# Routage inter-VLAN et protocole PPPoE

**Philippe Latu**
philippe.latu(at)inetdoc.net

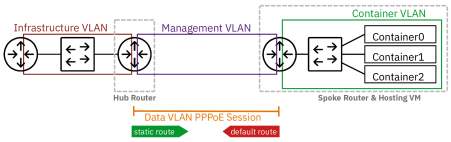
***https://www.inetdoc.net***

**Résumé**

La généralisation de l'utilisation de la fibre optique dans les réseaux étendus (WAN) jusqu'au raccordement domestique s'est accompagnée d'un changement important au niveau des liaisons de données. La technologie Ethernet est devenue universelle et couvre tous les besoins de commutation de circuits.

Cependant, pour raccorder les sites d'entreprises via des réseaux d'opérateurs, les fonctions historiques du protocole PPP (*Point-to-Point Protocol*) sont toujours utiles. C'est là que le protocole PPPoE intervient. Il permet d'associer un réseau de diffusion Ethernet avec un fonctionnement point à point typique des réseaux étendus.

Le but des manipulations présentées dans ce document est d'illustrer la mise en œuvre d'une session PPPoE entre un routeur virtuel central et un site distant factice (une autre machine virtuelle) qui héberge quelques services.




# Table des matières

1. Copyright et Licence ..... 2
2. Objectifs ..... 2
3. Interface Ethernet & protocole PPP ..... 3
4. Topologies logiques et virtuelles ..... 3
5. Raccordement au commutateur de distribution ..... 5
6. Routeur Hub ..... 7
    6.1. Configuration des interfaces du routeur ..... 7
    6.2. Activation de la fonction routage ..... 9
    6.3. Activation de la traduction d'adresses ..... 10
    6.4. Activation du service PPPoE ..... 11
7. Routeur Spoke ..... 16
    7.1. Configuration des interfaces du routeur ..... 16
    7.2. Activation de la fonction routage ..... 17
    7.3. Configurer le protocole PPP ..... 17
8. Réseau d'hébergement de conteneurs ..... 22
    8.1. Ajouter un commutateur virtuel ..... 22
    8.2. Routage du réseau d'hébergement ..... 24
    8.3. Adressage automatique dans le réseau d'hébergement ..... 26
9. Conteneurs système Incus ..... 27
    9.1. Installation du gestionnaire de conteneurs Incus ..... 27
    9.2. Configuration et lancement des conteneurs ..... 27
    9.3. Adressage statique des conteneurs ..... 30
10. Traces d'une ouverture de session PPPoE ..... 34
    10.1. Journaux du routeur Spoke ..... 34
    10.2. Journaux du routeur Hub ..... 35
11. Pour conclure... ..... 35

## 1. Copyright et Licence

Copyright (c) 2000,2026 Philippe Latu.
Permission is granted to copy, distribute and/or modify this document under the
terms of the GNU Free Documentation License, Version 1.3 or any later version
published by the Free Software Foundation; with no Invariant Sections, no
Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included
in the section entitled "GNU Free Documentation License".

Copyright (c) 2000,2026 Philippe Latu.
Permission est accordée de copier, distribuer et/ou modifier ce document selon
les termes de la Licence de Documentation Libre GNU (GNU Free Documentation
License), version 1.3 ou toute version ultérieure publiée par la Free Software
Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture,
et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est
incluse dans la section intitulée « Licence de Documentation Libre GNU ».

### Méta-information

Ce document est écrit avec **DocBook** XML sur un système **Debian GNU/Linux**. Il est disponible en version imprimable au format PDF : pppoe.pdf.

## 2. Objectifs

Après avoir réalisé les manipulations présentées dans ce document, les étudiants seront capables de :

1. Configurer et mettre en œuvre une topologie réseau complexe combinant routage inter-VLAN et protocole PPPoE.
2. Installer et configurer des conteneurs système Incus, y compris leur adressage réseau statique et dynamique.
3. Mettre en place et dépanner une session PPPoE entre deux routeurs virtuels avec les étapes d'authentification et de négociation du protocole.
4. Utiliser des outils d'automatisation comme les scripts Bash et Netplan pour configurer et gérer efficacement les interfaces réseau et les conteneurs.
5. Implémenter et tester la connectivité IPv4 et IPv6 dans un environnement réseau virtualisé, incluant la configuration de routes statiques et la traduction d'adresses.

## 3. Interface Ethernet & protocole PPP

Le format de trame historique HDLC est abandonné. Il faut dire que ce format de trame date du développement des liaisons séries asynchrones. Aujourd'hui, les liaisons sur fibres optiques sont *Full-Duplex* et on ne se préoccupe plus de synchronisation au niveau de la couche liaison de données. Le format de trame Ethernet devient ainsi une référence universelle.

Le protocole PPP offre depuis l'origine une configuration indépendante de la technologie du réseau étendu aussi bien au niveau de la couche liaison que de la couche réseau.

L'association entre trame Ethernet et PPP se fait grâce à un autre protocole baptisé PPPoE. Ce dernier permet d'encapsuler des trames PPP dans des trames Ethernet. Il est décrit à la page **Point-to-point protocol over Ethernet** qui permet de traiter les questions ci-après.

Q1 Quelle est la raison de l'ajout d'un nouveau protocole entre Ethernet et PPP ?

Consulter la page **Point-to-point protocol over Ethernet**.

---

Le protocole PPP, initialement destiné aux liaisons point à point, nécessite un mécanisme de découverte des extrémités pour fonctionner sur un réseau local Ethernet, qui est un réseau de diffusion où le canal de transmission est partagé entre tous les hôtes.

Q2 Quels sont les messages de découverte et de session PPPoE ? Préciser qui est l'initiative de la découverte.

Consulter la page **Point-to-point protocol over Ethernet**.

---

- Client to server: Initiation (PADI)
- Server to client: Offer (PADO)
- Client to server: request (PADR)
- Server to client: session-confirmation (PADS)
- Either end to other end: termination (PADT)

Q3 Quels sont les autres mécanismes de découverte de voisins connus dans un réseau local Ethernet ?

---

Voici la liste des «grands classiques».

- Address Resolution Protocol (ARP).

  Quelle est l'adresse MAC d'un hôte dont on connaît l'adresse IPv4 ?

- Neighbor Discovery Protocol (NDP).

  Ce protocole est associé à IPv6. Il définit 5 messages ICMPv6 qui couvrent les mêmes opérations que celles réalisées par le protocole ARP sans avoir recours à la diffusion et qui ajoutent de nouvelles fonctions.

- Multicast DNS (mDNS) ou *Bonjour*.

  Ce protocole entre dans la famille ***zeroconf*** qui a pour but d'annoncer et de fournir des éléments de configuration aux hôtes du réseau sans faire appel à une infrastructure de services de la couche application tels que DNS et DHCP.

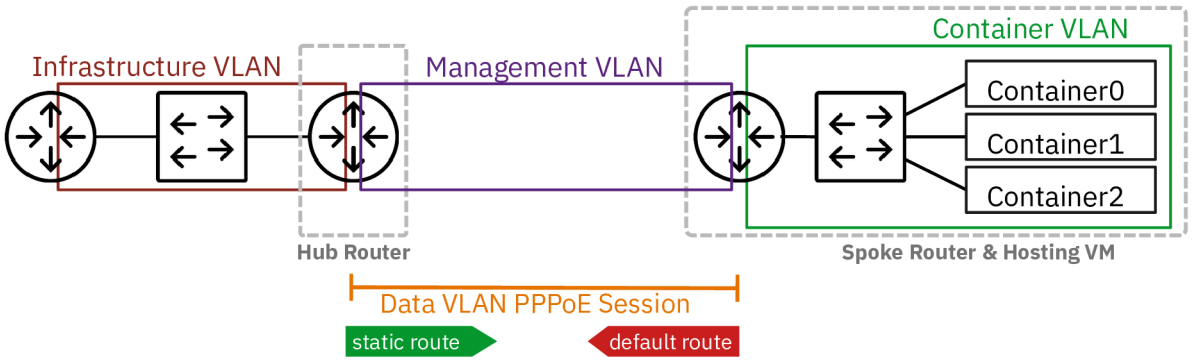
## 4. Topologies logiques et virtuelles

La représentation de la topologie logique ci-dessous montre deux routeurs. Le routeur placé à gauche assure l'interconnexion entre un réseau d'infrastructure (VLAN rouge) et le réseau opérateur (VLAN violet). Le routeur placé à droite assure l'interconnexion entre le même réseau opérateur et le réseau du site distant (VLAN vert). Les services hébergés sur le site distant sont compris dans le même réseau (VLAN vert).

Sur le réseau étendu factice, on distingue deux autres VLANs : le VLAN violet appelé **Management VLAN** est utilisé pour la supervision et le VLAN orange appelé **Data VLAN** est utilisé pour acheminer les données entre le site central (**Hub**) et le site distant (**Spoke**).

C'est sur ce dernier réseau (VLAN orange) que la session PPPoE doit être établie pour que le plan d'adressage réseau de l'entreprise soit conforme.

Enfin, les deux rectangles en gris pointillé identifient les machines virtuelles utilisées pour les manipulations.

