sudo route -n
```

Votre **VM Ubuntu** ne connait pas le réseau 192.168.10.0 (celui de **sta1**).

```
btssio@ubuntudocker:~/Documents/lab1$ sudo route -n
[sudo] password for btssio:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG    100    0        0 enp0s3
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 enp0s3
192.168.1.254   0.0.0.0         255.255.255.255 UH    100    0        0 enp0s3
btssio@ubuntudocker:~/Documents/lab1$
```

Les paquets envoyés par **routeur** depuis le réseau 192.168.10.0 arrive bien à l'hôte Ubuntu par son interface **docker0**. Cependant la VM Ubuntu ne connaissant pas où se trouve le réseau 192.168.10.0, va chercher à envoyer les paquets vers sa passerelle par défaut 192.168.1.254 qui est la box Internet. Celle-ci bien sûr ignore également ce réseau 192.168.10.0. Cela explique pourquoi la VMs **sta1** ne peut pas communiquer sur le réseau local et sur Internet. Pour cela, il va falloir activer **l'ipmasquerade** sur la VM routeur. Le principe est de **masquer** la VM **sta1** pour le réseau local et d'utiliser l'adresse IP de **routeur** pour toute communication sur le réseau local. Comme vous l'avez compris il s'agit de configurer un routage **NAT** sur la VM **routeur** pour la VM **sta1**.

La commande à utiliser sur **routeur** est la suivante :

```
root@routeur:~#   iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

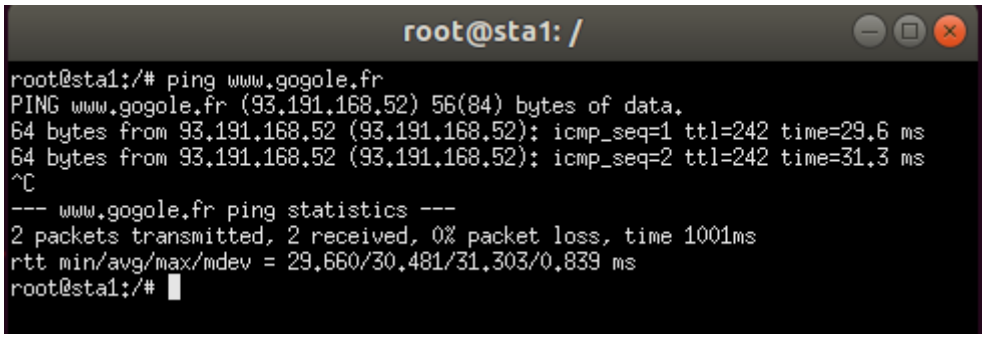
La règle utilise la table de correspondance de paquets de NAT (-t nat) et spécifie la chaîne intégrée POSTROUTING pour NAT (-A POSTROUTING) sur le périphérique réseau externe du pare-feu (-o eth1 de **routeur**). POSTROUTING permet aux paquets d'être modifiés lorsqu'ils quittent le périphérique externe du pare-feu. La cible -j MASQUERADE est spécifiée pour masquer l'adresse IP privée d'un nœud avec l'adresse IP externe du pare-feu / de la passerelle.

Cette **commande iptables** est à rajouter au fichier **routeur.startup** pour qu'elle soit exécutée au lancement du lab.

Cela doit permettre à la VM **sta1** située derrière la VM **routeur** d'accéder à la machine hôte et à internet.

