Routage inter-VLAN dans un contexte IaaS

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

Le routage inter-VLAN est très largement utilisé dans l'interconnexion entre les réseaux Ethernet. Les manipulations présentées dans ces travaux pratiques illustrent l'interconnexion entre un réseau appartenant à une infrastructure de type IaaS (Infrastructure as a Service) et un réseau d'hébergement de conteneurs Incus raccordés à l'aide de la technologie `macvlan`.

On introduit aussi un premier niveau de filtrage qui assure la traduction d'adresses entre les deux réseaux interconnectés.

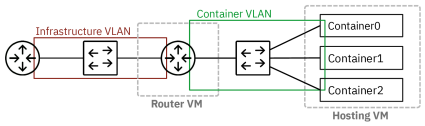


Table des matières

1. Copyright et Licence

Méta-information

Ce document est écrit avec DocBook XML sur un système Debian GNU/Linux. Il est disponible en version imprimable au format PDF : `inter-vlan-iaas.pdf`.

2. Topologies logiques et virtuelles

Les définitions importantes sur les réseaux locaux virtuels et le routage associé sont présentées dans l'article Routage Inter-VLAN

On rappelle simplement que la notion de réseau local virtuel ou VLAN permet de constituer des groupes logiques dans les réseaux Ethernet au niveau liaison de la modélisation. Lors du raccordement entre

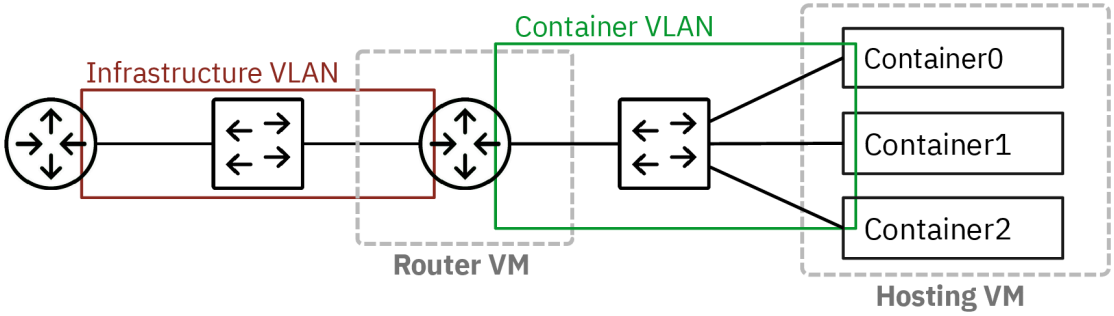
les équipements (commutateurs, routeurs, serveurs), certaines liaisons doivent véhiculer le trafic de plusieurs réseaux locaux virtuels (VLANs). Ces liaisons sont baptisées trunks dans le jargon. Pour distinguer le trafic appartenant à chaque réseau local, on ajoute à la trame une balise définie par le standard IEEE 802.1Q. C'est cette étiquetage de trame qui permet la distribution des domaines de diffusion entre plusieurs équipements physiques distincts.

On atteint ainsi un objectif très important. Il est possible de concevoir une topologie logique de réseau totalement indépendante de la topologie physique.

Réseau virtuel ou pas, il ne faut pas oublier les éléments suivants sur la segmentation des réseaux locaux.

- Une interface de ***commutateur*** délimite un domaine de ***collision***.
- Une interface de ***routeur*** délimite à la fois un domaine de ***collision*** et un domaine de ***diffusion***.

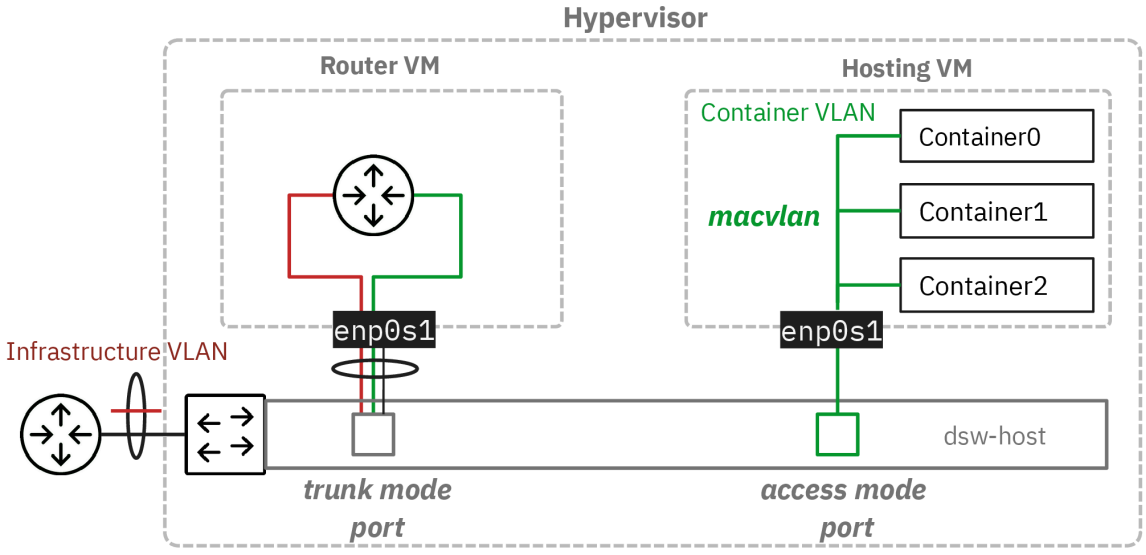
La représentation de la topologie logique ci-dessous montre que le routeur de couleur verte assure l'interconnexion entre un réseau d'infrastructure appelé Hosting VLAN et un réseau de conteneurs appelé Container VLAN. Les deux rectangles en gris “matérialisent“ les machines virtuelles qui sont utilisées pour les manipulations.