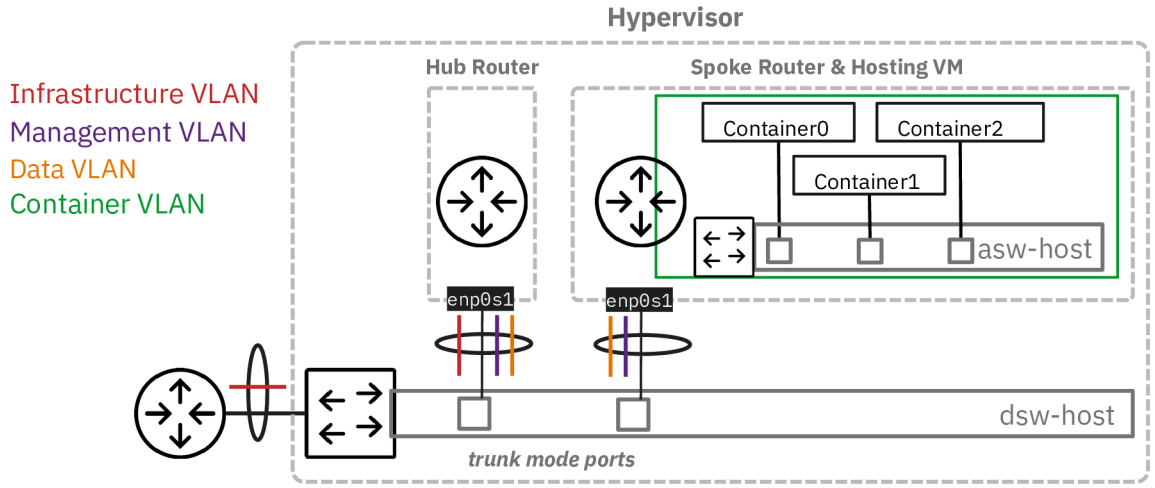



### Topologie logique

La représentation de la topologie vue sous l'angle de la virtualisation sur un système hôte hyperviseur montre que le VLAN vert appelé ***Container VLAN*** n'est visible qu'à l'intérieur de la machine virtuelle qui représente le site distant. Ce VLAN est isolé et ses préfixes réseau IPv4 et IPv6 doivent être **routés**. C'est la raison pour laquelle la machine virtuelle du site distant dispose de son propre commutateur : `asw-host`.

Tous les autres 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. Les deux routeurs virtuels **Hub** et **Spoke** sont raccordés sur des ports configurés en mode ***trunk*** sur lesquels le trafic de plusieurs VLANs doit transiter.

Côté conteneurs, le raccordement au commutateur `asw-host` sera assuré automatiquement par le gestionnaire **Incus**.




### Topologie hébergée

Voici le plan d'adressage utilisé pour la maquette qui sert à la rédaction de ce support de travaux pratiques.

**Tableau 1. Plan d'adressage de la maquette « Routage interVLAN et protocole PPPoE »**

| Planète | VLAN | Numéro | Type | Adresse |
|---|---|---|---|---|
| Maquette | Rouge | 360 | Passerelle | `192.168.104.130/29`<br>`2001:db8:168::1/64` |
| | Violet | 440 | Adresse | `fe80:1b8::1` |
| | | | | `fe80:1b8::2` |
| | Orange | 441 | Point à point | `10.4.41.1:10.4.41.2` |
| | | | Authentifiants | `spoke_site0 / 5p0k3` |
| | Vert | 40 | Passerelle | `203.0.113.1/24`<br>`fda0:7a62:28::1/64` |

## 5. Raccordement au commutateur de distribution

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

Q4 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.

Q5 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
```

Q6 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.

Q7 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 # Hub port
          type: OVSPort
          vlan_mode: trunk
          trunks: [360, 440, 441]
        - name: tap6 # Spoke port
          type: OVSPort
          vlan_mode: trunk
          trunks: [52, 440, 441] # Avec VLAN d'accès temporaire
#         trunks: [440, 441] # Sans VLAN d'accès temporaire
```

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 is already set to trunk
>> Port tap5 trunks set to [360, 440, 441]
>> Port tap6 vlan_mode set to trunk
>> Port tap6 trunks set to [52, 440, 441]
----------------------------------------
```

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

Q8 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: hub
      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: spoke
      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 lab2.yaml
```

```
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to hub.qcow2...done
Creating OVMF_CODE.fd symlink...
Creating hub_OVMF_VARS.fd file...
Starting hub...
~> Virtual machine filename   : hub.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
hub started!
Copying /home/etudianttest/masters/debian-testing-amd64.qcow2 to spoke.qcow2...done
Creating spoke_OVMF_VARS.fd file...
Starting spoke...
~> Virtual machine filename   : spoke.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, trunk mode
~> IPv6 LL address            : fe80::baad:caff:fefe:6%dsw-host
spoke started!
```

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

# 6. Routeur Hub

Dans cette section, on étudie la machine virtuelle qui joue le rôle de routeur entre le réseau d'infrastructure (VLAN rouge) et le réseau étendu (VLANs violet et orange) qui dessert le site distant.

## 6.1. 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.

Q9 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:~$ sudo hostnamectl hostname hub
etu@localhost:~$ sudo reboot
```

Q10 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'infrastructure (VLAN rouge). Cette interface doit désigner les passerelles IPv4 et IPv6 de façon à joindre l'Internet.
- Une sous-interface doit être créée pour le réseau étendu de l'exploitant des fourreaux et du câblage en fibres optiques (VLAN violet). Cette interface ne comprend qu'une adresse IPv6 de lien local pour les tests de connectivité ICMPv6 entre les deux sites.
- Une autre sous-interface doit être créée pour le réseau étendu de l'opérateur (VLAN orange). Cette interface ne contient aucune adresse lors de l'initialisation système. C'est le démon **pppd** qui est responsable de l'attribution des adresses IPv4 et IPv6 lotrs de l'établissement de la session du protocole PPP.

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:db8:3::2

  vlans:
    enp0s1.360: # VLAN rouge
      id: 360
      link: enp0s1
      addresses:
        - 192.168.104.130/29
        - 2001:db8:168::82/64
      routes:
        - to: default
          via: 192.168.104.129
        - to: "::/0"
          via: "fe80:168::1"
          on-link: true
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::1/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
```

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 **netplan** qui reprend l'ensemble de la configuration réseau.

```
sudo netplan status
```

```
     Online state: online
    DNS Addresses: 2001:db8:3::2 (compat)
                   172.16.0.2 (compat)
                   2001:db8:3::2 (compat)
       DNS Search: .

#  1: lo ethernet UNKNOWN/UP (unmanaged)
      MAC Address: 00:00:00:00:00:00
        Addresses: 127.0.0.1/8
                   ::1/128

#  2: enp0s1 ethernet UP (networkd: enp0s1)
      MAC Address: b8:ad:ca:fe:00:05 (Red Hat, Inc.)
        Addresses: fe80::baad:caff:fefe:5/64 (link)
    DNS Addresses: 172.16.0.2
                   2001:db8:3::2
           Routes: fe80::/64 metric 256

#  3: enp0s1.441 vlan UP (networkd: enp0s1.441)
      MAC Address: b8:ad:ca:fe:00:05
        Addresses: fe80::baad:caff:fefe:5/64 (link)
           Routes: fe80::/64 metric 256

#  4: enp0s1.440 vlan UP (networkd: enp0s1.440)
      MAC Address: b8:ad:ca:fe:00:05
        Addresses: fe80:1b8::1/64 (link)
                   fe80::baad:caff:fefe:5/64 (link)
           Routes: fe80::/64 metric 256
                   fe80:1b8::/64 metric 256

#  5: enp0s1.360 vlan UP (networkd: enp0s1.360)
      MAC Address: b8:ad:ca:fe:00:05
        Addresses: 192.168.104.130/29
                   2001:db8:168::baad:caff:fefe:5/64 (dynamic, ra)
                   2001:db8:168::82/64 (ra)
                   fe80::baad:caff:fefe:5/64 (link)
    DNS Addresses: 2001:db8:3::2
           Routes: default via 192.168.104.129 (static)
                   192.168.104.128/29 from 192.168.104.130 (link)
                   2001:db8:168::/64 metric 256
                   2001:db8:168::/64 metric 512 (ra)
                   fe80::/64 metric 256
                   default via fe80:168::1 metric 1024 (static)
```

Q11 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 10 mx4.quad9.net.
quad9.net mail is handled by 20 mx2.quad9.net.
quad9.net mail is handled by 5 mx1.quad9.net.
```

## 6.2. 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.

Q12 Comment activer le routage dans le sous-système réseau du noyau Linux ?

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/70-routing.conf`.

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

---

```
cat << EOF | sudo tee /etc/sysctl.d/70-routing.conf
# The 70-routing.conf parameters are applied last, after the files in the
# /usr/lib/sysctl.d directory.
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  | grep -E '(Appl|forward)'
```

```
* Applique /usr/lib/sysctl.d/10-coredump-debian.conf …
* Applique /usr/lib/sysctl.d/50-default.conf …
* Applique /usr/lib/sysctl.d/50-pid-max.conf …
*** Applique /etc/sysctl.d/70-routing.conf …**
**net.ipv4.ip_forward = 1**
**net.ipv6.conf.all.forwarding = 1**
```

Q13 **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.org
    2 0.000056361    192.0.2.2 → 9.9.9.9      DNS 81 Standard query 0xab92 AAAA 1.debian.pool.ntp.org
```