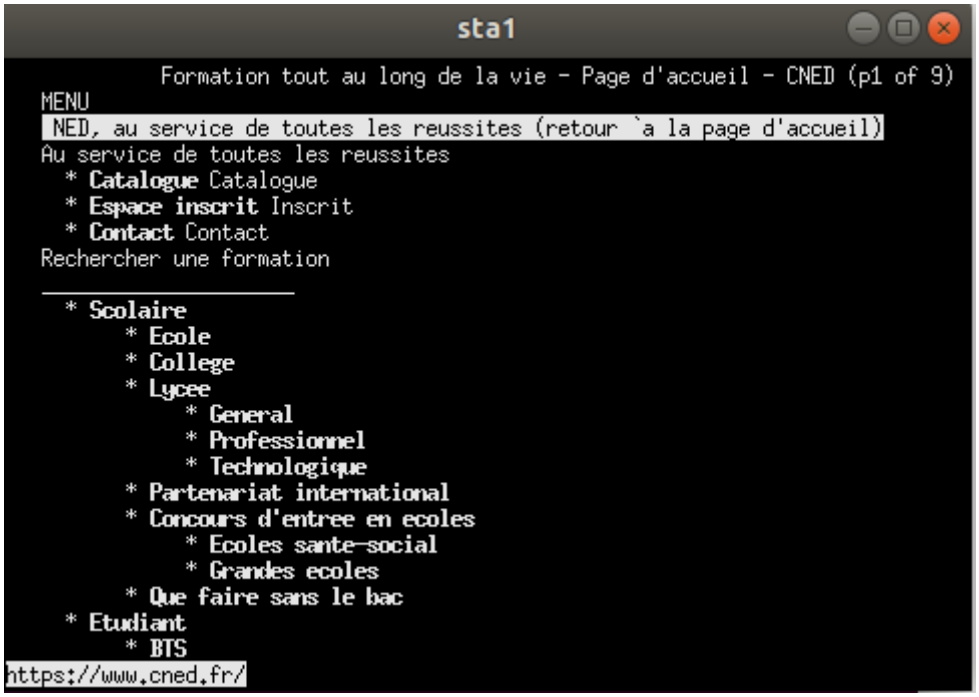
Maintenant la VM sta1 accède à Internet. Pour le vérifier :

```
root@sta1:~#   ping www.google.fr
```



Vous pouvez également utiliser le navigateur en mode texte **links** et accéder à un site Web de la manière suivante :

```
root@sta1:~# links www.cned.fr
```



Appuyer sur la touche ESC pour accéder au menu de links afin de quitter le logiciel.

En revanche, il n'est pas possible d'accéder à la VM **sta1** située après la VM routeur depuis l'hôte Ubuntu. Le réseau 192.168.10.0 est inaccessible.

Nous verrons plus loin comment remédier à cette situation. Mais vous avez peut-être une petite idée ?

INFORMATION
Quand vous **testez** des labs avec Kathara il est nécessaire **d'arrêter** les VMs pour pouvoir ensuite **relancer** le lab afin de vérifier qu'il a été correctement réalisé (l'arborescence et les différents fichiers de configuration). L'arrêt des VMs d'un lab se fait avec la commande **lclean**.

Ce premier lab était simpliste car on peut aller plus loin dans la configuration du lab.

Le fichier lab.conf

Le fichier lab.conf contient :

- la description du lab ;
- les paramètres des VMs ;
- la topologie du réseau.

Chaque machine est une déclaration **machine[arg]=valeur**

- machine est le nom de la VM ;
- si arg est un nombre (i), alors valeur est le nom du domaine de collision sur lequel l'interface ethi est atta-chée.
- si arg est une chaîne, alors c'est le nom d'une option de vstart et valeur est un argument de

cette option.

Exemple :

```
routeur[0]=HubDCB
# L'interface eth0 de la VM Routeur est attachee au domaine
# de collision  HubDCB
routeur[bridged]=true
# L'interface eth1 de la VM Routeur est attachee au bridge Docker
# pour communiquer avec l'hôte et acceder à Internet

sta1[0]=HubDCB
# L'interface eth0 de la VM sta1 est attachee au domaine
# de collision  HubDCB
```

Le fichier **lab.conf** peut également contenir des paramètres optionnels informatifs qui sont affichés dans la console de la VM au démarrage du lab :*

```
LAB_DESCRIPTION="Premier exemple de reseau "
LAB_VERSION=1.0
LAB_AUTHOR="Charles Techer"
LAB_EMAIL= techer.charles@educ-valadon-luimoges.fr
LAB_WEB=http://www.educ-valadon-limoges.fr
```

Il est possible de visualiser la liste des machines lancées :

```
btssio@ubuntudocker:~/Documents/lab1$ kathara linfo
```

Il est possible de préciser la liste des machines à démarrer au lancement du lab.

```
btssio@ubuntudocker:~/Documents/lab1$kathara lstart routeur
```

Les sous-répertoires du lab

Toute arborescence mise sur le lab, dans le répertoire d'une VM est recopié dans la VM en partant de la racine lors du lancement du lab. Par exemple, l'arborescence **/etc/resolv.conf** mise dans le répertoire **sta1** du lab **lab1**, sera recopiée dans la VM **sta1** à l'endroit **/etc/resolv.conf**. Cette copie de fichiers pré renseignés, permet de configurer les VMs lors de leur lancement. Cela complète l'utilisation des fichiers **VMx.startup** et **VMx.shutdown**. Les fichiers **shared.startup** et **shared.shutdown** concernent toutes les VMs du lab et sont exécutés avant **VMx.startup** et **VMx.shutdown**.