Topologie logique

La représentation de la topologie vue sous l'angle de l'hébergement sur un système hôte hyperviseur montre que les deux VLANs sont présents sur le commutateur virtuel de couche distribution appelé dsw-host. Ce commutateur appartient au système hôte. Il assure le raccordement entre les réseaux physiques et virtualisés. On retrouve le routeur de couleur verte raccordé avec un lien unique sur lequel le trafic des deux VLANs doit transiter.

Côté conteneurs, seule l'interface `enp0s1` est raccordé via un lien en mode accès. La technologie macVLAN permet d'utiliser plusieurs adresses MAC sur une même interface réseau.



Topologie hébergée

Tableau 1. Plan d'adressage des réseaux de la maquette

Réseau	Numéro VLAN	Adresses de passerelles
Hébergement VLAN rouge	360	`192.168.104.129/29` `2001:678:3fc:168::1/64`
Services VLAN vert	440	`192.0.2.1/24` `fda0:7a62:1b8::1/64`

3. Raccordement au commutateur de distribution

Dans cette section, on étudie le raccordement des deux machines virtuelles au commutateur de distribution sur le système hôte.

Q1. Comment contrôler la configuration des ports du commutateur de distribution sur le système hôte ?

Le commutateur virtuel implanté sur le système hôte est géré par Open vSwitch. On fait donc appel à la commande ovs-vsctl pour accéder aux paramètres de la configuration des ports.

- Pour le port nommé `tap200`, on obtient le paramètre `vlan_mode` avec l'instruction :

```
sudo ovs-vsctl get port tap200 vlan_mode
trunk
```

Le mode `trunk` correspond à un canal de transmission unique dans lequel circule le trafic de plusieurs domaines de diffusion ou VLANs.

- Pour le port nommé `tap2`, on obtient la valeur `access` pour le même paramètre :

```
sudo ovs-vsctl get port tap2 vlan_mode
access
```

Ici, le mode `access` correspond à un canal de transmission dans lequel circule le trafic d'un seul et unique domaine de diffusion ou VLAN.

Q2. Comment afficher le numéro de VLAN attribué au port en mode accès du commutateur de distribution sur le système hôte ?

On reprend la même commande que dans la question précédente avec le mot clé `tag`.

```
sudo ovs-vsctl get port tap2 tag
20
```

Q3. Comment affecter le numéro de VLAN attribué au port en mode accès du commutateur de distribution sur le système hôte ?

On reprend à nouveau la même commande avec l'option `set`.

```
sudo ovs-vsctl set port tap2 tag=440
```

Les valeurs données dans l'exemple ci-dessus sont à changer suivant les attributions du plan d'adressage des réseaux d'hébergement et de conteneurs.

Q4. Comment configurer les ports du commutateur avant le lancement des machines virtuelles ?

On utilise le script de procédure `switch-conf.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git startup-scripts.

Voici une copie du fichier de configuration des deux ports de commutateur.

```
ovs:
  switches:
    - name: dsw-host
      ports:
        - name: tap5 # Router port
          type: OVSPort
          vlan_mode: trunk
          trunks: [360, 440]
        - name: tap6 # Container hosting VM
          type: OVSPort
          vlan_mode: access
          tag: 440
```

On applique les paramètres définis ci-dessus.

```
$HOME/masters/scripts/switch-conf.py switch.yaml
```

On obtient les résultats suivants.

```
----------------------------------------
Configuring switch dsw-host
>> Port tap5 vlan_mode set to trunk
>> Port tap5 trunks set to [360, 440]
>> Port tap6 vlan_mode is already set to access
>> Port tap6 tag set to 440
----------------------------------------
```

Les numéros de port de commutateur et de VLAN donnés dans les exemples ci-dessus sont à changer suivant le contexte.

Q5. Comment lancer les machines virtuelles associées aux rôles routeur et hébergement de conteneurs ?

On utilise le script de procédure `lab-startup.py` qui applique les déclarations contenues dans un fichier YAML. Le code du script est accessible à partir du dépôt Git startup-scripts.