## 6.3. 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.

Q14 **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
```

Q15 **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

# Define variables
define RED_VLAN = enp0s1.360

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname $RED_VLAN counter packets 0 bytes 0 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
```

Q16 **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
```

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
```

```
Created symlink '/etc/systemd/system/sysinit.target.wants/nftables.service' → '/usr/lib/systemd/system/nftables.service'.
# nftables.service - nftables
     Loaded: loaded (/usr/lib/systemd/system/nftables.service; enabled; preset: enabled)
     Active: active (exited) since Wed 2024-09-18 19:47:08 CEST; 38ms ago
 Invocation: 91e0edd0be824dd89b4f9bed0bef6fce
       Docs: man:nft(8)
             http://wiki.nftables.org
    Process: 1066 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited, status=0/SUCCESS)
   Main PID: 1066 (code=exited, status=0/SUCCESS)
   Mem peak: 3.9M
        CPU: 31ms

sept. 18 19:47:07 hub systemd[1]: Starting nftables.service - nftables...
sept. 18 19:47:08 hub systemd[1]: Finished nftables.service - nftables.
```

Q17 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 inet nat {
        chain postrouting {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s1.360" counter packets 0 bytes 0 masquerade
        }
}
```

## 6.4. Activation du service PPPoE

Dans le scénario de ces travaux pratiques, il est nécessaire de passer par une authentification pour acheminer le trafic réseau entre les routeurs **Hub** et **Spoke**. Cette fonction est assurée à l'aide du protocole PPP.

Le protocole PPP n'a pas été conçu suivant le modèle client/serveur mais pair à pair. Il suppose que deux les processus pairs de la liaison point à point échangent des informations, dont l'authentification mutuelle.

Dans notre cas, le routeur qui tient le rôle **Hub** joue le rôle de serveur dans le sens où il exige que le routeur avec le rôle **Spoke** s'authentifie auprès de lui avant de fournir les adresses de couche réseau.

Q18 Quel paquet spécifique à la gestion du dialogue PPPoE à installer sur le routeur *Hub* ?

Rechercher dans le catalogue des paquets, la référence pppoe.

---

```
apt search ^pppoe
```

```
pppoe/testing 4.0-1 amd64
  Pilote PPP sur Ethernet
```

Le résultat de la commande **apt show pppoe** montre que c'est bien ce paquet qui répond au besoin. On peut donc l'installer.

```
sudo apt -y install pppoe
```

Q19 Quel est l'outil contenu dans le paquet demandé à la question précédente qui assure le rôle de serveur PPPoE ?

Rechercher dans la liste des outils fournis avec le paquet et les pages de manuels.

---

L'outil **pppoe-server** gère directement l'encapsulation des trames PPP dans les trames Ethernet. Il communique ensuite les paramètres utiles au démon pppd qui fonctionne de façon totalement transparente vis-à-vis de la technologie du réseau sous-jacent.

Q20 Quels sont les noms des deux sous-couches du protocole PPP qui apparaissent dans les journaux systèmes ?

Quels sont les rôles respectifs de ces deux sous-couches ?

Consulter la page **Point-to-Point Protocol**.

---

La consultation des journaux système lors du dialogue PPP fait apparaître les différents protocoles utilisés lors de l'ouverture de session.

Exemple de commande de consultation des journaux système.

```
journalctl -n 200 -f -u pppoe-server.service
```

1. LCP : *Link Control Protocol*
2. IPCP : *IP control protocol* pour la version IPv4
3. IPV6CP : *IP control protocol* pour la version IPv6

Des exemples complets de traces d'établissement de connexion PPP sont donnés à l'adresse : **Traces d'une ouverture de session PPPoE**.

Q21 **Quels sont les en-têtes du dialogue qui identifient les requêtes (émises|reçues), les rejets et les acquittements ?**

Consulter les journaux système contenant les traces d'une connexion PPP.

---

La copie d'écran donnée ci-dessus fait apparaître les directives `Conf*` pour chaque paramètre négocié.

- `ConfReq` indique une requête.
- `ConfAck` indique un acquittement.
- `ConfNak` indique un rejet.

Q22 **Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole EAP pour la méthode CHAP ?**

Consulter les pages de manuels du démon **pppd** à la section **AUTHENTICATION**.

---

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples *login/password* utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier.

```
# Secrets for authentication using CHAP
# client     server secret       IP addresses
"spoke_site0"   *       "5p0k3"      *
```

On peut le compléter directement à l'aide de la commande suivante.

```
echo '"spoke_site0"   *       "5p0k3"   *' | sudo tee -a /etc/ppp/chap-secrets
```

Q23 **Dans quel fichier sont stockés les paramètres passés au démon pppd lors du lancement du serveur PPPoE ?**

Consulter les pages de manuels de l'outil **pppoe-server**.

---

C'est le fichier `/etc/ppp/pppoe-server-options` qui contient la liste des paramètres utilisés lors du dialogue PPP.

Q24 **Quelles sont les options du protocole PPP qui doivent être implantées dans le fichier demandé à la question précédente ?**

Consulter les pages de manuels du démon **pppd** et rechercher les paramètres correspondant à la liste suivante.

- Afficher en détail toutes les étapes d'établissement de session dans les journaux système.
- Référencer l'identifiant du compte utilisateur à utiliser lors de l'authentification du routeur vert. Cette option implique que le compte utilisateur existe sur le système et qu'il soit présent dans le fichier `/etc/ppp/chap-secrets`.
- Imposer au routeur vert une authentification via EAP (*Extensible Authentication Protocol*) sans utiliser les certificats TLS pour simplifier la configuration.
- Préserver la route par défaut, et donc l'accès Internet, du routeur bleu.
- Publier l'adresse IP du serveur DNS à utiliser pour la résolution des noms de domaines.
- Activer l'utilisation des protocoles IPv6CP et IPv6.

---

Voici une copie de la commande de création du fichier `/etc/ppp/pppoe-server-options` qui contient la liste des paramètres demandés.

```
cat << 'EOF' | sudo tee /etc/ppp/pppoe-server-options
# Gestion de session avec PAM
login
# Authentification EAP
require-eap
# Le Routeur Hub détient déjà une route par défaut
nodefaultroute
# Envoi de l'adresse de résolution DNS avec les adresses IPv4
ms-dns 172.16.0.2
# Ajout du protocole IPv6
+ipv6
# Informations détaillées dans la journalisation
debug
# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q25 **Comment créer le compte utilisateur local sur le routeur bleu sachant qu'il n'est autorisé ni à se connecter ni à avoir un répertoire personnel ?**

Consulter les options de la commande **adduser**.

---

Voici un exemple de commande **adduser**.

```
sudo adduser --gecos 'Spoke Router site0' --disabled-login --no-create-home spoke_site0
```

---

**Avertissement**

Cet utilisateur doit porter le même nom que celui défini dans le fichier `/etc/ppp/chap-secrets`.

---

Q26 **Quels sont les paramètres à donner au lancement de l'outil pppoe-server pour qu'il délivre les adresses au routeur vert après authentification de celui-ci ?**

Consulter les options de la commande **pppoe-server**.

---

Voici un exemple de lancement manuel de la commande **pppoe-server**.

```
sudo pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1
```

Cette commande individuelle est à utiliser pour faire un tout premier test. Pour rendre la configuration persistante au redémarrage nous avons besoin de créer un service systemd. Il ne faut donc pas oublier d'arrêter le processus avant de passer à la question suivante.

```
sudo killall pppoe-server
```

Q27 Comment créer une nouvelle unité systemd responsable du lancement du processus **pppoe-server** ?

Consulter la page **systemd Services** et rechercher la procédure à suivre pour ajouter un service au lancement du système.

---

On commence par la création du fichier de service appelé : `/etc/systemd/system/pppoe-server.service` qui contient toutes les directives de lancement du processus **pppoe-server** avec les paramètres d'adressage du lien point à point.

Dans l'exemple ci-dessous, les paramètres suivants doivent être édités pour respecter le plan d'adressage défini.

-INom d'interface réseau

Le nom d'interface réseau contient l'identifiant du VLAN sur lequel la session PPP doit être établie.

-LAdresse locale

L'adresse locale correspond à l'adresse attribuée au routeur **Hub** par le démon **pppd**.

-RAdresse distante

L'adresse distante correspond à l'adresse attribuée au routeur **Spoke** par le démon **pppd** après authentification.

Voici un exemple de création du fichier d'unité systemd.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server.service)" = "0" ]'
ExecCondition=/bin/sh -c '[ -z "$(pgrep pppoe-server)" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Q28 Comment activer le nouveau service et contrôler son état après lancement ?

Consulter la page **systemd Services** et rechercher la procédure à suivre pour activer et lancer un service.