Récapitulons :

- Kathara crée un conteneur Docker pour chaque VM indiquée dans **lab.conf** ;
- Le contenu du répertoire d'une VM est copié dans l'arborescence de la VM correspondante ;
- Les fichiers **shared.startup** et **shared.shutdown** sont exécutés ;
- le fichier **VMx.startup** est exécuté au lancement de la VM ;
- le fichier **VMx.shutdown** est exécuté à l'arrêt de la VM.

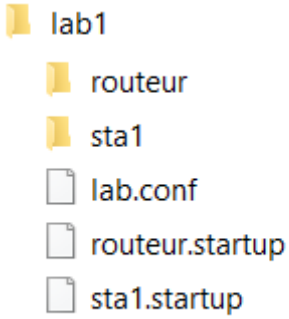
IMPORTANT
A **l'arrêt** du lab, toutes les **modifications** effectuées dans les VMs sont **perdues**. Comme vous accédez au dossier Home de BTSSIO de l'hôte Ubuntu depuis les VMs, faites une **sauvegarde** des fichiers de configuration modifiés des VMS dans les dossiers correspondant des VMs dans le dossier du lab.

Toute la configuration des VMs du lab (création des interfaces, routage, masquage d'adresse, installation de paquets, modifications des fichiers de configuration...) peut être **mis en place par les scripts .startup** et par les fichiers créés dans l'arborescence du lab. Tout cela est mis en place au lancement du lab.

Le lab complet

Voici le lab complet pour le réseau constitué avec les VMs routeur et sta1.

L'arborescence des dossiers :



Le fichier $HOME/Documents/lab/lab.conf

```
LAB_DESCRIPTION="Premier exemple de lab "
LAB_VERSION=1.0
LAB_AUTHOR="Techer"
LAB_EMAIL= techer.charles@educ-valadon-luimoges.fr
LAB_WEB=http://sio.educ-valadon-limoges.fr

routeur[0]=HubDCB
# interface eth0 de routeur est attachee au domaine de collision HubDCB

routeur[bridged]=true
# interface eth1 de la VM Routeur attachee au bridge Docker
# autoconfiguration par le bridge Docker


Sta1[0]=HubDCB
# L interface eth0 de sta1 est attachee au domaine de collision  HubDCB
```

Le fichier $HOME/ Documents/lab1/routeur.startup

```
# configuration de la carte eth0
ifconfig eth0 192.168.10.254 netmask 255.255.255.0 up

# Activer l ipmasquerade
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Le fichier $HOME/ Documents/lab1/sta1.startup