Voici une copie du fichier de déclaration des deux machines virtuelles.

```
kvm:
  vms:
    - vm_name: router
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 5
    - vm_name: hosting
      master_image: debian-testing-amd64.qcow2 # master image to be used
      force_copy: false # do not force copy the master image to the VM image
      memory: 1024
      tapnum: 6
```

On lance les deux machines virtuelles avec le script lab-startup.py.

```
$HOME/masters/scripts/lab-startup.py lab1.yaml
```

```
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to router.qcow2...done
Creating OVMF_CODE.fd symlink...
Creating router_OVMF_VARS.fd file...
Starting router...
~> Virtual machine filename   : router.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5905
~> telnet console port number : 2305
~> MAC address                : b8:ad:ca:fe:00:05
~> Switch port interface      : tap5, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:5%dsw-host
router started!
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to hosting.qcow2...done
Creating hosting_OVMF_VARS.fd file...
Starting hosting...
~> Virtual machine filename   : hosting.qcow2
~> RAM size                   : 1024MB
~> SPICE VDI port number      : 5906
~> telnet console port number : 2306
~> MAC address                : b8:ad:ca:fe:00:06
~> Switch port interface      : tap6, access mode
~> IPv6 LL address            : fe80::baad:caff:fefe:6%vlan440
hosting started!
```

Les deux machines virtuelles sont maintenant disponibles pour la suite des manipulations.

4. Rôle routeur

Dans cette section, on étudie la machine virtuelle qui joue le rôle de routeur entre le réseau d'hébergement et un réseau de conteneurs. Pour traiter les questions, il est nécessaire de mettre en œuvre une maquette avec un adressage indépendant. Voici les choix effectués pour la maquette.

4.1. Configuration des interfaces du routeur

Une fois la machine virtuelle routeur lancée, les premières étapes consistent à lui attribuer un nouveau nom et à configurer les interfaces réseau pour joindre les hôtes voisins.

Q6. Comment changer le nom de la machine virtuelle ?

Il faut éditer le fichier `/etc/hostname` en remplaçant le nom local par le nom voulu. Il est ensuite nécessaire de redémarrer pour que le nouveau nom soit pris en compte par tous les outils du système.

```
etu@localhost:~$ echo router | sudo tee /etc/hostname
etu@vm0:~$ sudo reboot
```

Q7. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface du routeur ?

Consulter la documentation de Netplan pour obtenir les informations sur la configuration des interfaces réseau à l'adresse Netplan documentation.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise Netplan dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration de base fournie avec l'image maître suppose que l'interface obtienne un bail DHCP pour la partie IPv4 et une configuration automatique via SLAAC pour la partie IPv6. Cette configuration par défaut doit être éditée et remplacée. Il faut configurer trois interfaces.

- L'interface principale correspond à l'interface "physique" de la machine. Elle est nommée `enp0s1` en fonction de l'ordre des adresses des composants raccordés au bus PCI.

- Une sous-interface doit être créée pour le réseau d'hébergement avec le numéro de VLAN désigné dans le plan d'adressage des réseaux d'hébergement et de conteneurs. Cette interface doit désigner les passerelles IPv4 et IPv6 de façon à joindre l'Internet.

- Une sous-interface doit être créée pour le réseau des conteneurs avec, là encore, le bon numéro de VLAN. Les adresses IPv4 et IPv6 de cette interface deviendront les passerelles du serveur et des conteneurs.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` de la maquette.

```
network:
    version: 2
    ethernets:
      enp0s1:
        dhcp4: false
        dhcp6: false
        accept-ra: false
        nameservers:
          addresses:
            - 172.16.0.2
            - 2001:678:3fc:3::2

    vlans:
      enp0s1.360:
        id: 360
        link: enp0s1
        addresses:
          - 192.168.104.130/29
          - 2001:678:3fc:168::82/64
        routes:
          - to: default
            via: 192.168.104.129
          - to: "::/0"
            via: "fe80::168:1"
            on-link: true
      enp0s1.440:
        id: 440
        link: enp0s1
        addresses:
          - 192.0.2.1/24
          - fda0:7a62:1b8::1/64
```

Une fois le fichier de configuration en place, il suffit de faire appel à la commande netplan pour appliquer les changements.

```
sudo netplan apply
```

Pour vérifier que l'adressage réseau est correct, on dispose de plusieurs solutions. Voici un exemple avec la commande networkctl qui synthétise l'ensemble de la configuration réseau.

```
networkctl status
```

```
# Interfaces: 1, 2, 3, 4
    State: routable
Online state: online
  Address: 192.168.104.130 on enp0s1.360
           192.0.2.1 on enp0s1.440
           2001:678:3fc:168::82 on enp0s1.360
           2001:678:3fc:168:baad:caff:fefe:5 on enp0s1.360
           fda0:7a62:1b8::1 on enp0s1.440
           fe80::baad:caff:fefe:5 on enp0s1
           fe80::baad:caff:fefe:5 on enp0s1.360
           fe80::baad:caff:fefe:5 on enp0s1.440
  Gateway: 192.168.104.129 on enp0s1.360
           fe80:168::1 on enp0s1.360
      DNS: 172.16.0.2
           2001:678:3fc:3::2
```

Q8. Quels sont les tests de connectivité réalisables après application de la nouvelle configuration des interfaces réseau ?

Relever l'état des trois interfaces et procédez aux tests ICMP et DNS en respectant l'ordre des couches de la modélisation.

