Topologie Hub & Spoke avec le protocole PPPoE

Philippe Latu
philippe.latu(at)inetdoc.net

https://www.inetdoc.net

Résumé

Ce support de travaux pratiques est une illustration d'une topologie réseau classique baptisée *Hub & Spoke*. Le *Hub* est un routeur qui concentre tous les flux des routeurs d'extrémités appelés *Spoke*. Les liaisons entre le *Hub* et les routeurs *Spoke* sont point à point et utilisent le protocole PPP. Avec la généralisation de la fibre optique dans les réseaux étendus (WAN), on doit encapsuler les trames PPP dans un VLAN Ethernet à l'aide de la technologie PPPoE.

Les manipulations proposées reprennent en grande partie celles du support *Routage inter-VLAN et protocole PPPoE* en les adaptant à la topologie en triangle.

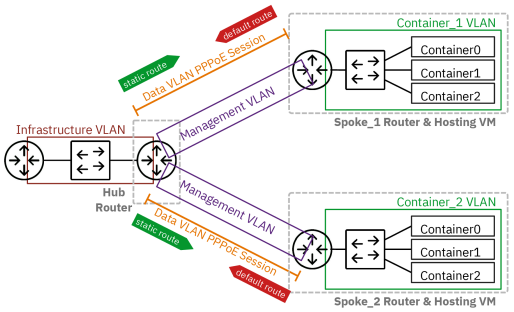


Table des matières

1. Copyright et Licence

Méta-information

Cet article est écrit avec *DocBook* XML sur un système *Debian GNU/Linux*. Il est disponible en version imprimable au format PDF : `hub-and-spoke.pdf`.

2. Objectifs

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

1. Configurer une topologie réseau Hub & Spoke utilisant le protocole PPPoE entre le routeur concentrateur (rôle ***Hub***) et les routeurs d'extrémité (rôle ***Spoke***).
2. Établir et gérer des sessions PPP entre les routeurs, en configurant l'authentification et l'attribution d'adresses IP.
3. Mettre en place un routage complet IPv4 et IPv6 entre les différents sites de la topologie, y compris la configuration de routes statiques.
4. Déployer et gérer des conteneurs sur les routeurs ***Spoke***, en utilisant le gestionnaire Incus.
5. Tester et valider la connectivité réseau à travers la topologie, en utilisant des outils comme ping, tracepath et en configurant des services web basiques.

3. Topologie Hub & Spoke - Protocole PPPoE

Le Protocole Point à Point (PPP) est utilisé pour établir une communication directe entre deux hôtes. Il relie deux routeurs de façon logique au dessus d'une topologie de réseau physique qui peut comprendre divers composants et différentes technologies. Il permet aux deux extrémités en communication de négocier des paramètres de transmission tels que l'authentification et l'attribution d'adresses de couche réseau. Dans ce document, on utilise le protocole PPP au dessus d'un réseau Ethernet qui représente la technologie mise en œuvre par un opérateur Internet.

Comme un Ethernet est par définition un réseau de diffusion, il est nécessaire d'introduire un protocole intermédiaire appelé PPPoE qui sert à identifier les deux hôtes de la liaison logique point à point.

Ce support met en œuvre une “figure“ classique appelée topologie ***Hub & Spoke***. Voici une description des rôles des différents routeurs de cette topologie.

Hub

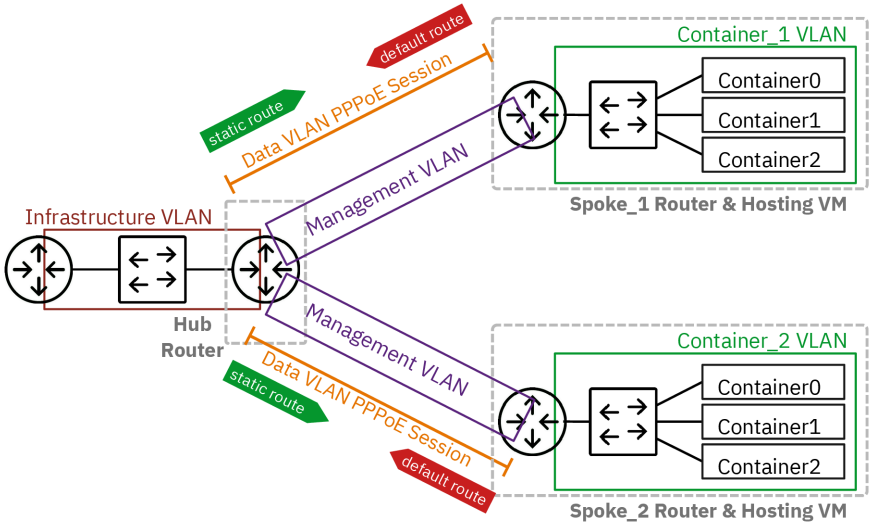
Traduit mot à mot, le rôle ***Hub*** correspond à un concentrateur. Il concentre tous les flux réseau des routeurs qui ont le rôle ***Spoke***. En effet, les échanges entre deux routeurs ***Spoke*** doivent passer par le routeur ***Hub***.

On lui attribue aussi la fonction de ***Broadband Remote Access Server*** ou BRAS. Dans notre contexte, cette fonction se caractérise par le fait que ce routeur détient le plan d'adressage. C'est lui qui a la responsabilité de délivrer les adresses IP lors de l'initiation de la session PPP.

Spoke

Le rôle ***Spoke*** correspond à un réseau d'extrémité au delà duquel on ne trouve aucune interconnexion. Le routeur ***Spoke*** doit s'adresser au routeur Hub dès qu'il veut acheminer un flux réseau. Il s'agit bien d'un routeur d'extrémité qui ne dispose d'aucun chemin alternatif pour joindre l'Internet.

Dans les réseaux domestiques, la «box» correspond bien au rôle ***Spoke*** dans la mesure où elle se voit attribuer des adresses IPv4 et IPv6 publiques par le fournisseur d'accès Internet. Les seules informations qu'elle détient sont les authentifiants du client de l'opérateur.



Topologie entre deux routeurs ***Hub*** et ***Spoke*** avec PPPoE

Comme le montre le graphique ci-dessus, l'opérateur distingue 4 réseaux ou VLANs différents.

Réseau d'infrastructure (VLAN rouge)

Ce raccordement réseau donne l'accès à Internet. Il représente la dorsale de l'opérateur.