```
ifconfig eth0 192.168.10.1 netmask 255.255.255.0 up
route add default gw 192.168.10.254
```

Générateur de lab Kathara

Le projet Kathara propsoe un générateur de lab disponible sur Github :

- https://github.com/KatharaFramework/Netkit-Lab-Generator

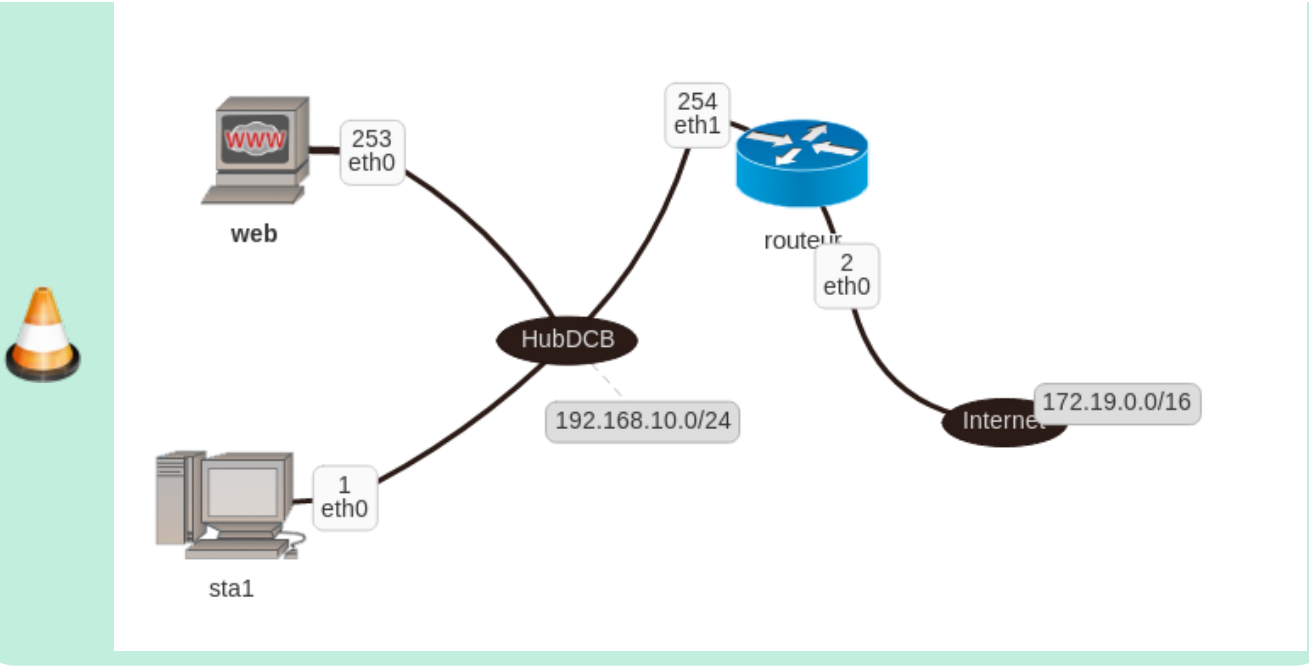
Exercice 1

EXERCICE 1
Modifiez le lab afin de rajouter un serveur **Web** avec les caractéristiques suivantes :

- le nom de la VM est web et elle est dans le réseau de sta1 ;
- Adresse IP de la VM web : 192.168.10.253/24 ;
- Ce serveur web doit fournir une page html qui aura le contenu de votre choix.

Pour vous aider :

- Dans l'arborescence du lab, vous pouvez indiquer les fichiers qui sont à recopier dans l'arborescence de la VM ;
- Les pages html du serveur web doivent être dans le dossier **/var/www/html** ;
- Le lancement du service web se fait avec la commande **service apache2 start** ;
- Le navigateur **links** peut être utilisé depuis sta1 pour tester le bon fonctionnement du serveur web.



Communiquer avec les VMs Kathara

Vos VMs peuvent communiquer sur le réseau mais sont **masquées** et donc **inaccessibles depuis votre réseau local**. Il est tout à fait envisageable de réaliser des infrastructures réseaux qui utilisent à la fois vos VMs crées avec Kathara et des VMs créées avec Virtualbox. Toutes ces VMs doivent pouvoir communiquer entre elles.

Le bridge (pont) Kathara créé en associant un domaine de collision à une des interfaces de la VM routeur de l'exercice précédent, pernet à celle-ci de communiquer avec votre hôte Ubuntu, votre réseau local et Internet. Kathara à configuré automatiquement l'interface de la VM et depuis votre hôte Ubuntu vous pouvez faire un ping sur l'interface eth1 de routeur (adresse IP 172.17.0.2/16).

Par contre vous ne pouvez pas faire de ping réussis sur l'interface eth0 (adresse IP 192.168.10.254/24) ni sur les interfaces de **sta1** (adresse IP 192.168.10.1/24) et de **web** (adresse IP 192.168.10.253/24).

Vous pouvez donc rajouter une route statique à l'hôte Ubuntu afin de lui indiquer comment accéder aux réseaux créés après la VM Routeur.

Voici la commande à exécuter dans le Terminal de la VM Ubuntu :

```
btssio@ubuntunetkit:~$   sudo route add –net 192.168.10.0 netmask
255.255.255.0 gw 172.17.0.2
```

ATTENTION
Indiquez bien l'adresse de l'**interface eth1** de la VM routeur comme passerelle (**gw**) dans la commande route.

La commande est à interpréter de la manière suivante :

Pour atteindre le réseau 192.168.10.0 (celui sur lequel se trouve les VM **sta1 et web**), il faut envoyer les paquets à l'équipement qui a l'adresse IP 172.17.0.2 et qui va router les paquets : il s'agit de l'interface de la VM Routeur.

Visualisez à nouveau la table de routage d'Ubuntu :