Sans la confirmation que la configuration du serveur de conteneurs est prête, c'est du côté hébergement et accès Internet qu'il faut orienter les tests. Classiquement, on cherche à joindre la passerelle en premier puis l'Internet ensuite via des requêtes ICMP. Enfin, on effectue un test de couche application avec une requête DNS.

```
ping -q -c2 192.168.104.129
PING 192.168.104.129 (192.168.104.129) 56(84) bytes of data.

--- 192.168.104.129 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.501/1.516/1.531/0.015 ms
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 49.304/52.098/54.892/2.794 ms
```

```
ping -q -c2 fe80:168::1%enp0s1.360
PING fe80:168::1%enp0s1.360(fe80:168::1%enp0s1.360) 56 data bytes

--- fe80:168::1%enp0s1.360 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 1.459/13.305/25.152/11.846 ms
```

```
ping -q -c2 2620:fe::fe
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.812/42.437/43.063/0.625 ms
```

```
host quad9.net
quad9.net has address 216.21.3.77
quad9.net has IPv6 address 2620:0:871:9000::77
quad9.net mail is handled by 5 mx1.quad9.net.
quad9.net mail is handled by 20 mx2.quad9.net.
quad9.net mail is handled by 100 keriomail.pch.net.
```

4.2. Activation de la fonction routage

Sans modification de la configuration par défaut, un système GNU/Linux n'assure pas la fonction de routage du trafic d'une interface réseau à une autre.

L'activation du routage correspond à un réglage de paramètres du sous-système réseau du noyau Linux. L'outil qui permet de consulter et modifier les réglages de paramètre sur le noyau est appelé sysctl.

Q9. Comment activer le routage dans le sous-système réseau du noyau Linuxe ?

Utiliser la commande sysctl pour effectuer des recherches et identifier les paramètres utiles. Par exemple :

```
sudo sysctl -a -r ".*forward.*"
```

Il est dorénavant recommandé de créer un fichier de configuration spécifique par fonction. C'est la raison pour laquelle on crée un nouveau fichier `/etc/sysctl.d/10-routing.conf`.

Attention ! Il ne faut pas oublier d'appliquer les nouvelles valeurs des paramètres de configuration.

```
cat << EOF | sudo tee /etc/sysctl.d/10-routing.conf
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv4.conf.all.log_martians=1
EOF
```

Voici un exemple des résultats obtenus après application des nouveaux paramètres.

```
sudo sysctl --system
```

```
* Applique /usr/lib/sysctl.d/10-coredump-debian.conf …
* Applique /etc/sysctl.d/10-routing.conf …
* Applique /usr/lib/sysctl.d/50-default.conf …
* Applique /usr/lib/sysctl.d/50-pid-max.conf …
* Applique /etc/sysctl.conf …
kernel.core_pattern = core
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv4.conf.all.log_martians = 1
kernel.sysrq = 0x01b6
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.enp0s1/360.rp_filter = 2
net.ipv4.conf.enp0s1/440.rp_filter = 2
net.ipv4.conf.enp0s1.rp_filter = 2
net.ipv4.conf.lo.rp_filter = 2
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.enp0s1/360.accept_source_route = 0
net.ipv4.conf.enp0s1/440.accept_source_route = 0
net.ipv4.conf.enp0s1.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.enp0s1/360.promote_secondaries = 1
net.ipv4.conf.enp0s1/440.promote_secondaries = 1
net.ipv4.conf.enp0s1.promote_secondaries = 1
net.ipv4.conf.lo.promote_secondaries = 1
net.ipv4.ping_group_range = 0 2147483647
net.core.default_qdisc = fq_codel
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.protected_regular = 2
fs.protected_fifos = 1
kernel.pid_max = 4194304
```

Q10. Quelles sont les conditions à réunir pour tester le fonctionnement du routage ?

Rechercher comment utiliser l'analyseur réseau tshark pour caractériser l'acheminement du trafic d'un réseau à l'autre.

Le plan d'adressage prévoit d'utiliser des préfixes ayant une portée locale pour les réseaux de conteneurs. Il n'est donc pas possible de passer par une requête ICMP pour caractériser l'accès aux réseaux distants. En effet, l'adresse source n'est pas reconnue par l'hôte distant et les routeurs de l'Internet ne disposent d'aucune solution pour joindre le réseau des conteneurs.

Voici un extrait de capture qui montre que le serveur de conteneur cherche à joindre un hôte sur l'Internet sans succès. Cette capture étant réalisée sur l'interface réseau côté hébergement, elle montre que le trafic est bien acheminé d'un réseau à l'autre.

```
tshark -i enp0s1.360
```

```
Capturing on 'enp0s1.360'
    1 0.000000000    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xbdab A 1.debian.pool.ntp.or
    2 0.000056361    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xab92 AAAA 1.debian.pool.ntp
```

4.3. Activation de la traduction d'adresses

Le résultat de la question ci-dessus montre que les hôtes situés dans le réseau des conteneurs ne peuvent pas joindre l'Internet puisque les préfixes réseau utilisés ont une portée limitée.

Q11. Quels sont les paquets qui fournissent les outils de gestion de la traduction d'adresses ?

Rechercher les paquets relatifs au filtrage et à la gestion des règles de pare-feux.

Dans le contexte des ces manipulations, nous utilisons `nftables` comme outil de gestion du filtrage.

C'est la partie outils de l'espace utilisateur qui nous intéresse ici.

```
apt search ^nftables$
nftables/testing,now 1.1.0-2 amd64  [installé]
  programme de contrôle des règles de filtrage de paquets du projet Netfilter
```

```
sudo apt -y install nftables
```

Q12. Quelles sont les règles à appliquer pour assurer une traduction des adresses sources en sortie sur le réseau d'infrastructure (VLAN rouge) ?