Management/Supervision (VLAN violet)

Ce réseau correspond à la gestion des équipements et à la supervision des liens en fibre optique. Il n'utilise que des adresses de lien local IPv6. Il représente le rôle de l'exploitant des chemins de câbles.

Réseau fournisseur d'accès Internet (VLAN orange)

C'est sur ce réseau que la session PPP est établie. Le site distant de l'entreprise cliente (rôle ***Spoke***) s'authentifie auprès du BRAS de l'opérateur et obtient en échange une adresse IPv4 et une adresse IPv6 qui permettent d'accéder au site central (rôle ***Hub***) et à Internet.

Réseau d'hébergement de site distant (VLAN vert)

C'est le réseau des services hébergés sur son propre site par l'entreprise cliente de l'opérateur. Ici, on choisit de déployer plusieurs conteneurs pour illustrer plusieurs hôtes ou serveurs dont le trafic doit transiter uniquement par le site central (rôle ***Hub***) via la session PPP.

Voici le plan d'adressage de la maquette utilisée pour rédiger ce support de travaux pratiques.

Tableau 1. Plan d'adressage de la maquette « Topologie Hub & Spoke et protocole PPPoE »

Rôle	VLAN	Numéro	Type	Destination	Adresse/Authentification
Hub bleu	Rouge	360	Passerelle	-	`192.168.104.129/29` `fe80:168::1/64`
	Violet	440	Lien local	Spoke1	`fe80:1b8::1/64`
	Orange	441	Point à point	Spoke1	`10.44.1.1:10.44.1.2`
	Violet	442	Lien local	Spoke2	`fe80:1ba::1/64`
	Orange	443	Point à point	Spoke2	`10.44.3.1:10.44.3.2`
Spoke1 Vert	Violet	440	Lien local	Hub	`fe80:1b8::2/64`
	Orange	441	Authentifiants	Hub	`spoke_site1 / 0r4ng3_1`

	Vert	10	Passerelle	-	`10.0.10.1/24`
					`fda0:7a62:a::1/64`
					`fe80:a::1/64`

Spoke2 Vert	Violet	442	Lien local	Hub	`fe80:1ba::2/64`
	Orange	443	Authentifiants	Hub	`spoke_site2 / 0r4ng3_2`

	Vert	20	Passerelle	-	`10.0.20.1/24`
					`fda0:7a62:14::1/64`
					`fe80:14::1/64`

4. Routeur Hub

Le rôle du routeur ***Hub*** est d'interconnecter un réseau de collecte opérateur (LAN) qui donne accès à l'Internet et plusieurs réseaux étendus (WAN) de raccordement de sites distants. Le routeur ***Hub*** assure aussi la fonction ***Broadband Remote Access Server*** (BRAS). C'est la raison pour laquelle il détient les adresses IPv4 et IPv6 à attribuer aux routeurs d'extrémité appelés ***Spoke***.

Le routeur ***Hub*** doit aussi gérer l'encapsulation des trames PPP sur un réseau de diffusion Ethernet.

Cette section reprend essentiellement la partie ***Routeur Hub*** du support *Routage inter-VLAN et protocole PPPoE*. On doit cependant adapter la configuration des interfaces réseau du routeur ***Hub*** en ajoutant un second jeu d'interfaces pour le deuxième site distant.

Q1. 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/10-routing.conf`.

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

Attention ! N'oubliez pas d'appliquer les nouvelles valeurs des paramètres de configuration.

```
sudo sysctl --system
```

Q2. Comment assurer la traduction d'adresses sources pour tous les flux réseaux sortants 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
```

Il faut aussi activer ce service pour assurer le chargement automatique des règles de filtrage au démarrage.

```
sudo systemctl enable --now nftables.service
sudo systemctl status nftables.service
```

Q3. 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
```

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

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

Q4. 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_site1" *  "0r4ng3_1"  *
"spoke_site2" *  "0r4ng3_2"  *
```

Q5. 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.

Ce fichier contient tous les paramètres communs aux deux démons `pppd` qui sont lancés via pppoe-server. Voici comment créer le fichier.

```
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
```

Q6. Comment créer les comptes utilisateurs locaux sur le routeur ***Hub*** sachant qu'ils ne sont autorisés ni à se connecter ni à avoir un répertoire personnel ?

Consulter les options de la commande adduser.

Voici un exemple dans le contexte de la maquette.

```
sudo adduser --gecos 'Spoke 1' --disabled-login --no-create-home spoke_site1
```

```
sudo adduser --gecos 'Spoke 2' --disabled-login --no-create-home spoke_site2
```

Q7. Quel paramètre supplémentaire doit être ajouté à la configuration de la commande pppoe-server pour distinguer les échanges entre les deux routeurs ***Spoke*** ?

Relativement au support *Routage inter-VLAN et protocole PPPoE*, il est essentiel de définir correctement les routes statiques vers les réseaux d'extrémité de chaque routeur ***Spoke***.

Consulter les options de la commande pppoe-server.

L'option `-u` permet de désigner une “unité“ qui sert à nommer l'interface. Par exemple, `-u 0` correspond à l'interface `ppp0`.

Q8. Comment créer deux nouvelles unités systemd responsables du lancement des 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.

Voici un exemple de création du fichier d'unité systemd pour le premier service.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server1.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-server1.service)" = "0" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.441 -C BRAS -L 10.44.1.1 -R 10.44.1.2 -N 1 -u 0
Restart=on-failure
RestartSec=5

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

Voici un exemple de création du fichier d'unité systemd pour le second service.

```
cat << 'EOF' | sudo tee /etc/systemd/system/pppoe-server2.service
[Unit]
Description=PPPoE Server
After=systemd-networkd.service
Wants=systemd-networkd.service
BindsTo=sys-subsystem-net-devices-enp0s1.443.device
After=sys-subsystem-net-devices-enp0s1.443.device

[Service]
Type=forking
ExecCondition=/bin/sh -c '[ "$(systemctl show --property MainPID --value pppoe-server2.service)" = "0" ]'
ExecStart=/usr/sbin/pppoe-server -I enp0s1.443 -C BRAS -L 10.44.3.1 -R 10.44.3.2 -N 1 -u 1
Restart=on-failure
RestartSec=5

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

Q9. Comment activer les deux nouveaux services et contrôler leur 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 les nouveaux services.

```
for i in {1..2}; do sudo systemctl enable pppoe-server$i.service; done
```

On lance ce nouveau service.

```
for i in {1..2}; do sudo systemctl start pppoe-server$i.service; done
```

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

```
for i in {1..2}; do systemctl status pppoe-server$i.service; done
```

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.