---

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active le nouveau service.

```
sudo systemctl enable pppoe-server.service
```

On lance ce nouveau service.

```
sudo systemctl start pppoe-server.service
```

On vérifie que l'opération s'est déroulée correctement.

```
systemctl status pppoe-server.service
```

```
# pppoe-server.service - PPPoE Server
     Loaded: loaded (/etc/systemd/system/pppoe-server.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-21 15:39:39 CEST; 36min ago
 Invocation: 23332f16c26f4889a9f063870e77da9c
    Process: 2958 ExecCondition=/bin/sh -c [ "$(systemctl show --property MainPID --value pppoe-server.service)" = "0" ] (c
    Process: 2960 ExecCondition=/bin/sh -c [ -z "$(pgrep pppoe-server)" ] (code=exited, status=0/SUCCESS)
    Process: 2964 ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1 (code=exited, statu
   Main PID: 2966 (pppoe-server)
      Tasks: 1 (limit: 1086)
     Memory: 208K (peak: 1.8M)
        CPU: 67ms
     CGroup: /system.slice/pppoe-server.service
             └─2966 /usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.4.41.1 -R 10.4.41.2 -N 1

sept. 21 15:39:39 hub systemd[1]: Starting pppoe-server.service - PPPoE Server...
sept. 21 15:39:39 hub systemd[1]: Started pppoe-server.service - PPPoE Server.
```

la copie d'écran ci-dessus montre que le service **pppoe-server** est lancé avec le jeu de paramètres de la maquette.

En l'état actuel de la configuration, aucune session PPP n'a encore été établie. Il faut maintenant passer à la configuration réseau du routeur **Spoke** pour avancer dans l'utilisation du protocole PPP.

# 7. Routeur Spoke

Dans cette section, on étudie la machine virtuelle qui joue le rôle de routeur entre le réseau étendu (VLANs violet et orange) et le réseau d'hébergement des conteneurs (VLAN vert) du site distant.

## 7.1. Configuration des interfaces du routeur

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.

Q29 Comment changer le nom de la machine virtuelle ?

---

Il faut éditer les deux fichiers `/etc/hosts` et `/etc/hostname` en remplaçant le nom de l'image maître **vm0** 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:~$ sudo hostnamectl hostname spoke
etu@localhost:~$ sudo reboot
```

Q30 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 étendu de l'exploitant des fourreaux et du câblage en fibres optiques (VLAN violet). Cette interface ne comprend qu'une adresse IPv6 de lien local pour les tests de connectivité ICMPv6 entre les deux sites.
- Une autre sous-interface doit être créée pour le réseau étendu de l'opérateur (VLAN orange). Cette interface ne contient aucune adresse lors de l'initialisation système. C'est le démon **pppd** qui est responsable de l'attribution des adresses IPv4 et IPv6 lotrs de l'établissement de la session du protocole PPP.
- Une sous-interface temporaire doit être créée dans le but d'installer les paquets d'outils nécessaires à l'établissement de la session du protocole PPP. Dès que la configuration de session PPP est en place, cette interface doit être détruite pour éviter toute confusion sur l'acheminement du trafic.
- Une sous-interface devra être créée par la suite pour le réseau d'hébergement 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:db8:3::2

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
    enp0s1.52: # VLAN accès temporaire
      id: 52
      link: enp0s1
      dhcp4: true
      dhcp6: false
      accept-ra: true
```

## 7.2. 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.

On retrouve ici les mêmes opérations que celles effectuées sur le routeur qui tient le rôle **Hub**.

Q31 Comment activer le routage dans le sous-système réseau du noyau Linux ?

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/70-routing.conf`.

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

---

```
cat << EOF | sudo tee /etc/sysctl.d/70-routing.conf
# The 70-routing.conf parameters are applied last, after the files in the
# /usr/lib/sysctl.d directory.
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
```

## 7.3. Configurer le protocole PPP

Le routeur **Spoke** doit utiliser un démon pppd sur le VLAN `Data` (orange) pour établir une session PPP avec le routeur **Hub**. À la différence de ce dernier, il n'est pas à l'initiative du dialogue PPPoE mais il doit être capable de gérer l'encapsulation des trames PPP sur un réseau local Ethernet.

Q32 Quel paquet fournit le démon de gestion des sessions du protocole PPP sur le routeur **Spoke** ?

Rechercher dans le catalogue des paquets, la référence ppp.

---

```
apt search ^ppp
```

```
ppp/testing 2.5.0-1+2 amd64
  protocole point à point (PPP) - démon

ppp-dev/testing 2.5.0-1+2 all
  protocole point à point (PPP) – fichiers de développement

ppp-gatekeeper/testing 0.1.0-201406111015-1.1 all
  PPP manager for handling balanced, redundant and failover links

pppoe/testing 4.0-1 amd64
  Pilote PPP sur Ethernet

pppoeconf/testing 1.21+nmu3 all
  configures PPPoE/ADSL connections

wmppp.app/testing 1.3.2-2 amd64
  contrôle de connexion et surveillance de la charge réseau avec aspect NeXTStep
```

Le résultat de la commande **apt show ppp** montre que c'est bien ce paquet qui répond au besoin.

```
sudo apt -y install ppp
```

Q33 Comment utiliser l'encapsulation des trames PPP dans Ethernet à partir du démon **pppd** fourni avec le paquet ppp ?

Rechercher dans le répertoire de documentation du paquet ppp.

---

Dans le répertoire `/usr/share/doc/ppp/`, on trouve le fichier `README.pppoe` qui indique que l'appel au module `rp-pppoe.so` permet d'encapsuler des trames PPP sur un réseau local Ethernet.

Toujours à partir du même répertoire, on trouve dans la liste des fichiers d'exemples de configuration un modèle adapté à notre contexte : `peers-pppoe`.

Q34 Dans quel fichier sont stockés les paramètres d'identité et d'authentification utilisés par le protocole CHAP ?

Consulter les pages de manuels du démon **pppd** à la section **AUTHENTICATION**.

---

C'est le fichier `/etc/ppp/chap-secrets` qui contient les couples *login/password* utilisés lors de l'authentification.

Voici un exemple du contenu de ce fichier. Le nom du client ainsi que son mot de passe secret doivent être identiques à chaque extrémité de la session PPP.

```
# Secrets for authentication using CHAP
# client server secret   IP addresses
"spoke_site0"   *  "5p0k3"   *
```

Q35 Quelles sont les options de configuration du démon **pppd** à placer dans le fichier `/etc/ppp/peers/pppoe-provider` pour assurer l'établissement de la session PPP entre les routeurs ?

Utiliser le fichier exemple PPPoE fourni avec la documentation du paquet ppp.

---

Voici comment créer un fichier `/etc/ppp/peers/pppoe-provider` avec les options correspondant au contexte de la maquette du routeur vert.

```
cat << 'EOF' | sudo tee /etc/ppp/peers/pppoe-provider
# Le nom d'utilisateur désigne l'entrée du fichier /etc/ppp/chap-secrets
user spoke_site0

# Chargement du module PPPoE avec les détails dans la journalisation
plugin rp-pppoe.so rp_pppoe_ac BRAS rp_pppoe_verbose 1

# Interface (VLAN) utilisé pour l'établissement de la session PPP
enp0s1.441

# Les adresses sont attribuées par le "serveur" PPPoE
noipdefault
# L'adresse de résolution DNS est aussi fournie par le serveur PPPoE
usepeerdns
# La session PPP devient la route par défaut du routeur Spoke
defaultroute

# Demande de réouverture de session automatique en cas de rupture
persist

# Le routeur Spoke n'exige pas que le routeur Hub s'authentifie
noauth

# Messages d'informations détaillés dans la journalisation
debug

# Utilisation du protocole IPv6
+ipv6

# Options préconisées par la documentation
noaccomp
default-asyncmap
nodeflate
nopcomp
novj
novjccomp
lcp-echo-interval 10
EOF
```

Q36 Comment lancer le démon **pppd** pour qu'il prenne en compte les paramètres définis dans le fichier complété à la question précédente ?

Consulter les pages de manuels du démon **pppd**.

---

C'est l'outil **pon** qui permet de désigner le fichier de configuration à utiliser. Voici une copie d'écran du lancement du démon **pppd**.

```
sudo pon pppoe-provider
```

Cette commande individuelle est à utiliser pour faire un tout premier test. Pour rendre la configuration persistante au redémarrage nous avons besoin de créer un service systemd. Il ne faut donc pas oublier d'arrêter le processus avant de passer à la question suivante. Le paquet fournit un outil dédié : **poff**.

```
sudo poff -a pppoe-provider
```

Q37 Quels sont les noms des deux sous-couches du protocole PPP qui apparaissent dans les journaux systèmes ? Quels sont les rôles respectifs de ces deux sous-couches ?

Consulter la page **Point-to-Point Protocol**.

---

La consultation des journaux système lors du dialogue PPP fait apparaître tous les détails. Voir les exemples de traces à l'adresse : **Traces d'une ouverture de session PPPoE**.

Q38 Quels sont les en-têtes du dialogue qui identifient les requêtes (émises|reçues), les rejets et les acquittements ?

Consulter les journaux système contenant les traces d'une connexion PPP.

---

La copie d'écran donnée ci-dessus fait apparaître les directives `Conf*` pour chaque paramètre négocié.

- `ConfReq` indique une requête.
- `ConfAck` indique un acquittement.
- `ConfNak` indique un rejet.

Q39 Comment assurer une ouverture automatique de la session PPP à chaque réinitialisation système ?

Consulter la page **systemd Services** et rechercher la procédure à suivre pour ajouter un service au lancement du système.

---

On commence par la création du fichier de service appelé : /etc/systemd/system/ppp.service qui contient les appels aux outils **pon** et **poff**.