Rechercher dans des exemples de configuration `nftables` avec la fonction `MASQUERADE`.

Voici un exemple de création du fichier `/etc/nftables.conf` avec le jeu d'instructions qui assure la traduction d'adresses sources pour IPv4 et IPv6.

```
cat << EOF | sudo tee /etc/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp0s1.360" masquerade
    }
}
EOF
```

Avertissement

Il faut impérativement changer le nom d'interface en utilisant le numéro de VLAN attribué dans le plan d'adressage des travaux pratiques.

La création de ce fichier de règles n'est pas suffisante. Il faut appliquer les règles contenues dans le fichier.

```
sudo nft -f /etc/nftables.conf
```

Q13. Comment rendre le chargement des règles de filtrage automatique au redémarrage du système ?

Afficher l'état du service `nftables.service`. Activer ce service si celui est à l'état désactivé (disabled).

Pour afficher l'état du service, on utilise la commande suivante.

```
systemctl status nftables.service
```

```
# nftables.service - nftables
     Loaded: loaded (/usr/lib/systemd/system/nftables.service; disabled; preset: enabled)
     Active: inactive (dead)
       Docs: man:nft(8)
             http://wiki.nftables.org
```

On constate qu'il faut activer ce service pour assurer le chargement automatique des règles de filtrage au démarrage.

```
sudo systemctl enable nftables.service
sudo systemctl start nftables.service
sudo systemctl status nftables.service
```

```
# nftables.service - nftables
     Loaded: loaded (/usr/lib/systemd/system/nftables.service; enabled; preset: enabled)
     Active: active (exited) since Sat 2024-09-14 18:35:00 CEST; 7s ago
 Invocation: 1dcc395a33c74606bd7dab7f33b90787
       Docs: man:nft(8)
             http://wiki.nftables.org
    Process: 729 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited, status=0/SUCCESS)
   Main PID: 729 (code=exited, status=0/SUCCESS)
   Mem peak: 5.9M
        CPU: 44ms

sept. 14 18:34:59 router systemd[1]: Starting nftables.service - nftables...
sept. 14 18:35:00 router systemd[1]: Finished nftables.service - nftables.
```

Q14. Comment caractériser le fonctionnement de la traduction d'adresses sources ?

Rechercher dans les pages de manuel de la commande nftables les options d'affichage du décompte du trafic traité.

Voici un exemple d'affichage des règles actives avec visualisation des compteurs d'utilisation.

```
sudo nft list ruleset
```

```
table ip nat {
 chain postrouting {
  type nat hook postrouting priority srcnat; policy accept;
  oifname "enp0s1.360" counter packets 17 bytes 1284 masquerade
 }
}
table ip6 nat {
 chain postrouting {
  type nat hook postrouting priority srcnat; policy accept;
  oifname "enp0s1.360" counter packets 13 bytes 1220 masquerade
 }
}
```

4.4. Activation de l'adressage automatique pour le réseau de conteneurs

Dans le but mettre en place un adressage automatique des conteneurs hébergés sur l'autre machine virtuelle, on utilise l'outil `dnsmasq`. L'idée est de fournir un service DHCPv4 et SLAAC en un seul et unique fichier de configuration.

On débute par l'installation du paquet.

```
sudo apt -y install dnsmasq
```

Q15. Comment remplacer le fichier de configuration fourni lors de l'installation du paquet par notre propre fichier de configuration ?

Consulter le contenu du fichier `/etc/dnsmasq.conf` et extraire les options de configuration utiles au contexte de ces manipulations.

Voici la commande de copie du fichier issu de l'installation.

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.dist
```

Voici un exemple de configuration adaptée à la maquette.

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=enp0s1.440

# Enable DHCPv4 on Container VLAN
dhcp-range=192.0.2.20,192.0.2.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:enp0s1.440,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[260:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
bind-interfaces
listen-address=192.0.2.1
EOF
```

Avertissement

Il faut impérativement changer le numéro de VLAN ainsi que les adresses IPv4 de l'exemple ci-dessus par les informations données dans le plan d'adressage des travaux pratiques.

De plus, une fois le fichier créé, il ne faut pas oublier de redémarrer le service et de contrôler l'état de son fonctionnement.

```
sudo systemctl restart dnsmasq
systemctl status dnsmasq
```

```
# dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
     Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-14 18:49:02 CEST; 10min ago
 Invocation: 8887da64fc36432db7be668ec71cbfb7
    Process: 1072 ExecStartPre=/usr/share/dnsmasq/systemd-helper checkconfig (code=exited, status
    Process: 1077 ExecStart=/usr/share/dnsmasq/systemd-helper exec (code=exited, status=0/SUCCESS
    Process: 1084 ExecStartPost=/usr/share/dnsmasq/systemd-helper start-resolvconf (code=exited,
   Main PID: 1083 (dnsmasq)
      Tasks: 1 (limit: 1087)
     Memory: 720K (peak: 2.8M)
        CPU: 94ms
     CGroup: /system.slice/dnsmasq.service
     └─1083 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -r /run/dnsmasq/resolv.conf \
       -7 /etc/dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg-new --local-service
       --trust-anchor=.,20326,8,2,e06d44b80b8f1d39a95c0b0d7c65d08458e880409bbc683457104237c7f8ec8d

sept. 14 18:49:02 router systemd[1]: Started dnsmasq.service - dnsmasq - A lightweight DHCP and c
```

5. Rôle serveur de conteneurs

5.1. Configuration de l'interface du serveur

Une fois la machine virtuelle serveur de conteneurs lancée, les premières étapes consistent à lui attribuer un nouveau nom et à configurer les interfaces réseau pour joindre le routeur voisin et l'Internet.