5. Routeurs Spoke

Dans le scénario défini dans la Section 3, « Topologie Hub & Spoke - Protocole PPPoE », un routeur de site d'extrémité ou ***Spoke*** ne peut accéder aux autres réseaux que via le routeur ***Hub***. Son interface WAN joue donc le rôle de route par défaut pour le réseau local des hôtes hébergé sur un site distant.

Cette section, comme la précédente, reprend les éléments du support *Routage inter-VLAN et protocole PPPoE* en dédoublant les routeurs d'extrémité.

Les routeurs ***Spoke*** utilisent un démon `pppd` dans le VLAN `Data` (Orange) pour établir une session PPP avec le routeur ***Hub***.

Avant d'aborder les questions ci-dessous, il faut s'assurer que :

- Le nom d'hôte est correctement attribué sur chaque routeur ***Spoke***.
  - `sudo hostnamectl hostname spoke_1`
  - `sudo hostnamectl hostname spoke_2`
- Le routage est configuré et activé.

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

```
sudo sysctl --system
```

- Les paquets `ppp` sont installés sur chaque routeur ***Spoke*** via l'accès réseau temporaire. Il faut donc éditer et appliquer les modifications faites dans le fichier `/etc/netplan/enp0s1.yaml`.

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

- Le fichier `/etc/ppp/chap-secrets` contenant les authentifiants pour l'établissement de la session PPP est complété.
  - ```
    # Secrets for authentication using CHAP
    # client        server  secret          IP addresses
    "spoke_site1"   *       "0r4ng3_1"      *
    ```
  - ```
    # Secrets for authentication using CHAP
    # client        server  secret          IP addresses
    "spoke_site2"   *       "0r4ng3_2"      *
    ```
- Le fichier `/etc/ppp/peers/pppoe-provider` de définition du profil de session PPP est créé.

> **Avertissement**
>
> Le nom d'utilisateur doit correspondre à l'entrée du fichier `/etc/ppp/chap-secrets` !
>
> Le numéro de VLAN de la sous-interface doit désigner le bon côté du triangle de la topologie !

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

# 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.VVV

# 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
```

- Les unités `systemd` sont créées et activées sur chaque routeur ***Spoke***.

> **Avertissement**
>
> Le numéro de VLAN de la sous-interface doit désigner le bon côté du triangle de la topologie !

```
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.VVV.device
After=sys-subsystem-net-devices-enp0s1.VVV.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
```

On active les nouvelles unités de service avec les instructions suivantes.

```
sudo systemctl daemon-reload
sudo systemctl enable --now ppp.service
```

- Une fois la session PPP établie, n'oubliez pas de désactiver l'accès réseau temporaire et s'assurer que c'est bien cette session qui sert de route par défaut pour accéder à tous les autres réseaux.

  Il faut donc éditer à nouveau et appliquer les modifications faites dans le fichier `/etc/netplan/enp0s1.yaml`.

  Voici un exemple de test lancé sur le second routeur ***Spoke*** de la maquette.

```
ip route get 9.9.9.9
9.9.9.9 dev ppp0 src 10.44.3.2 uid 1000
    cache
```

- On doit éditer le fichier `/etc/systemd/resolved.conf` sur chaque routeur ***Spoke*** pour affecter directement l'adresse de résolution DNS.

> ⚠ 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.

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

N'oubliez pas de relancer le service pour prendre en compte les modifications du fichier.

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

À ce stade de la configuration, les sessions PPP des deux routeurs ***Spoke*** sont en place et on peut analyser les messages présents dans les journaux système et identifier les traitements réalisés par les différentes fonctions des protocoles.

Q10. Comment vérifier que le protocole PPPoE a bien permis d'identifier les extrémités en communication dans le réseau de diffusion (VLAN orange) avant de lancer l'ouverture de session PPP ?

Rechercher les options de la commande journalctl pour faire afficher les messages utiles de la journalisation système.

Dans le but de minimiser le nombre de lignes affichées, on peut combiner les commandes journalctl et grep. Les possibilités sont très diverses. Voici un exemple côté routeur Spoke.

```
journalctl --grep pppoe | grep -i pad
```

```
spoke2 pppd[489]: Send PPPOE Discovery V1T1 PADI session 0x0 length 12
spoke2 pppd[489]: Recv PPPOE Discovery V1T1 PADO session 0x0 length 44
spoke2 pppd[489]: Send PPPOE Discovery V1T1 PADR session 0x0 length 36
spoke2 pppd[489]: Recv PPPOE Discovery V1T1 PADS session 0x1 length 12
```

L'extrait ci-dessus montre la séquence spécifique au protocole PPPoE :

- Découverte avec émission de la trame PADI depuis le routeur ***Spoke***.
- Offre avec réception de la trame PADO depuis le routeur ***Hub***.
- Requête avec émission de la trame PADR depuis le routeur ***Spoke*** pour accepter l'offre.
- Ouverture de session avec la trame PADS quand les deux extrémités de la liaison point à point sont en accord.

Côté routeur ***Hub***, les journaux de chaque unité de service `pppoe-serverX` permettent de vérifier que les adresses MAC correspondent bien au bon routeur ***Spoke*** pour chaque côté du triangle de la topologie.

Voici deux exemples pour la maquette de rédaction de ce document.

```
journalctl -u pppoe-server1.service -n 100  | grep created
```

```
hub pppoe-server[621]: Session 1 created for client b8:ad:ca:fe:00:06 (10.44.1.2) on enp0s1.441 using Service-Name ''
```

```
journalctl -u pppoe-server2.service -n 100  | grep created
```

```
hub pppoe-server[673]: Session 1 created for client b8:ad:ca:fe:00:07 (10.44.3.2) on enp0s1.443 using Service-Name ''
```

Q11. Quels sont les messages des journaux système qui montrent que la session PPP a bien été établie ?

Après avoir consulté la page *Point-to-Point Protocol* repérer les messages relatifs aux deux sous-couches LCP et NCP du protocole PPP.

Les messages PPP sont présents sur tous les routeurs de la topologie. Voici deux exemples prélevés côté ***Hub*** et côté ***Spoke***.

- extrait des négociations de paramètres et de l'authentification dans la sous-couche LCP côté ***Hub***.