Voici l'instruction de création du fichier de service.

```
cat << EOF | sudo tee /etc/systemd/system/ppp.service
[Unit]
Description=PPPoE Client Connection
After=network.target
Wants=network.target
BindsTo=sys-subsystem-net-devices-enp0s1.441.device
After=sys-subsystem-net-devices-enp0s1.441.device

[Service]
Type=forking
ExecStart=/usr/bin/pon pppoe-provider
ExecStop=/usr/bin/poff pppoe-provider
Restart=on-failure
RestartSec=20

[Install]
WantedBy=multi-user.target
EOF
```

Q40 Comment activer le nouveau service et contrôler son état après lancement ?

Consulter la page **systemd Services** et rechercher la procédure à suivre pour activer et lancer un service.

---

On commence par la relecture de la liste des services disponibles par le gestionnaire systemd.

```
sudo systemctl daemon-reload
```

On active le nouveau service.

```
sudo systemctl enable ppp.service
```

On lance ce nouveau service.

```
sudo systemctl start ppp.service
```

On vérifie que l'opération s'est déroulée correctement.

```
systemctl status ppp.service
```

```
# ppp.service - PPPoE Client Connection
     Loaded: loaded (/etc/systemd/system/ppp.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-22 08:21:32 CEST; 13min ago
 Invocation: 69386a40f7574821b3986ed6c6c242f7
   Main PID: 496 (pppd)
      Tasks: 1 (limit: 1086)
     Memory: 2.8M (peak: 4.9M)
        CPU: 56ms
     CGroup: /system.slice/ppp.service
             └─496 /usr/sbin/pppd call pppoe-provider

sept. 22 08:21:37 spoke pppd[496]: rcvd [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-pre-up started (pid 510)
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-pre-up finished (pid 510), status = 0x0
sept. 22 08:21:37 spoke pppd[496]: local  IP address 10.4.41.2
sept. 22 08:21:37 spoke pppd[496]: remote IP address 10.4.41.1
sept. 22 08:21:37 spoke pppd[496]: primary   DNS address 172.16.0.2
sept. 22 08:21:37 spoke pppd[496]: secondary DNS address 172.16.0.2
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-up started (pid 514)
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ipv6-up finished (pid 509), status = 0x0
sept. 22 08:21:37 spoke pppd[496]: Script /etc/ppp/ip-up finished (pid 514), status = 0x0
```

Q41 Comment utiliser la session PPP (le VLAN orange) comme lien unique de raccordement réseau du routeur **Spoke** ?

Maintenant que le fonctionnement de la session PPP est validé, nous n'avons plus besoin du raccordement temporaire sur le routeur **Spoke**. Il faut donc commenter les entrées du fichier /etc/netplan/enp0s1.yaml qui ne sont plus utiles et attribuer l'adresse de résolution DNS de secours.

Une fois ces opérations effectuées, on peut redémarre le routeur **Spoke** pour se placer en situation de raccordement distant.

---

Pour commencer, on commente les entrées inutile du fichier /etc/netplan/enp0s1.yaml.

```
cat /etc/netplan/enp0s1.yaml
```

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false
#     nameservers:
#       addresses:
#          - 172.16.0.2
#          - 2001:db8:3::2

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
#   enp0s1.52: # VLAN accès temporaire
#     id: 52
#     link: enp0s1
#     dhcp4: true
#     dhcp6: false
#     accept-ra: true
```

On peut appliquer directement les modifications à l'aide de la commande **netplan**.

```
sudo netplan apply
```

```
sudo netplan status
```

**Attention**

L'affectation de l'adresse IPv4 ou IPv6 de résolution DNS pose problème. En effet, si le démon **pppd** propose bien deux adresses via l'option `usepeerdns`, ces propositions ne sont pas prises en charge par le service systemd-resolved.

On contourne cette difficulté en affectant une adresse IPv4 directement au service systemd-resolved.

On édite le fichier `/etc/systemd/resolved.conf` pour affecter directement l'adresse de résolution DNS. Voici une copie des lignes utiles du fichier modifié. Toutes les autres lignes sont commentées.

```
grep -Ev '(^#|^$)'  /etc/systemd/resolved.conf
[Resolve]
DNS=172.16.0.2
```

Il ne faut pas oublier de relancer le service pour prendre en compte les modifications du fichier.

```
sudo systemctl restart systemd-resolved
```

Le routeur **Spoke** est maintenant prêt à être redémarré pour utiliser le lien de raccordement distant comme seul canal d'accès aux autres réseaux.

```
sudo reboot
```

## 8. Réseau d'hébergement de conteneurs

À ce stade des manipulations, le routeur **Spoke** utilise la session PPP et le routage IPv4 pour accéder à tous les réseaux.

On peut le vérifier en affichant les tables de routage IPv4 et IPv6.

Dans la table de routage IPv4, on trouve la route par défaut.

```
ip route ls
```

```
default dev ppp0 scope link
10.4.41.1 dev ppp0 proto kernel scope link src 10.4.41.2
```

Dans la table de routage IPv6, on ne trouve que des entrées de lien local correspondant au préfixe `fe80::/10`.

```
ip -6 route ls
```

```
fe80::d1b3:8c16:93d1:8370 dev ppp0 proto kernel metric 256 pref medium
fe80::f118:5b7b:cfb5:7b54 dev ppp0 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.441 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.440 proto kernel metric 256 pref medium
fe80:1b8::/64 dev enp0s1.440 proto kernel metric 256 pref medium
```

Dans cette partie, nous devons ajouter un commutateur pour raccorder les services hébergés sur le site distant et compléter la configuration du routage pour assurer les accès IPv4 et IPv6.

### 8.1. Ajouter un commutateur virtuel

Dans le scénario étudié, les services sont hébergés dans un réseau de conteneurs propre au routeur **Spoke**. La mise en œuvre de cette configuration passe par l'installation d'un commutateur virtuel appelé `asw-host`. On utilise Open vSwitch pour configurer ce commutateur.

Q42 **Quel est le paquet à installer pour ajouter un commutateur virtuel au routeur Spoke ?**

Rechercher le mot clé openvswitch dans la liste des paquets.

---

Voici un exemple de recherche.

```
apt search ^openvswitch
```

C'est le paquet `openvswitch-switch` qui nous intéresse. On l'installe.

```
sudo apt -y install openvswitch-switch
```

Q43 **Comment déclarer un commutateur à l'aide de l'outil netplan.io ?**

Consulter la documentation de **Netplan** pour obtenir les informations sur la configuration des commutateurs virtuels `openvswitch` à l'adresse **Netplan documentation**.

On peut aussi rechercher les informations dans les fichiers exemples fournis avec le paquet netplan.io.

---

Voici un exemple de recherche.

```
find /usr/share/doc/netplan* -type f -iname "openvswitch*"
/usr/share/doc/netplan/examples/openvswitch.yaml
```

Q44 **Quelles sont les modifications à apporter au fichier de déclaration YAML `/etc/netplan/enp0s1.yaml` pour créer le commutateur `asw-host` ?**

---

Voici une copie du fichier `/etc/netplan/enp0s1.yaml` qui contient les instructions de création du commutateur `asw-host` seul.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

    openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
```

On applique les nouvelles déclarations.

```
sudo netplan apply
```

On vérifie que le nouveau commutateur a bien été créé dans la base Open vSwitch.

```
sudo ovs-vsctl show
```

```
e288cc30-e290-44ae-8ed1-5e2a8d184033
    Bridge asw-host
        fail_mode: standalone
        Port asw-host
            Interface asw-host
                type: internal
    ovs_version: "3.4.0"
```

Q45 Comment ajouter une nouvelle interface virtuelle commutée (*Switched Virtual Interface*) qui servira de passerelle par défaut pour tous les hôtes du réseau d'hébergement du site distant ?

Rechercher dans la documentation Netplan des exemples de déclarations d'interfaces de type SVI appartenant à des VLANs.

---

Voici une nouvelle copie du fichier /etc/netplan/enp0s1.yaml auquel on ajouté la déclaration d'une interface vlan40 avec les adresses IPv4 et IPv6 conformes au contexte de la maquette utilisée pour la rédaction de ce document.

```
network:
  version: 2
  ethernets:
    enp0s1:
      dhcp4: false
      dhcp6: false
      accept-ra: false

  openvswitch: {}

  bridges:
    asw-host:
      openvswitch: {}

  vlans:
    enp0s1.440: # VLAN violet
      id: 440
      link: enp0s1
      addresses:
        - fe80:1b8::2/64
    enp0s1.441: # VLAN orange
      id: 441
      link: enp0s1
      addresses: []
    vlan40:    # VLAN vert
      id: 40
      link: asw-host
      addresses:
        - 203.0.113.1/24
        - fda0:7a62:28::1/64
        - fe80:28::1/64
```

## 8.2. Routage du réseau d'hébergement

L'objectif de cette section est de rendre le réseau d'hébergement accessible depuis le routeur **Hub** et que le protocole IPv6 soit utilisable depuis le routeur **Spoke**.

Pour rendre le réseau d'hébergement du site distant accessible depuis le routeur **Hub**, il est nécessaire d'ajouter des routes statiques IPv4 et IPv6 à l'ouverture de la session PPP.

Pour utiliser IPv6 depuis le routeur **Spoke**, il faut ajouter une route par défaut IPv6 aussi à l'ouverture de session PPP.

On commence par l'ajout de routes statiques IPv4 et IPv6 côté routeur **Hub**.

Q46 Comment ajouter manuellement les routes IPv4 et IPv6 vers le réseau desservi par le routeur vert ?

Consulter les pages de manuel sur le routage avec la commande : `man ip-route`.