```
btssio@ubuntudocker:~/Documents/lab_exercice1$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
0.0.0.0         192.168.1.254     0.0.0.0         UG    100    0        0
enp0s3
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0
docker0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0
enp0s3
192.168.1.254   0.0.0.0         255.255.255.255 UH    100    0        0
enp0s3
192.168.10.0    172.17.0.2      255.255.255.0   UG    0      0        0
docker0
btssio@ubuntudocker:~/Documents/lab_exercice1$
```

Une nouvelle route est ajoutée vers 192.168.10.0 en passant par le bridge virtuel **docker0**. Maintenant, à partir de l'hôte Ubuntu Server, vous pouvez communiquer avec le réseau **192.168.10.0** et donc avec **sta1** (adresse IP 192.168.10.1) et **web** (adresse IP 192.168.10.253).

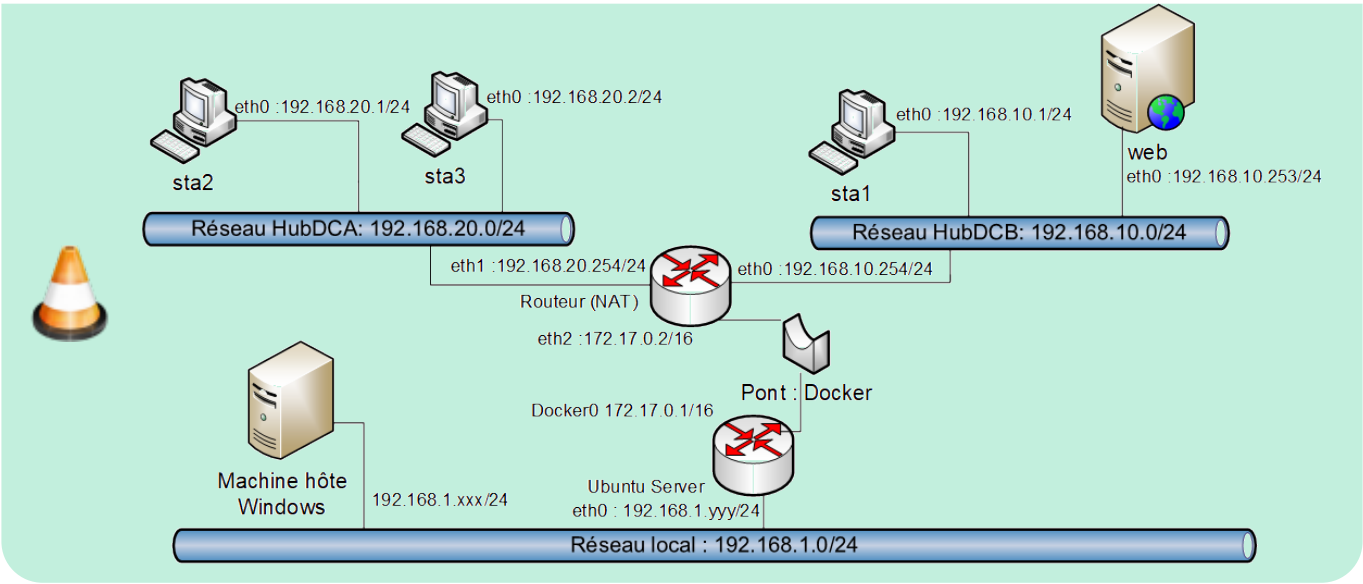
```
btssio@ubuntudocker:~/Documents/lab_exercice1$ ping -c 4 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=63 time=0.051 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=63 time=0.050 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=63 time=0.049 ms
64 bytes from 192.168.10.1: icmp_seq=4 ttl=63 time=0.050 ms

--- 192.168.10.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.049/0.050/0.051/0.000 ms
btssio@ubuntudocker:~/Documents/lab_exercice1$
```

Cette route statique sera supprimée lors de l'arrêt du lab ou du redémarrage de la VM Ubuntu server. La faudra la recréer à ce moment-là.

EXERCICE 2

Complétez le lab pour rajouter le réseau 192.168.20.0/24 contenant les VMs sta2 et sta3 et permettre la communication entre les toutes les VMs de ce réseau (sta1, sta2, sta3, web) et l'hôte Ubuntu server.



Retour Accueil SISR3

- SISR3