```
journalctl -u pppoe-server2.service -n 100  | grep -E '(LCP|EAP)'
```

```
hub pppd[673]: sent [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
hub pppd[673]: rcvd [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
hub pppd[673]: rcvd [LCP ConfReq id=0x1 <mru 1492> <magic 0x8b9f1855>]
hub pppd[673]: sent [LCP ConfAck id=0x1 <mru 1492> <magic 0x8b9f1855>]
hub pppd[673]: sent [LCP EchoReq id=0x0 magic=0xef9c0d86]
hub pppd[673]: sent [EAP Request id=0x95 Identity <Message "Name">]
hub pppd[673]: rcvd [LCP EchoReq id=0x0 magic=0x8b9f1855]
hub pppd[673]: sent [LCP EchoRep id=0x0 magic=0xef9c0d86]
hub pppd[673]: rcvd [LCP EchoRep id=0x0 magic=0x8b9f1855]
hub pppd[673]: rcvd [EAP Response id=0x95 Identity <Name "spoke_site2">]
hub pppd[673]: EAP: unauthenticated peer name "spoke_site2"
hub pppd[673]: EAP id=0x95 'Identify' -> 'MD5Chall'
hub pppd[673]: sent [EAP Request id=0x96 MD5-Challenge <Value 14 d4 e2 ec 00 a1 26 ca f8 98 c5 7e d6 f4 a7 ef d7> <Name
hub pppd[673]: rcvd [EAP Response id=0x96 MD5-Challenge <Value cd 5d 07 6a 5b 59 2b 1c ec f8 d6 7f 9b 40 44 63> <Name "s
hub pppd[673]: sent [EAP Success id=0x97]
hub pppd[673]: EAP id=0x97 'MD5Chall' -> 'Open'
```

Dans l'exemple ci-dessus, on repère immédiatement le dernier message qui conclue la phase d'authentification du routeur ***Spoke*** auprès du routeur ***Hub*** avec succès.

Plus haut, on repère aussi l'identité utilisée pour cette authentification : `spoke_site2`.

- Extrait des mêmes négociations de paramètres et de l'authentification dans la sous-couche LCP côté *Spoke*.

```
journalctl -u ppp -n 100 | grep -E '(LCP|EAP)'
```

```
spoke2 pppd[489]: sent [LCP ConfReq id=0x1 <mru 1492> <magic 0x8b9f1855>]
spoke2 pppd[489]: rcvd [LCP ConfReq id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
spoke2 pppd[489]: sent [LCP ConfAck id=0x1 <mru 1492> <auth eap> <magic 0xef9c0d86>]
spoke2 pppd[489]: sent [LCP ConfReq id=0x1 <mru 1492> <magic 0x8b9f1855>]
spoke2 pppd[489]: rcvd [LCP ConfAck id=0x1 <mru 1492> <magic 0x8b9f1855>]
spoke2 pppd[489]: sent [LCP EchoReq id=0x0 magic=0x8b9f1855]
spoke2 pppd[489]: rcvd [LCP EchoReq id=0x0 magic=0xef9c0d86]
spoke2 pppd[489]: sent [LCP EchoRep id=0x0 magic=0x8b9f1855]
spoke2 pppd[489]: rcvd [EAP Request id=0x95 Identity <Message "Name">]
spoke2 pppd[489]: EAP: Identity prompt "Name"
spoke2 pppd[489]: sent [EAP Response id=0x95 Identity <Name "spoke_site2">]
spoke2 pppd[489]: rcvd [LCP EchoRep id=0x0 magic=0xef9c0d86]
spoke2 pppd[489]: rcvd [EAP Request id=0x96 MD5-Challenge <Value 14 d4 e2 ec 00 a1 26 ca f8 98 c5 7e d6 f4 a7 ef d7> <Na
spoke2 pppd[489]: sent [EAP Response id=0x96 MD5-Challenge <Value cd 5d 07 6a 5b 59 2b 1c ec f8 d6 7f 9b 40 44 63> <Name
spoke2 pppd[489]: rcvd [EAP Success id=0x97]
spoke2 pppd[489]: EAP authentication succeeded
```

Comme dans l'extrait précédent, on retrouve les traces de l'identité utilisée et du succès de l'authentification.

- On reprend le même travail pour la sous-couche NCP dans laquelle on recherche l'attribution des adresses des liaisons point à point côté *Hub*.

  Dans les journaux, la lettre `N` est remplacée par IP pour le protocole IPv4 et IPV6 pour le protocole IPv6.

```
journalctl -u pppoe-server1.service -n 100  | grep -E '(IP.*CP|CCP)'
```

```
pppd[1055]: peer from calling number b8:ad:ca:fe:00:06 authorized
hub pppd[1055]: sent [CCP ConfReq id=0x1 <bsd v1 15>]
hub pppd[1055]: sent [IPCP ConfReq id=0x1 <addr 10.44.1.1>]
hub pppd[1055]: sent [IPV6CP ConfReq id=0x1 <addr fe80::0421:e270:de27:332c>]
hub pppd[1055]: EAP id=0xaa 'MD5Chall' -> 'Open'
hub pppd[1055]: rcvd [IPCP ConfReq id=0x3 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
hub pppd[1055]: sent [IPCP ConfNak id=0x3 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[1055]: rcvd [IPV6CP ConfReq id=0x2 <addr fe80::79ea:402e:6158:8922>]
hub pppd[1055]: sent [IPV6CP ConfAck id=0x2 <addr fe80::79ea:402e:6158:8922>]
hub pppd[1055]: rcvd [CCP ConfReq id=0x2]
hub pppd[1055]: sent [CCP ConfAck id=0x2]
hub pppd[1055]: rcvd [CCP ConfRej id=0x1 <bsd v1 15>]
hub pppd[1055]: sent [CCP ConfReq id=0x2]
hub pppd[1055]: rcvd [IPCP ConfAck id=0x1 <addr 10.44.1.1>]
hub pppd[1055]: rcvd [IPV6CP ConfAck id=0x1 <addr fe80::0421:e270:de27:332c>]
hub pppd[1055]: local  LL address fe80::0421:e270:de27:332c
hub pppd[1055]: remote LL address fe80::79ea:402e:6158:8922
hub pppd[1055]: Script /etc/ppp/ipv6-up started (pid 1061)
hub pppd[1055]: rcvd [IPCP ConfReq id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[1055]: sent [IPCP ConfAck id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
hub pppd[1055]: Script /etc/ppp/ip-pre-up started (pid 1062)
hub pppd[1055]: Script /etc/ppp/ip-pre-up finished (pid 1062), status = 0x0
hub pppd[1055]: local  IP address 10.44.1.1
hub pppd[1055]: remote IP address 10.44.1.2
hub pppd[1055]: Script /etc/ppp/ip-up started (pid 1066)
hub pppd[1055]: Script /etc/ppp/ipv6-up finished (pid 1061), status = 0x0
hub pppd[1055]: rcvd [CCP ConfAck id=0x2]
hub pppd[1055]: Script /etc/ppp/ip-up finished (pid 1066), status = 0x0
```

- On poursuit la même démarche côté routeurs *Spoke*.

Le point important ici, c'est relever le statut des scripts d'application des routes par défaut vers la liaison PPP pour les routeurs ***Spoke***. En l'état actuel de la configuration, l'attribution de la route par défaut IPv6 ne se fait pas. Il faudra donc corriger ça dans la partie suivante.

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