Q16. Comment changer le nom de la machine virtuelle ?

Il faut créer le fichier `/etc/hostname` avec le nom d'hôte voulu.

```
echo hosting | sudo tee /etc/hostname
sudo reboot
```

Q17. Comment appliquer les configurations réseau IPv4 et IPv6 à partir de l'unique interface de la machine d'hébergement ?

Consulter la documentation de Netplan pour obtenir les informations sur la configuration des interfaces réseau à l'adresse Netplan documentation.

Il existe plusieurs possibilités pour configurer une interface réseau. Dans le contexte de ces manipulations, on utilise Netplan dans le but de séparer la partie déclarative du moteur de configuration.

C'est `systemd-networkd` qui joue le rôle de moteur de configuration sur les machines virtuelles utilisées avec ces manipulations.

La configuration de base fournie avec l'image maître suppose que l'interface obtienne un bail DHCP pour la partie IPv4 et une configuration automatique via SLAAC pour la partie IPv6. Cette configuration par défaut doit être éditée et remplacée.

- L'interface réseau appartient un seul et unique domaine de diffusion : le VLAN 440 dans le contexte de cette maquette.

- On fait le choix d'attribuer des adresses statiques à l'interface `enp0s1` pour être en mesure de tester le routage et la traduction d'adresses sources au niveau du routeur. Ainsi, on ne dépend pas du service `dnsmasq` pour les tests de communication.

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` de la maquette.

```
cat << EOF | sudo tee /etc/netplan/enp0s1.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 192.0.2.2/24
        - fda0:7a62:1b8::2/64
      routes:
        - to: default
          via: 192.0.2.1
        - to: "::/0"
          via: fe80::baad:caff:fefe:5
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
```

Bien sûr, il ne faut pas oublier d'appliquer les paramètre de configuration de l'interface.

```
sudo netplan apply
```

Q18. Comment valider la configuration réseau du serveur de conteneurs ?

Lancer une série de tests ICMP IPv4 et IPv6.

On reprend les tests usuels avec les commandes ping et host.

```
etu@server:~$ ping -q -c2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 19.433/19.586/19.740/0.153 ms
```

```
etu@server:~$ ping -q -c2 2620:fe::fe
PING 2620:fe::fe(2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 42.983/43.114/43.246/0.131 ms
```

```
etu@server:~$ host kernel.org
kernel.org has address 139.178.84.217
kernel.org has IPv6 address 2604:1380:4641:c500::1
kernel.org mail is handled by 10 smtp1.kernel.org.
kernel.org mail is handled by 10 smtp3.kernel.org.
kernel.org mail is handled by 10 smtp2.kernel.org.
```

5.2. Installation du gestionnaire de conteneurs Incus

Sur l'hôte qui tient le rôle de serveur d'hébergement, la gestion des conteneurs est confiée à Incus.

Selon l'exergue du site, Incus est un gestionnaire de conteneurs et de machines virtuelles puissant, sûr et moderne.

Dans le contexte de ces manipulations, c'est la variété des modes d'interconnexion réseau offerte par Incus qui est le point le plus déterminant. Ici on utilise le mode macvlan qui raccorde les conteneurs au même domaine de diffusion que la machine hôte. C'est le mode de raccordement le plus simple. Cependant, il n'autorise pas les communications entre les conteneurs du fait de l'isolation des espaces de noms.

Q19. Comment installer le gestionnaire de conteneurs Incus ?

Lancer une recherche dans la liste des paquets Debian.

Le paquet s'appelle tout simplement incus.

```
apt search ^incus
```

```
sudo apt -y install incus
```

Q20. Comment faire pour que l'utilisateur normal `etu` devienne administrateur et gestionnaire des conteneurs ?

Rechercher le nom du groupe système correspondant à l'utilisation des outils Incus.

Il faut que l'utilisateur normal appartienne au groupes systèmes `incus` et `incus-admin` pour qu'il ait tous les droits sur la gestion des conteneurs.

```
grep incus /etc/group
incus:x:990:
incus-admin:x:989:
```

```
sudo adduser etu incus
sudo adduser etu incus-admin
```

> **Avertissement**
>
> Attention ! Il faut se déconnecter/reconnecter pour bénéficier de la nouvelle attribution de groupe. On peut utiliser les commandes groups ou id pour vérifier le résultat.

```
groups
etu adm sudo users incus-admin incus
```

5.3. Configuration du gestionnaire de conteneurs Incus

Q21. Quelle est l'instruction de configuration initiale du gestionnaire Incus ?

Utiliser l'aide de la commande incus.

C'est l'instruction `incus admin init` qui nous intéresse.