---

Sachant que le site distant est raccordé via une liaison point à point unique, on choisit de désigner la destination par l'interface de la liaison.

```
sudo ip route add 203.0.113.0/24 dev ppp0
sudo ip -6 route add fda0:7a62:28::/64 dev ppp0
```

Q47 Comment appliquer ces routes statiques dans la configuration système pour qu'elles soient activées à chaque établissement de session PPP ?

Il faut parcourir l'arborescence du répertoire `/etc/ppp/` pour repérer les scripts exécutés lors de l'ouverture de session. Créer un script pour chaque protocole de couche réseau qui ajoute la route statique voulue.

---

- Pour IPv4, le répertoire est `/etc/ppp/ip-up.d/`. Voici comment créer le script exécutable `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/staticroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
    ip route add 203.0.113.0/24 dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/staticroute
```

- Pour IPv6, le répertoire est `/etc/ppp/ipv6-up.d/`. Voici comment créer le script exécutable `staticroute`.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/staticroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
    ip -6 route add fda0:7a62:28::/64 dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/staticroute
```

Q48 Comment tester l'ajout de ces routes statiques et les communications vers le réseau d'hébergement depuis le routeur **Hub** ?

Afficher les tables de routage après réinitialisation d'une session PPP et lancer des tests ICMP **vers** les adresses de l'interface virtuelle commutée du routeur Spoke.

---

En redémarrant le service `pppoe-server` sur le routeur **Hub** ou le service `ppp` sur le routeur **Spoke**, on provoque un renouvellement de session PPP.

On peut ensuite afficher les tables de routage du routeur **Hub**.

```
ip route ls
default via 192.168.104.129 dev enp0s1.360 proto static
10.4.41.2 dev ppp0 proto kernel scope link src 10.4.41.1
192.168.104.128/29 dev enp0s1.360 proto kernel scope link src 192.168.104.130
203.0.113.0/24 dev ppp0 scope link
```

```
ip -6 route ls
2001:db8:168::/64 dev enp0s1.360 proto kernel metric 256 pref medium
2001:db8:168::/64 dev enp0s1.360 proto ra metric 512 expires 2591839sec pref high
fda0:7a62:28::/64 dev ppp0 metric 1024 pref medium
fe80::d580:e038:8d05:636e dev ppp0 proto kernel metric 256 pref medium
fe80::dcfe:544d:d6ac:8b0f dev ppp0 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.441 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.440 proto kernel metric 256 pref medium
fe80::/64 dev enp0s1.360 proto kernel metric 256 pref medium
fe80:1b8::/64 dev enp0s1.440 proto kernel metric 256 pref medium
default via fe80:168::1 dev enp0s1.360 proto static metric 1024 onlink pref medium
```

On peut aussi afficher la solution de routage pour une adresse destination.

```
ip route get 203.0.113.1
203.0.113.1 dev ppp0 src 10.4.41.1 uid 1000
    cache
```

```
ip -6 route get fda0:7a62:28::1
fda0:7a62:28::1 from :: dev ppp0 src 2001:db8:168::baad:caff:fefe:5 metric 1024 pref medium
```

On peut maintenant passer au routeur **Spoke** pour effectuer le même travail sur les routes par défaut.

Q49 Comment ajouter des routes par défaut dans la configuration système pour qu'elles soient activées à chaque établissement de session PPP ?

Il faut parcourir l'arborescence du répertoire /etc/ppp/ pour repérer les scripts exécutés lors de l'ouverture de session. Créer un script pour chaque protocole de couche réseau qui ajoute la route statique voulue.

---

- Pour IPv4, le répertoire est /etc/ppp/ip-up.d/. Voici comment créer un script exécutable appelé defaultroute.

```
cat << 'EOF' | sudo tee /etc/ppp/ip-up.d/defaultroute
#!/bin/sh

if [ -z "${CONNECT_TIME}" ]; then
 ip route add default dev ${PPP_IFACE}
fi
EOF
```

```
sudo chmod +x /etc/ppp/ip-up.d/defaultroute
```

- Pour IPv6, le répertoire est /etc/ppp/ipv6-up.d/. Voici comment créer un script exécutable aussi appelédefaultroute.

```
cat << 'EOF' | sudo tee /etc/ppp/ipv6-up.d/defaultroute
#!/bin/sh

  if [ -z "${CONNECT_TIME}" ]; then
      ip -6 route add default dev ${PPP_IFACE}
  fi
EOF
```

```
sudo chmod +x /etc/ppp/ipv6-up.d/defaultroute
```

Q50 Comment tester l'ajout des routes par défaut et les communications IPv6 depuis le routeur **Spoke** ?

Afficher les tables de routage après réinitialisation d'une session PPP et lancer des tests ICMP **depuis** les adresses de l'interface virtuelle commutée du routeur Spoke.

---

En redémarrant le service pppoe-server sur le routeur **Hub** ou le service ppp sur le routeur **Spoke**, on provoque un renouvellement de session PPP.

On peut ensuite afficher les tables de routage du routeur **Spoke**.

```
ip route ls default
default dev ppp0 scope link
```

```
ip -6 route ls default
default dev ppp0 metric 1024 pref medium
```

Il ne reste plus que le test ICMPv6 pour qualifier le routage complet au niveau du routeur **Spoke**.

```
ping -c2 2620:fe::fe
PING 2620:fe::fe (2620:fe::fe) 56 data bytes
64 bytes from 2620:fe::fe: icmp_seq=1 ttl=58 time=54.3 ms
64 bytes from 2620:fe::fe: icmp_seq=2 ttl=58 time=41.2 ms

--- 2620:fe::fe ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 41.208/47.735/54.262/6.527 ms
```

## 8.3. Adressage automatique dans le réseau d'hébergement

Pour que les hôtes du réseau de conteneurs obtiennent automatiquement une configuration IPv4 et IPv6, il faut ajouter le service dnsmasq sur le routeur **Spoke**. Il fournit les services DHCPv4 et SLAAC en un seul et unique fichier de configuration.

On débute par l'installation du paquet.

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

Q51 **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=vlan40

# Enable DHCPv4 on Container VLAN
dhcp-range=203.0.113.20,203.0.113.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:vlan40,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:db8:3::2],[2620:fe::fe]

# Avoid DNS listen port conflict between dnsmasq and systemd-resolved
port=0
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
```

# 9. Conteneurs système Incus

Maintenant que la configuration réseau de la topologie étudiée est complète, on peut passer à la gestion des conteneurs de service du site distant.

## 9.1. Installation du gestionnaire de conteneurs Incus

Sur le routeur **Spoke**, la gestion des conteneurs est confiée à Incus. Dans le contexte de ces manipulations, nous utilisons le mode **bridge** pour le raccordement réseau des conteneurs. Que l'on lance 3 conteneurs ou 300, ceux-ci seront raccordés de façon transparente au commutateur virtuel `asw-host` et bénéficieront d'un adressage automatique.

Q52 **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
```

Q53 **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
```

## 9.2. Configuration et lancement des conteneurs

Q54 **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]:
Where should this storage pool store its data? [default=/var/lib/incus/storage-pools/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: asw-host
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]:
```

Q55 **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: asw-host
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: default
used_by: []
project: default
```

L'affichage de ce profil par défaut permet d'identifier les clés à modifier ou à ajouter.

Mode de raccordementnictype

La valeur attribuée par défaut est `macvlan`. Elle ne correspond pas à notre besoin puisque les services hébergés sur le site distant doivent pouvoir communiquer entre eux. On remplace cette valeur par `bridged`.

```
incus profile device set default eth0 nictype bridged
```

```
incus profile device get default eth0 nictype
bridged
```

Attribution de VLANvlan

La configuration du profil par défaut n'attribue pas l'appartenance à un VLAN particulier. Dans le contexte de cette maquette, le réseau des conteneurs correspond au VLAN `40`. On ajoute la clé `vlan` au profil par défaut.

```
incus profile device set default eth0 vlan 40
```

```
incus profile device get default eth0 vlan
40
```

Q56 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    | SNAPSHOTS |
+------+---------+----------------------+-----------------------------------------+-----------+-----------+
| c0   | RUNNING | 203.0.113.144 (eth0) | fda0:7a62:28:0:216:3eff:fe67:b848 (eth0) | CONTAINER | 0         |
+------+---------+----------------------+-----------------------------------------+-----------+-----------+
| c1   | RUNNING | 203.0.113.76 (eth0)  | fda0:7a62:28:0:216:3eff:fe0a:c3ba (eth0) | CONTAINER | 0         |
+------+---------+----------------------+-----------------------------------------+-----------+-----------+
| c2   | RUNNING | 203.0.113.184 (eth0) | fda0:7a62:28:0:216:3eff:febb:bfb9 (eth0) | CONTAINER | 0         |
+------+---------+----------------------+-----------------------------------------+-----------+-----------+
```

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

Q57 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
```

```
>>>>>>>>>>>>>>>> c0
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 = 29.420/29.666/29.913/0.246 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 = 41.829/47.305/52.782/5.476 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 1002ms
rtt min/avg/max/mdev = 29.477/29.490/29.504/0.013 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 = 41.401/41.546/41.692/0.145 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 1001ms
rtt min/avg/max/mdev = 28.820/29.427/30.034/0.607 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 = 40.708/41.037/41.367/0.329 ms
```

Q58 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 constitué des arguments de la ligne de commande. Ces commandes sont exécutées sur chacun des conteneurs actifs.