```
pppd[496]: EAP authentication succeeded
spoke1 pppd[496]: peer from calling number B8:AD:CA:FE:00:05 authorized
spoke1 pppd[496]: sent [IPCP ConfReq id=0x3 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
spoke1 pppd[496]: sent [IPV6CP ConfReq id=0x2 <addr fe80::79ea:402e:6158:8922>]
spoke1 pppd[496]: rcvd [CCP ConfReq id=0x1 <bsd v1 15>]
spoke1 pppd[496]: sent [CCP ConfReq id=0x2]
spoke1 pppd[496]: sent [CCP ConfRej id=0x1 <bsd v1 15>]
spoke1 pppd[496]: rcvd [IPCP ConfReq id=0x1 <addr 10.44.1.1>]
spoke1 pppd[496]: sent [IPCP ConfAck id=0x1 <addr 10.44.1.1>]
spoke1 pppd[496]: rcvd [IPV6CP ConfReq id=0x1 <addr fe80::0421:e270:de27:332c>]
spoke1 pppd[496]: sent [IPV6CP ConfAck id=0x1 <addr fe80::0421:e270:de27:332c>]
spoke1 pppd[496]: rcvd [IPCP ConfNak id=0x3 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke1 pppd[496]: sent [IPCP ConfReq id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke1 pppd[496]: rcvd [IPV6CP ConfAck id=0x2 <addr fe80::79ea:402e:6158:8922>]
spoke1 pppd[496]: local  LL address fe80::79ea:402e:6158:8922
spoke1 pppd[496]: remote LL address fe80::0421:e270:de27:332c
spoke1 pppd[496]: Script /etc/ppp/ipv6-up started (pid 1640)
spoke1 pppd[496]: rcvd [CCP ConfAck id=0x2]
spoke1 pppd[496]: rcvd [CCP ConfReq id=0x2]
spoke1 pppd[496]: sent [CCP ConfAck id=0x2]
spoke1 pppd[496]: rcvd [IPCP ConfAck id=0x4 <addr 10.44.1.2> <ms-dns1 172.16.0.2> <ms-dns2 172.16.0.2>]
spoke1 pppd[496]: Script /etc/ppp/ip-pre-up started (pid 1642)
spoke1 pppd[496]: Script /etc/ppp/ip-pre-up finished (pid 1642), status = 0x0
spoke1 pppd[496]: local  IP address 10.44.1.2
spoke1 pppd[496]: remote IP address 10.44.1.1
spoke1 pppd[496]: primary   DNS address 172.16.0.2
spoke1 pppd[496]: secondary DNS address 172.16.0.2
spoke1 pppd[496]: Script /etc/ppp/ip-up started (pid 1645)
spoke1 pppd[496]: Script /etc/ppp/ipv6-up finished (pid 1640), status = 0x0
spoke1 pppd[496]: Script /etc/ppp/ip-up finished (pid 1645), status = 0x0
```

6. Interconnexion IPv4 et IPv6

À ce stade des manipulations, les routeurs ***Spoke*** utilisent leur session PPP et le routage IPv4 pour accéder à tous les réseaux.

L'objectif de cette section est de valider les communications entre les routeurs ***Spoke***, aussi bien avec le protocole réseau IPv4 qu'avec le protocole réseau IPv6.

Il s'agit donc de configurer les routes statiques vers les réseaux d'hébergement des deux sites distants côté routeur ***Hub*** et de valider ou d'ajouter les routes par défaut côté routeurs ***Spoke***.

IPv6 nécessite une attention particulière. En effet, à la suite de l'établissement de session PPP, on ne dispose que d'adresses de lien local. Ces adresses ne permettent pas d'acheminer du trafic vers un autre réseau quel qu'il soit. Il est donc nécessaire de mettre en place des interfaces commutées virtuelles (SVI) avec des adresses IPv6 de type ULA (***Unique Local Address***) pour être en mesure d'échanger du trafic entre des réseaux différents.

Q12. Comment connaître l'état actuel des communications entre les routeurs de la topologie ***Hub and Spoke*** ?

On doit lancer une série de tests ICMP pour chaque protocole de couche réseau et identifier les dysfonctionnements.

Protocole IPv4

Commençons par les tests des routeurs ***Spoke*** vers Internet. Comme les sessions PPP sont établies à la suite de la partie précédente, tous les paquets transitent avec succès.

```
etu@spoke1:~$ ping -qc2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 29.534/35.666/41.798/6.132 ms
```

```
etu@spoke2:~$ ping -qc2 9.9.9.9
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.

--- 9.9.9.9 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 29.515/29.622/29.730/0.107 ms
```

Passons aux communications entre routeurs ***Spoke*** entre les extrémités des liaisons point à point. Là aussi, tout fonctionne puisque le routeur ***Hub*** détient le plan d'adressage.

```
etu@spoke2:~$ ip addr ls dev ppp0
5: ppp0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1492 qdisc fq_codel state UNKNOWN group default qlen 3
    link/ppp
    inet 10.44.3.2 peer 10.44.3.1/32 scope global ppp0
       valid_lft forever preferred_lft forever
    inet6 fe80::5826:d9cb:d027:4cc9 peer fe80::552:a7c:9fba:55af/128 scope link nodad
       valid_lft forever preferred_lft forever
```

```
etu@spoke2:~$ ping -qc2 10.44.1.2
PING 10.44.1.2 (10.44.1.2) 56(84) bytes of data.