Voici une copie d'écran de son exécution.

```
incus admin init
```

```
Would you like to use clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Would you like to create a new local network bridge? (yes/no) [default=yes]: no
Would you like to use an existing bridge or host interface? (yes/no) [default=no]: yes
Name of the existing bridge or host interface: enp0s1
Would you like the server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "init" preseed to be printed? (yes/no) [default=no]:
```

Q22. Quelle est l'instruction qui permet d'afficher le profil par défaut des conteneurs ?

Rechercher dans les options de la commande incus profile.

Voici un exemple d'exécution.

```
incus profile show default
```

```
config: {}
description: Default Incus profile
devices:
  eth0:
    name: eth0
    nictype: macvlan
    parent: enp0s1
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: default
used_by: []
project: default
```

L'affichage de ce profil par défaut permet de vérifier que tous les paramètres voulus sont correctement positionnés.

Q23. Quelle est l'instruction de création et de lancement de nouveaux conteneurs ?

Rechercher dans les options de la commande incus.

Tester son exécution avec un conteneur de type `debian/trixie`.

Voici un exemple d'exécution pour 3 nouveaux conteneurs.

```
for i in {0..2}; do incus launch images:debian/trixie c$i; done
```

```
Launching c0
Launching c1
Launching c2
```

```
incus ls
```

```
+------+---------+---------------------+----------------------------------------------+-----------+--
| NAME |  STATE  |        IPV4         |                    IPV6                      |   TYPE    | S
+------+---------+---------------------+----------------------------------------------+-----------+--
| c0   | RUNNING | 192.0.2.52 (eth0)   | fda0:7a62:1b8:0:216:3eff:febd:a233 (eth0) | CONTAINER | 0
+------+---------+---------------------+----------------------------------------------+-----------+--
| c1   | RUNNING | 192.0.2.133 (eth0)  | fda0:7a62:1b8:0:216:3eff:fe8e:cc62 (eth0) | CONTAINER | 0
+------+---------+---------------------+----------------------------------------------+-----------+--
| c2   | RUNNING | 192.0.2.187 (eth0)  | fda0:7a62:1b8:0:216:3eff:fe7d:6898 (eth0) | CONTAINER | 0
+------+---------+---------------------+----------------------------------------------+-----------+--
```

La copie d'écran ci-dessus montre que l'adressage automatique des conteneurs depuis le routeur a fonctionné.

Q24. Comment tester les communications réseau depuis chaque conteneur ?

Rechercher dans les options de la commande incus celle qui permet de lancer un traitement dans le conteneur.

C'est la commande incus exec qui correspond à notre besoin. Voici une exemple de boucle qui permet de lancer les tests ICMP IPv4 et IPv6 dans les 3 conteneurs actifs.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ping -qc2 9.9.9.9
    incus exec c$i -- ping -qc2 2620:fe::fe
done
```

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 27.887/28.676/29.465/0.789 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.727/48.212/54.697/6.485 ms
>>>>>>>>>>>>>>>>> c1
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 28.179/34.740/41.302/6.561 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 40.002/40.529/41.057/0.527 ms
>>>>>>>>>>>>>>>>> c2
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 28.539/35.408/42.278/6.869 ms
PING 2620:fe::fe (2620:fe::fe) 56 data bytes

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 40.428/47.517/54.606/7.089 ms
```

Q25. Comment exécuter des jeux d'instructions dans les conteneurs depuis le serveur d'hébergement ?

On entre ici dans le domaine de l'automatisation à l'aide de scripts Bash. Même si l'ambition reste très modeste, on peut développer un script qui utilise la liste des conteneurs actifs pour lancer une suite de traitements dans ces mêmes conteneurs.

Comme les conteneurs Incus appartiennent à la famille des conteneurs système, ils disposent d'une arborescence complète et d'une gestion de paquets. Allons y pour une mise à jour des paquets de chaque conteneur actif.

Voici un exemple de code qui stocke les commandes à lancer dans un tableau et qui les exécute sur chacun des conteneurs actifs.

```
#!/bin/bash

cmds=(
  "apt -y update"
  "apt -y full-upgrade"
  "apt -y clean"
  "apt -y autopurge"
)

clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

for c in $clist; do
  echo ">>>>>>>>>>>>>>>>> $c"
  for cmd in "${cmds[@]}"; do
    eval "incus exec $c -- $cmd"
  done
done

exit 0
```

Si ce script est enregistré dans le fichier `run-commands-in-containers.sh`, on peut lancer les traitements comme dans l'exemple ci-dessous.

```
bash run-commands-in-containers.sh
>>>>>>>>>>>>>>>>> c0
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
>>>>>>>>>>>>>>>>> c1
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
>>>>>>>>>>>>>>>>> c2
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie-updates InRelease
Hit:3 http://deb.debian.org/debian-security trixie-security InRelease
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

Pour compléter, voici une seconde version du script qui utilise les arguments de la ligne de commande pour remplir le tableau des commandes à exécuter.