```
#!/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
```

Voici un exemple d'exécution du script `run-commands-in-containers.sh` qui contient les instructions ci-dessus.

```
bash run-commands-in-containers.sh "apt update" "apt -y full-upgrade" "apt clean" "apt -y autopurge"
```

```
>>>>>>>>>>>>>>>> c0
Get:1 http://deb.debian.org/debian trixie InRelease [169 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [49.6 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.5 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages.diff/Index [27.9 kB]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Fetched 296 kB in 1s (216 kB/s)
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
Get:1 http://deb.debian.org/debian trixie InRelease [169 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [49.6 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.5 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages.diff/Index [27.9 kB]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Fetched 296 kB in 1s (279 kB/s)
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
Get:1 http://deb.debian.org/debian trixie InRelease [169 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [49.6 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.5 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages.diff/Index [27.9 kB]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Get:5 http://deb.debian.org/debian trixie/main amd64 Packages 2024-09-22-0804.28.pdiff [5327 B]
Fetched 296 kB in 1s (277 kB/s)
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
```

## 9.3. Adressage statique des conteneurs

Avec l'adressage automatique des conteneurs système, on sait les administrer depuis le routeur **Spoke**. Dans un scénario plus réaliste, il faut être capable d'administrer ces mêmes conteneurs depuis le routeur **Hub** ou mieux encore depuis l'infrastructure de déploiement CI/CD de l'entreprise. Ce type d'usage suppose que le conteneurs du site distant utilisent des adresses IPv4 et IPv6 statiques.

Q59 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
$(cat << EOF > eth0.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 203.0.113.$((addr_idx + 10))/24
        - fda0:7a62:28::$(printf "%x" $((addr_idx + 10)))/64
      routes:
        - to: default
          via: 203.0.113.1
        - to: "::/0"
          via: fe80:28::1
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:db8:3::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 ls
```

```
+------+---------+---------------------+-----------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4         |                  IPV6                   |   TYPE    | SNAPSHOTS |
+------+---------+---------------------+-----------------------------------------+-----------+-----------+
| c0   | RUNNING | 203.0.113.10 (eth0) | fda0:7a62:28::a (eth0)                  | CONTAINER | 0         |
|      |         |                     | fda0:7a62:28:0:216:3eff:fe67:b848 (eth0) |           |           |
+------+---------+---------------------+-----------------------------------------+-----------+-----------+
| c1   | RUNNING | 203.0.113.11 (eth0) | fda0:7a62:28::b (eth0)                  | CONTAINER | 0         |
|      |         |                     | fda0:7a62:28:0:216:3eff:fe0a:c3ba (eth0) |           |           |
+------+---------+---------------------+-----------------------------------------+-----------+-----------+
| c2   | RUNNING | 203.0.113.12 (eth0) | fda0:7a62:28::c (eth0)                  | CONTAINER | 0         |
|      |         |                     | fda0:7a62:28:0:216:3eff:febb:bfb9 (eth0) |           |           |
+------+---------+---------------------+-----------------------------------------+-----------+-----------+
```

Q60 Comment vérifier la connectivité réseau depuis les conteneurs ?

La question précédente montre que la configuration réseau des conteneurs est complète. On doit donc lancer des tests IPv4 et IPv6.

---

Voici deux exemples de tests ICMP.

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- ping -c2 2620:fe::fe
done
```

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- ping -c2 9.9.9.9
done
```

Q61 Comment ouvrir un accès SSH avec un compte utilisateur normal dans les conteneurs ?

Reprendre le script d'exécution des commandes à l'intérieur des conteneurs pour installer les paquets nécessaires, créer le compte utilisateur et autoriser la connexion SSH par mot de passe.

---

Voici un exemple de script qui assure les traitements demandés.

```
#!/bin/bash

# Function to check if a variable is a non-empty string
is_non_empty_string() {
    if [[ -n "$1" && "$1" == *[!\ ]* ]]; then
        return 0 # True
    else
        return 1 # False
    fi
}

# Check if both arguments are provided
if [ $# -ne 2 ]; then
    echo "Error: This script requires exactly two arguments."
    echo "Usage: $0 <username> <password>"
    exit 1
fi

# Check if $1 is a non-empty string
if ! is_non_empty_string "$1"; then
    echo "Error: First argument is not a valid non-empty string."
    exit 1
fi

# Check if $2 is a non-empty string
if ! is_non_empty_string "$2"; then
    echo "Error: Second argument is not a valid non-empty string."
    exit 1
fi

USER=$1
shift
PASSWD=$1

. run-commands-in-containers.sh \
        "apt -y install ssh sudo" \
        "adduser --gecos \"\" --disabled-password ${USER}" \
        "adduser ${USER} sudo" \
        "adduser ${USER} adm" \
        "sh -c \"echo '${USER}:${PASSWD}' | chpasswd\"" \
        "sed -i 's/#Port 22/Port 22\nPort 2222/' /etc/ssh/sshd_config" \
        "sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd_config" \
        "dpkg-reconfigure openssh-server"

exit 0
```

Voici un exemple de lancement du script avec un nom d'utilisateur et son mot de passe comme arguments de la ligne de commande.

```
bash set-user-and-ssh.sh "etu" "xxxxxxx"
```

Le résultat est évalué par une connexion SSH depuis le routeur **Hub**.

```
etu@hub:~$ ssh etu@fda0:7a62:28::b
The authenticity of host 'fda0:7a62:28::b (fda0:7a62:28::b)' can't be established.
ED25519 key fingerprint is SHA256:6qkAjHutYP6hbpa4iNh94y1Bk4wRqRD62ahOC+3WC9g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'fda0:7a62:28::b' (ED25519) to the list of known hosts.
etu@fda0:7a62:28::b's password:
Linux c1 6.10.9-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.10.9-1 (2024-09-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep 22 14:24:07 2024 from 203.0.113.1
etu@c1:~$
```

## 10. Traces d'une ouverture de session PPPoE

Pour obtenir la trace des transactions entre les deux routeurs de la maquette, on fait appel à la commande **journalctl** et on consulte les journaux de l'unité systemd ajoutée sur chaque routeur.