--- 10.44.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.249/2.364/2.479/0.115 ms
```

```
etu@hub:~$ ip route ls
default via 192.168.104.129 dev enp0s1.360 proto static
10.44.1.2 dev ppp0 proto kernel scope link src 10.44.1.1
10.44.3.2 dev ppp1 proto kernel scope link src 10.44.3.1
192.168.104.128/29 dev enp0s1.360 proto kernel scope link src 192.168.104.130
```

Protocole IPv6

Pour IPv6, la configuration est clairement incomplète dans la mesure où il est impossible d'acheminer du trafic au-delà du lien local de chaque routeur ***Spoke***.

```
etu@spoke1:~$ ip -6 route get 2620:fe::fe
RTNETLINK answers: Network is unreachable
```

```
etu@spoke1:~$ ip -6 route ls default
```

La route par défaut n'existe pas.

Pour conclure ces tests, nous devons mettre en place les réseaux d'hébergement de chaque routeur ***Spoke*** ainsi que les routes statiques pour avancer dans l'interconnexion des sites distants.

Q13. Comment créer les routes par défaut IPv4 et IPv6 sur les deux routeurs ***Spoke*** ?

Créer les scripts exécutables `defaultroute` pour chaque protocole réseau et renouveler les sessions PPP.

Pour IPv4, c'est le répertoire est `/etc/ppp/ip-up.d/` qui doit contenir le script exécutable `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, c'est le répertoire est `/etc/ppp/ipv6-up.d/` qui doit contenir le script exécutable appelé `defaultroute`.

```
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
```

Comme demandé, pas oublier de relancer les sessions PPP pour provoquer la création des routes par défaut sur chaque routeur ***Spoke***.

```
sudo systemctl restart ppp.service
```

Une fois les sessions PPP renouvelées, on dispose d'une solution pour router les paquets IPv6 vers les autres réseaux.

```
ip -6 route get 2620:fe::fe
2620:fe::fe from :: dev ppp0 src fda0:7a62:14::1 metric 1024 pref medium
```

Malheureusement, ce n'est pas suffisant pour acheminer du trafic depuis le routeur ***Hub***. Celui-ci n'a aucune connaissance des adresses réseau des réseaux d'hébergement des routeurs ***Spoke***. L'ajout des routes statiques vers ces réseaux 'hébergement est justement l'objet de la question suivante.

Q14. Comment créer les routes statiques vers les réseaux d'hébergement sur le routeur ***Hub*** ?

Créer un script exécutable par protocole réseau dans lequel on utilise les noms d'interfaces PPP dérivés des options `-u` utilisées dans les paramètres de configuration des deux serveurs PPPoE.

Pour IPv4, c'est le répertoire est `/etc/ppp/ip-up.d/` qui doit contenir le script exécutable `staticroute`.

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

if [ -z "${CONNECT_TIME}" ]; then
    case "${PPP_IFACE}" in
        "ppp0")
            ip route add 10.0.10.0/24 dev ${PPP_IFACE}
            ;;
        "ppp1")
            ip route add 10.0.20.0/24 dev ${PPP_IFACE}
            ;;
    esac
fi
EOF
```

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

Pour IPv6, c'est le répertoire est `/etc/ppp/ipv6-up.d/` qui doit contenir le script exécutable appelé `staticroute`.

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

if [ -z "${CONNECT_TIME}" ]; then
    case "${PPP_IFACE}" in
        "ppp0")
            ip -6 route add fda0:7a62:a::/64 dev ${PPP_IFACE}
            ;;
        "ppp1")
            ip -6 route add fda0:7a62:14::/64 dev ${PPP_IFACE}
            ;;
    esac
fi
EOF
```

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

Une fois de plus, il faut relancer les sessions PPP pour observer les résultats et lancer les tests ICMP.

Auparavant, on peut afficher les tables de routages IPv4 et IPv6 du routeur ***Hub*** pour vérifier la présence des nouvelles routes statiques.

- En direction du premier routeur ***Spoke***.

```
ip route ls dev ppp0
```

```
10.0.10.0/24 scope link
10.44.1.2 proto kernel scope link src 10.44.1.1
```

```
ip -6 route ls dev ppp0
```

```
fda0:7a62:a::/64 metric 1024 pref medium
fe80::2c4a:3bb9:eead:8fa5 proto kernel metric 256 pref medium
fe80::cca6:346e:80d0:20cf proto kernel metric 256 pref medium
```

- En direction du second routeur ***Spoke***.

```
ip route ls dev ppp1
```

```
10.0.20.0/24 scope link
10.44.3.2 proto kernel scope link src 10.44.3.1
```

```
ip -6 route ls dev ppp1
```

```
fda0:7a62:14::/64 metric 1024 pref medium
fe80::b4d6:f38c:a930:c8f5 proto kernel metric 256 pref medium
fe80::f0fe:10bd:88f0:cb26 proto kernel metric 256 pref medium
```

Q15. Comment créer et configurer les interfaces commutées virtuelles sur chaque routeur ***Spoke*** ?

Après avoir installé le paquet `openvswitchi-switch`, compléter le fichier de configuration netplan.io pour déclarer un commutateur applé `asw-host` et l'interface SVI correspondant au réseau d'hébergement de site (VLAN vert)

On installe le paquet du commutateur virtuel.

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

On créé le nouveau commutateur et l'interface commutée virtuel en respectant bien le plan d'adressage.

Voici un exemple de fichier de déclaration `/etc/netplan/enp0s1.yaml` pour le premier routeur ***Spoke***.

```
network:
  version: 2
  renderer: networkd
  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: []
    vlan10:     # VLAN vert
      id: 10
      link: asw-host
      addresses:
        - 10.0.10.1/24
        - fda0:7a62:a::1/64
        - fe80:a::1/64
```

Le même travail doit être réalisé sur le second routeur ***Spoke*** pour que la configuration soit complète.

Q16. Quels sont les tests ICMP à réaliser pour valider les communications entre les routeurs ***Spoke*** ?

Il faut valider les communications réseau entre les adresses des deux interfaces commutées virtuelles des deux routeurs ***Spoke***.

Il faut aussi vérifier que ces mêmes routeurs ***Spoke*** communiquent avec le protocole IPv6 vers l'Internet.

On commence par tester avec succès les échanges entre le second routeur ***Spoke*** et le premier avec IPv4.

```
etu@spoke2:~$ ping -qc2 10.0.10.1
PING 10.0.10.1 (10.0.10.1) 56(84) bytes of data.

--- 10.0.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.799/1.943/2.088/0.144 ms
```

On poursuit dans le même sens avec IPv6.