```
#!/bin/bash

cmds=("$@")

clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

for c in $clist; do
  echo ">>>>>>>>>>>>>>>>> $c"
  for cmd in "${cmds[@]}"; do
    eval "incus exec $c -- $cmd"
  done
done
```

Q26. Comment passer d'un adressage automatique à un adressage statique pour chaque conteneur ?

Comme la gestion de la configuration des interfaces est assurée par `systemd-networkd`, il faut s'intéresser à la syntaxe du fichier `/etc/systemd/network/eth0.network` de chaque conteneur.

Cette question est un prétexte pour utiliser le transfert de fichier depuis le serveur d'hébergement vers les conteneurs.

Voici une liste des actions à réaliser sur tous les conteneurs actifs.

1. Installer le paquet netplan.io

2. Générer le fichier de déclaration YAML des paramètres de configuration réseau des interfaces `eth0`

3. Transférer le fichier de déclaration YAML dans le dossier `/etc/netplan/`

4. Effacer le fichier `/etc/systemd/network/eth0.network`

5. Appliquer la nouvelle configuration réseau

Pour connaître les paramètres de configuration réseau d'une interface de conteneur, on peut extraire le fichier `/etc/systemd/network/eth0.network` et consulter son contenu.

```
incus file pull c0/etc/systemd/network/eth0.network .
cat eth0.network
```

```
[Match]
Name=eth0

[Network]
DHCP=true

[DHCPv4]
UseDomains=true

[DHCP]
ClientIdentifier=mac
```

On vérifie ainsi que la configuration réseau issue de la source de tirage des conteneurs implique un adressage automatique au moins en IPv4. On propose donc de remplacer cet adressage automatique par un adressage statique.

Voici une proposition de script qui traite chacun des points définis dans la question.

```
#!/bin/bash

# Préparation -> générer la liste des conteneurs actifs
clist=$(incus list status=running -c n -f compact | grep -v NAME | tr '\n' ' ' | tr -s ' ')

# Étape 1 -> installer le paquet netplan.io
. run-commands-in-containers.sh "apt -y install netplan.io"

addr_idx=0
for c in $clist; do
  echo ">>>>>>>>>>>>>>>> $c"

# Étape 2 -> générer le fichier de configuration réseau YAML
$(cat << EOF > eth0.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 192.0.2.$((addr_idx + 10))/24
        - fda0:7a62:1b8::$(printf "%x" $((addr_idx + 10)))/64
      routes:
        - to: default
          via: 192.0.2.1
        - to: "::/0"
          via: fe80::baad:caff:fefe:5
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
)

# Étape 3 -> transférer le fichier de déclaration YAML
  incus file push eth0.yaml $c/etc/netplan/eth0.yaml

# Étape 4 -> effacer le fichier /etc/systemd/network/eth0.network
  incus exec $c -- rm /etc/systemd/network/eth0.network

# Étape 5 -> appliquer la nouvelle configuration
  incus exec $c -- netplan apply

  ((addr_idx++))
done

exit 0
```

Si le code du script ci-dessus est placé dans un fichier appelé `set-static-addressing.sh`, on peut l'exécuter directement et relever les résultats.

```
bash set-static-addressing.sh
```

```
incus restart --all
```

```
incus ls
+------+---------+-------------------+-----------------------------------------------+-----------+---
| NAME |  STATE  |       IPV4        |                     IPV6                      |   TYPE    | SN
+------+---------+-------------------+-----------------------------------------------+-----------+---
| c0   | RUNNING | 192.0.2.10 (eth0) | fda0:7a62:1b8::a (eth0)                       | CONTAINER | 0
|      |         |                   | fda0:7a62:1b8:0:216:3eff:febd:a233 (eth0)     |           |
+------+---------+-------------------+-----------------------------------------------+-----------+---
| c1   | RUNNING | 192.0.2.11 (eth0) | fda0:7a62:1b8::b (eth0)                       | CONTAINER | 0
|      |         |                   | fda0:7a62:1b8:0:216:3eff:fe8e:cc62 (eth0)     |           |
+------+---------+-------------------+-----------------------------------------------+-----------+---
| c2   | RUNNING | 192.0.2.12 (eth0) | fda0:7a62:1b8::c (eth0)                       | CONTAINER | 0
|      |         |                   | fda0:7a62:1b8:0:216:3eff:fe7d:6898 (eth0)     |           |
+------+---------+-------------------+-----------------------------------------------+-----------+---
```

6. Pour conclure

Le routage inter-VLAN joue un rôle essentiel dans les environnements de virtualisation et les réseaux en nuage modernes. Cette technique permet une segmentation logique efficace des réseaux, offrant une flexibilité et une sécurité accrues dans des infrastructures complexes. En permettant la communication entre différents VLANs via un routeur, elle facilite la création de topologies réseau évolutives et adaptables.

L'outil Incus se distingue par sa simplicité d'utilisation et sa flexibilité, offrant une gestion efficace des conteneurs et des machines virtuelles. Sa capacité à gérer divers modes d'interconnexion réseau, comme le mode macvlan utilisé dans ce document, en fait un choix idéal pour les administrateurs système cherchant à optimiser et automatiser leurs infrastructures.

Enfin, pour ce qui est de l'automatisation, les traitements “atomiques” proposés sont pertinents. Ce qui est nettement moins satisfaisant, c'est l'utilisation des boucles dans des scripts Bash. Si l'automatisation dépasse le cadre des manipulations proposées ici, il ne faut pas oublier la séparation entre l'inventaire des dispositifs (routeur, commutateurs, machines virtuelles, conteneurs) et les procédures est un pilier essentiel d'une automatisation efficace.