### 10.1. Journaux du routeur Spoke

Côté routeur **Spoke**, l'unité systemd s'appelle `ppp.service`. Voici la commande de consultation de l'ouverture de session PPP la plus récente.

```
journalctl -n 100 -f -u ppp.service
```

```
spoke systemd[1]: Starting ppp.service - PPPoE Client Connection...
spoke pppd[629]: Plugin rp-pppoe.so loaded.
spoke pon[629]: Plugin rp-pppoe.so loaded.
spoke pppd[639]: pppd 2.5.0 started by root, uid 0
spoke systemd[1]: Started ppp.service - PPPoE Client Connection.
spoke pppd[639]: Send PPPOE Discovery V1T1 PADI session 0x0 length 12❶
spoke pppd[639]:  dst ff:ff:ff:ff:ff:ff  src b8:ad:ca:fe:00:06
spoke pppd[639]:  [service-name] [host-uniq 7f 02 00 00]
spoke pppd[639]: Recv PPPOE Discovery V1T1 PADO session 0x0 length 44
spoke pppd[639]:  dst b8:ad:ca:fe:00:06  src b8:ad:ca:fe:00:05
spoke pppd[639]:  [AC-name BRAS] [service-name] [AC-cookie eb 9f 92 61 e1 ee 01 a1 5d 8f 11 61 8a fb c8 4b fc 01 00 00] [host-unic
spoke pppd[639]: Access-Concentrator: BRAS
spoke pppd[639]: Cookie: eb 9f 92 61 e1 ee 01 a1 5d 8f 11 61 8a fb c8 4b fc 01 00 00
spoke pppd[639]: AC-Ethernet-Address: b8:ad:ca:fe:00:05
spoke pppd[639]: ----------------------------------------------------
spoke pppd[639]: Send PPPOE Discovery V1T1 PADR session 0x0 length 36
spoke pppd[639]:  dst b8:ad:ca:fe:00:05  src b8:ad:ca:fe:00:06
spoke pppd[639]:  [service-name] [host-uniq 7f 02 00 00] [AC-cookie eb 9f 92 61 e1 ee 01 a1 5d 8f 11 61 8a fb c8 4b fc 01 00 00]
spoke pppd[639]: Recv PPPOE Discovery V1T1 PADS session 0x1 length 12
spoke pppd[639]:  dst b8:ad:ca:fe:00:06  src b8:ad:ca:fe:00:05
spoke pppd[639]:  [service-name] [host-uniq 7f 02 00 00]
spoke pppd[639]: PPP session is 1
spoke pppd[639]: Connected to B8:AD:CA:FE:00:05 via interface enp0s1.441
spoke pppd[639]: using channel 1
spoke pppd[639]: Using interface ppp0
spoke pppd[639]: Connect: ppp0 <--> enp0s1.441❷
spoke pppd[639]: sent [LCP ConfReq id=0x1 <mru 1492> <magic 0xbf826095>]
spoke pppd[639]: rcvd [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
spoke pppd[639]: sent [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
spoke pppd[639]: rcvd [LCP ConfAck id=0x1 <mru 1492> <magic 0xbf826095>]
spoke pppd[639]: sent [LCP EchoReq id=0x0 magic=0xbf826095]
spoke pppd[639]: rcvd [LCP EchoReq id=0x0 magic=0x471e9bda]
spoke pppd[639]: sent [LCP EchoRep id=0x0 magic=0xbf826095]
spoke pppd[639]: rcvd [EAP Request id=0xa1 Identity <Message "Name">]
spoke pppd[639]: EAP: Identity prompt "Name"
spoke pppd[639]: sent [EAP Response id=0xa1 Identity <Name "spoke_site0">]
spoke pppd[639]: rcvd [LCP EchoRep id=0x0 magic=0x471e9bda]
spoke pppd[639]: rcvd [EAP Request id=0xa2 MD5-Challenge <Value 81 3b a7 d1 eb 86 42 15 2c b9 1b 07 83 98 e2 dd b7 c6 57 b4 b5 0f>
spoke pppd[639]: sent [EAP Response id=0xa2 MD5-Challenge <Value 16 e7 3d fa d2 4b 6a 73 41 5f 86 c8 84 97 ed f0> <Name "spoke_sit
spoke pppd[639]: rcvd [EAP Success id=0xa3]
spoke pppd[639]: EAP authentication succeeded
spoke pppd[639]: peer from calling number B8:AD:CA:FE:00:05 authorized
spoke pppd[639]: sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]❸
spoke pppd[639]: sent [IPV6CP ConfReq id=0x1 <addr fe80::9869:8831:104a:570a>]
spoke pppd[639]: rcvd [CCP ConfReq id=0x1 <bsd v1 15>]
spoke pppd[639]: sent [CCP ConfReq id=0x1]
spoke pppd[639]: sent [CCP ConfRej id=0x1 <bsd v1 15>]
spoke pppd[639]: rcvd [IPCP ConfReq id=0x1 <addr 10.4.41.1>]
spoke pppd[639]: sent [IPCP ConfAck id=0x1 <addr 10.4.41.1>]
spoke pppd[639]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
spoke pppd[639]: sent [IPV6CP ConfAck id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
spoke pppd[639]: rcvd [IPCP ConfNak id=0x1 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke pppd[639]: sent [IPCP ConfReq id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke pppd[639]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::9869:8831:104a:570a>]
spoke pppd[639]: local  LL address fe80::9869:8831:104a:570a
spoke pppd[639]: remote LL address fe80::61d3:68e0:1e34:e123
spoke pppd[639]: Script /etc/ppp/ipv6-up started (pid 653)
spoke pppd[639]: rcvd [CCP ConfAck id=0x1]
spoke pppd[639]: rcvd [CCP ConfReq id=0x2]
spoke pppd[639]: sent [CCP ConfAck id=0x2]
spoke pppd[639]: rcvd [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke pppd[639]: Script /etc/ppp/ip-pre-up started (pid 658)
spoke pppd[639]: Script /etc/ppp/ip-pre-up finished (pid 658), status = 0x0
spoke pppd[639]: local  IP address 10.4.41.2
spoke pppd[639]: remote IP address 10.4.41.1
spoke pppd[639]: primary   DNS address 172.16.0.2
spoke pppd[639]: secondary DNS address 172.16.0.2
spoke pppd[639]: Script /etc/ppp/ip-up started (pid 662)
spoke pppd[639]: Script /etc/ppp/ipv6-up finished (pid 653), status = 0x0
spoke pppd[639]: Script /etc/ppp/ip-up finished (pid 662), status = 0x0
```

❶ Sur un réseau de diffusion il est nécessaire d'identifier les deux hôtes qui doivent établir une session PPP. Cette toute première phase d'identification utilise des trames spécifiques avec les messages décrits dans la Section 3, « Interface Ethernet & protocole PPP ».

❷ La sous-couche *Link Control Protocol* (LCP) assure la configuration automatique des interfaces à chaque extrémité. Les paramètres négociés entre les deux hôtes en communication sont multiples : l'adaptation de la taille de datagramme, les caractères d'échappement, les numéros magiques et la sélection des options d'authentification.
❸ La sous-couche *Network Control Protocol* (NCP) assure l'encapsulation de multiples protocoles de la couche réseau. Dans l'exemple donné, c'est le protocole IPv4 qui est utilisé ; d'où l'acronyme IPCP.

## 10.2. Journaux du routeur Hub

Côté routeur **hub**, l'unité systemd s'appelle `pppoe-server.service`. Voici la commande de consultation de l'ouverture de session PPP la plus récente.

```
journalctl -n 100 -f -u pppoe-server.service
```

```
hub pppoe-server[610]: Session 1 created for client b8:ad:ca:fe:00:06 (10.4.41.2) on enp0s1.441 using Service-Name ''
hub pppd[610]: pppd 2.5.0 started by root, uid 0
hub pppd[610]: using channel 2
hub pppd[610]: Using interface ppp0
hub pppd[610]: Connect: ppp0 <--> /dev/pts/0
hub pppd[610]: rcvd [LCP ConfReq id=0x1 <mru 1492> <magic 0xbf826095>]
hub pppd[610]: sent [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
hub pppd[610]: sent [LCP ConfAck id=0x1 <mru 1492> <magic 0xbf826095>]
hub pppd[610]: rcvd [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0x471e9bda>]
hub pppd[610]: sent [LCP EchoReq id=0x0 magic=0x471e9bda]
hub pppd[610]: sent [EAP Request id=0xa1 Identity <Message "Name">]
hub pppd[610]: rcvd [LCP EchoReq id=0x0 magic=0xbf826095]
hub pppd[610]: sent [LCP EchoRep id=0x0 magic=0x471e9bda]
hub pppd[610]: rcvd [LCP EchoRep id=0x0 magic=0xbf826095]
hub pppd[610]: rcvd [EAP Response id=0xa1 Identity <Name "spoke_site0">]
hub pppd[610]: EAP: unauthenticated peer name "spoke_site0"
hub pppd[610]: EAP id=0xa1 'Identify' -> 'MD5Chall'
hub pppd[610]: sent [EAP Request id=0xa2 MD5-Challenge <Value 81 3b a7 d1 eb 86 42 15 2c b9 1b 07 83 98 e2 dd b7 c6 57 b4 b5 0f> <
hub pppd[610]: rcvd [EAP Response id=0xa2 MD5-Challenge <Value 16 e7 3d fa d2 4b 6a 73 41 5f 86 c8 84 97 ed f0> <Name "spoke_site0
hub pppd[610]: sent [EAP Success id=0xa3]
hub pppd[610]: peer from calling number b8:ad:ca:fe:00:06 authorized
hub pppd[610]: sent [CCP ConfReq id=0x1 <bsd v1 15>]
hub pppd[610]: sent [IPCP ConfReq id=0x1 <addr 10.4.41.1>]
hub pppd[610]: sent [IPV6CP ConfReq id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
hub pppd[610]: EAP id=0xa3 'MD5Chall' -> 'Open'
hub pppd[610]: rcvd [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
hub pppd[610]: sent [IPCP ConfNak id=0x1 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[610]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::9869:8831:104a:570a>]
hub pppd[610]: sent [IPV6CP ConfAck id=0x1 <addr fe80::9869:8831:104a:570a>]
hub pppd[610]: rcvd [CCP ConfReq id=0x1]
hub pppd[610]: sent [CCP ConfAck id=0x1]
hub pppd[610]: rcvd [CCP ConfRej id=0x1 <bsd v1 15>]
hub pppd[610]: sent [CCP ConfReq id=0x2]
hub pppd[610]: rcvd [IPCP ConfAck id=0x1 <addr 10.4.41.1>]
hub pppd[610]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::61d3:68e0:1e34:e123>]
hub pppd[610]: local  LL address fe80::61d3:68e0:1e34:e123
hub pppd[610]: remote LL address fe80::9869:8831:104a:570a
hub pppd[610]: Script /etc/ppp/ipv6-up started (pid 616)
hub pppd[610]: rcvd [IPCP ConfReq id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[610]: sent [IPCP ConfAck id=0x2 <addr 10.4.41.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[610]: Script /etc/ppp/ip-pre-up started (pid 619)
hub pppd[610]: Script /etc/ppp/ip-pre-up finished (pid 619), status = 0x0
hub pppd[610]: local  IP address 10.4.41.1
hub pppd[610]: remote IP address 10.4.41.2
hub pppd[610]: Script /etc/ppp/ip-up started (pid 623)
hub pppd[610]: Script /etc/ppp/ipv6-up finished (pid 616), status = 0x0
hub pppd[610]: rcvd [CCP ConfAck id=0x2]
hub pppd[610]: Script /etc/ppp/ip-up finished (pid 623), status = 0x0
```

# 11. Pour conclure...

Ce support de travaux pratiques a permis d'explorer en détail la mise en œuvre d'une topologie réseau complexe combinant routage inter-VLAN, protocole PPPoE et conteneurisation. La démarche proposée permet de configurer pas à pas les différents éléments, des routeurs virtuels jusqu'aux conteneurs système.

L'automatisation des tâches est introduite avec l'utilisation de scripts pour la configuration réseau et le déploiement des conteneurs. Cela permet de découvrir les pratiques actuelles en matière d'administration système et réseau dans un contexte d'infrastructure as code.

Enfin, toutes ces manipulations offrent une expérience pratique des protocoles utilisés sur les réseau étendus comme PPPoE, tout en illustrant l'adressage IPv6 et la virtualisation réseau.