```
etu@spoke2:~$ ping -qc2 fda0:7a62:a::1
PING fda0:7a62:a::1 (fda0:7a62:a::1) 56 data bytes

--- fda0:7a62:a::1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.146/2.272/2.398/0.126 ms
```

Enfin, on termine avec les communications IPv6 depuis les deux routeurs ***Spoke***.

```
etu@spoke1:~$ ping -qc2 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 = 41.104/47.623/54.142/6.519 ms
```

```
etu@spoke2:~$ ping -qc2 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 1002ms
rtt min/avg/max/mdev = 40.897/47.183/53.470/6.286 ms
```

Au terme de cette partie, les tables de routage de routeurs de la topologie ***Hub and Spoke*** sont complètes et les communications IPv4 et IPv6 fonctionnent entre les routeurs d'extrémités.

7. Installation et gestion des conteneurs

Maintenant que le routage du trafic réseau est complet, on peut passer à l'hébergement des services sur les réseaux des sites distants.

Dans cette partie, on installe le gestionnaire de conteneurs systèmes Incus et on le configure pour que les conteneurs puissent échanger entre les sites.

Avant d'aborder les questions, on s'assure que les outils et les permissions sont en place.

- Le gestionnaire Incus est installé sur chaque routeur ***Spoke***.

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

- L'utilisateur normal `etu` a la capacité à gérer les conteneurs en appartenant aux groupes `incus` et `incus-admin`.

```
for grp in incus incus-admin; do sudo adduser etu $grp; done
```

N'oubliez pas de vous déconnecter/reconnecter pour que l'appartenance aux groupes soit effective.

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

- La configuration par défaut du gestionnaire de conteneurs doit être initialisée.

Voici un exemple de déclaration de configuration.

```
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]:
```

La configuration du raccordement réseau doit être complétée avec les instructions suivantes.

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

```
incus profile device set default eth0 vlan 20
```

> Avertissement
>
> L'exemple ci-dessus doit être adapté au contexte réseau en changeant le numéro de VLAN.

- L'adressage automatique IPv4 et IPv6 doit aussi être installé et configuré sur chaque routeur ***Spoke*** avec l'outil dnsmasq.

Voici un exemple de fichier de configuration à mettre en place. Bien sûr, le nom d'interface sur laquelle les services sont actifs et les adresses IPv4 doivent être changées.

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

```
cat << EOF | sudo tee /etc/dnsmasq.conf
# Specify Container VLAN interface
interface=vlan10

# Enable DHCPv4 on Container VLAN
dhcp-range=10.0.10.100,10.0.10.200,3h

# Enable IPv6 router advertisements
enable-ra

# Enable SLAAC
dhcp-range=::,constructor:vlan10,ra-names,slaac

# Optional: Specify DNS servers
dhcp-option=option:dns-server,172.16.0.2,9.9.9.9
dhcp-option=option6:dns-server,[2001:678:3fc:3::2],[2620:fe::fe]

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

```
sudo systemctl restart dnsmasq.service
```

Q17. Comment lancer 3 nouveaux conteneurs sur chaque routeur ***Spoke*** ?

Rechercher dans les options de la commande incus le moyen de créer un nouveau conteneur système.

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

C'est l'option `launch` qui correspond au besoin. Voici un exemple de création de trois nouveaux conteneurs.

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

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

```
etu@spoke1:~$ incus ls
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4        |                  IPV6                   |   TYPE    | SNAPSHOTS |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| c0   | RUNNING | 10.0.10.140 (eth0) | fda0:7a62:a:0:216:3eff:fe6a:395c (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.0.10.196 (eth0) | fda0:7a62:a:0:216:3eff:fe37:f12a (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
| c2   | RUNNING | 10.0.10.116 (eth0) | fda0:7a62:a:0:216:3eff:fed9:a25b (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------+-----------+-----------+
```

Q18. Comment tester les communications réseau depuis les conteneurs des routeurs ***Spoke*** ?

Reprendre les tests ICMP déjà réalisés depuis les routeurs en les exécutant dans les conteneurs.

Pour traiter cette question, on peut utiliser une boucle qui rassemble les tests IPv4 et IPv6 :

```
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
```

On vérifie qu'aucun paquet n'a été perdu avec le message : `0% packet loss`.

On peut aussi tester les services de la couche application avec la gestion de paquets.

```
for i in {0..2}
do
 echo ">>>>>>>>>>>>>>>> c$i"
 incus exec c$i -- apt -y update
 incus exec c$i -- apt -y full-upgrade
 incus exec c$i -- apt clean
done
```

Q19. Comment combiner l'adressage automatique à l'adressage statique pour chaque conteneur ?

On reprend les traitements réalisés dans le support *Routage inter-VLAN et protocole PPPoE* avec la création d'un script qui regroupe les traitements de gestion de paquets et de création du fichier de déclaration des paramètres des interfaces réseau des conteneurs.

Voici un exemple de script pour un routeur ***Spoke***.

```
#!/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 ' ')

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

# Étape 1 -> installer le paquet netplan.io
  incus exec $c -- apt -y install netplan.io

# Étape 2 -> générer le fichier de configuration réseau YAML
$(cat << EOF > eth0.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: false
      dhcp6: false
      accept-ra: true
      addresses:
        - 10.0.10.$((addr_idx + 10))/24
        - fda0:7a62:a::$(printf "%x" $((addr_idx +10)))/64
      routes:
        - to: default
          via: 10.0.10.1
        - to: "::/0"
          via: fe80:a::1
          on-link: true
      nameservers:
        addresses:
          - 172.16.0.2
          - 2001:678:3fc:3::2
EOF
)

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

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

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

  ((addr_idx++))
done

exit 0
```

Si le script ci-dessus est enregistré dans le fichier `static-addr.sh`, on peut le lancer.

```
bash static-addr.sh
```

Les résultats des traitements sont visibles en affichant la liste des conteneurs actifs.

```
incus ls
```

```
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| NAME |  STATE  |       IPV4        |                  IPV6                   |   TYPE    | SNAPSHOTS |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| c0   | RUNNING | 10.0.10.10 (eth0) | fda0:7a62:a::a (eth0)                   | CONTAINER | 0         |
|      |         |                   | fda0:7a62:a:0:216:3eff:fe6a:395c (eth0) |           |           |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.0.10.11 (eth0) | fda0:7a62:a::b (eth0)                   | CONTAINER | 0         |
|      |         |                   | fda0:7a62:a:0:216:3eff:fe37:f12a (eth0) |           |           |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
| c2   | RUNNING | 10.0.10.12 (eth0) | fda0:7a62:a::c (eth0)                   | CONTAINER | 0         |
|      |         |                   | fda0:7a62:a:0:216:3eff:fed9:a25b (eth0) |           |           |
+------+---------+-------------------+-----------------------------------------+-----------+-----------+
```

On doit suivre le même processus pour le second routeur ***Spoke***.

Q20. Comment ouvrir et tester des services Web dans les réseaux d'hébergement ?

On commence par installer un paquet comme nginx dans chaque conteneur et on effectue les tests d'accès ensuite.

pour simuler la consultation de pages Web depuis les conteneurs, on installe aussi un paquet comme wget.

L'installation des paquets se fait à l'aide d'une boucle.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt update
    incus exec c$i -- apt -y install nginx wget
    incus exec c$i -- apt clean
done
```

Le test de disponibilité des serveurs Web se fait aussi à l'aide d'une boucle.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- ss -at '( sport = :http )'
done
```

L'affichage produit par la commande ss et ses paramètres montre que le service `HTTP` est en écoute dans chaque conteneur.

Q21. Comment valider l'accès à ces services Web à partir du routeur ***Spoke*** situé à l'autre extrémité de la topologie en triangle ?

Il s'agit de faire un test au niveau de la couche application. À la console, les deux outils adaptés sont wget et curl.

Voici un exemple de test pour chaque protocole de la couche réseau avec wget dans le contexte de la maquette.

```
for addr in {10..12}
do
    wget -O /dev/null http://10.0.20.$addr 2>&1 | grep "HTTP "
done
```

```
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
```

```
for addr in {10..12}
do
    wget -O /dev/null http://[fda0:7a62:14::$(printf "%x" $addr)] 2>&1 | grep "HTTP "
done
```

```
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
requête HTTP transmise, en attente de la réponse… 200 OK
```

Q22. Comment matérialiser le chemin suivi par les paquets IPv4 ou IPv6 entre les deux routeurs ***Spoke*** ?

On peut utiliser un outil comme tracepath fourni avec le paquet `iputils-tracepath`. Il suffit d'installer ce paquet dans les conteneurs du second routeur ***Spoke*** et de lancer les tests à destination des conteneurs du premier routeur ***Spoke***.

On installe le paquet dans les conteneurs su second routeur ***Spoke***.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- apt -y install iputils-tracepath
done
```

On lance ensuite les tests, toujours à l'aide d'une boucle.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- tracepath -n 10.0.10.$(($i + 10))
done
```

On obtient les résultats suivants dans lesquels apparait la valeur de l'encapsulation de la quantité de données dans la trame : 1492.

```
>>>>>>>>>>>>>>>> c0
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.20.1                                             0.868ms
 1:  10.0.20.1                                             0.112ms
 2:  10.0.20.1                                             0.078ms pmtu 1492
 2:  10.44.3.1                                             1.260ms
 3:  10.44.1.2                                             1.658ms
 4:  10.0.10.10                                            2.244ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c1
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.20.1                                             1.722ms
 1:  10.0.20.1                                             0.134ms
 2:  10.0.20.1                                             0.056ms pmtu 1492
 2:  10.44.3.1                                             1.193ms
 3:  10.44.1.2                                             1.676ms
 4:  10.0.10.11                                            2.569ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c2
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.20.1                                             2.055ms
 1:  10.0.20.1                                             0.077ms
 2:  10.0.20.1                                             0.074ms pmtu 1492
 2:  10.44.3.1                                             1.094ms
 3:  10.44.1.2                                             1.774ms
 4:  10.0.10.12                                            2.132ms reached
     Resume: pmtu 1492 hops 4 back 4
```

On suit la même démarche avec le protocole IPv6.

```
for i in {0..2}
do
    echo ">>>>>>>>>>>>>>>> c$i"
    incus exec c$i -- tracepath -n fda0:7a62:a::$(printf "%x" $(($i + 10)))
done
```

Les résultats sont identiques aux précédents à l'adressage près.

```
>>>>>>>>>>>>>>>> c0
 1?: [LOCALHOST]                        0.050ms pmtu 1500
 1:  fda0:7a62:14::1                                       1.815ms
 1:  fda0:7a62:14::1                                       0.129ms
 2:  fda0:7a62:14::1                                       3.201ms pmtu 1492
 2:  2001:678:3fc:168:baad:caff:fefe:5                     1.571ms
 3:  fda0:7a62:a:0:9c04:62ff:fe95:f948                     1.742ms
 4:  fda0:7a62:a::a                                        2.239ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c1
 1?: [LOCALHOST]                        0.030ms pmtu 1500
 1:  fda0:7a62:14::1                                       0.572ms
 1:  fda0:7a62:14::1                                       0.116ms
 2:  fda0:7a62:14::1                                       0.071ms pmtu 1492
 2:  2001:678:3fc:168:baad:caff:fefe:5                     1.374ms
 3:  fda0:7a62:a:0:9c04:62ff:fe95:f948                     2.088ms
 4:  fda0:7a62:a::b                                        2.396ms reached
     Resume: pmtu 1492 hops 4 back 4
>>>>>>>>>>>>>>>> c2
 1?: [LOCALHOST]                        0.032ms pmtu 1500
 1:  fda0:7a62:14::1                                       0.666ms
 1:  fda0:7a62:14::1                                       0.091ms
 2:  fda0:7a62:14::1                                       0.093ms pmtu 1492
 2:  2001:678:3fc:168:baad:caff:fefe:5                     1.252ms
 3:  fda0:7a62:a:0:9c04:62ff:fe95:f948                     2.177ms
 4:  fda0:7a62:a::c                                        3.664ms reached
     Resume: pmtu 1492 hops 4 back 4
```

8. Pour conclure...

Ce document présente une configuration détaillée d'une topologie réseau ***Hub & Spoke*** utilisant le protocole PPPoE. Il couvre l'installation et la configuration des routeurs ***Hub*** et ***Spoke***, en établissant des connexions PPP et en configurant le routage IPv4 et IPv6.

Ce support de travaux pratiques aborde également la mise en place de conteneurs sur les routeurs ***Spoke***, démontrant l'hébergement de services web et la communication entre les sites distants. Les étapes sont expliquées de manière approfondie, avec des exemples de commandes et de configurations.

Enfin, le document fournit des méthodes pour tester et valider la connectivité réseau, utilisant des outils comme ping, tracepath et wget. Il offre une approche pratique pour comprendre et mettre en œuvre une topologie réseau complexe avec des protocoles modernes.