DOCUMENTATION METTRE EN PLACE UNE SOLUTION INFO

SAE203

Par Quentin FRANCQ ROUSSEL TALENT

Institut Universitaire de Technologie
Aix*Marseille Université

Aix*Marseille université
Initiative d'excellence

Table des matières

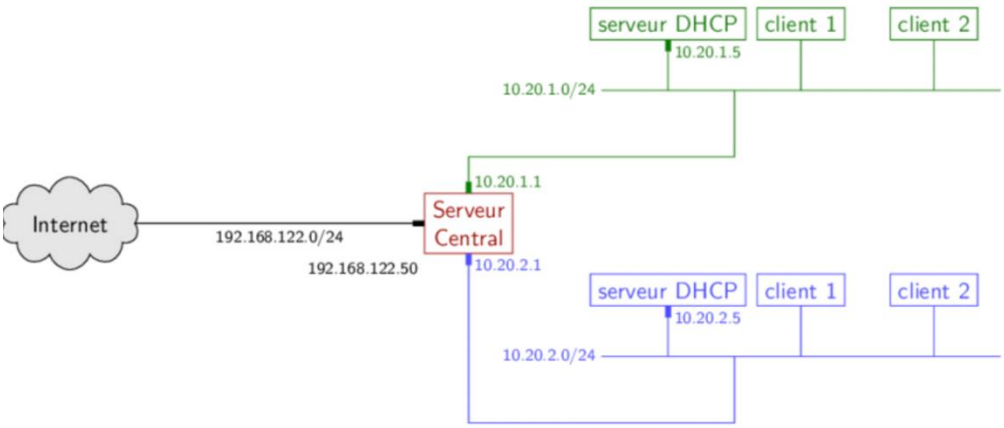
Demande du client et Architecture :

Une entreprise a plusieurs réseaux, sur lesquels les machines sont configurées pour obtenir des adresses IP via DHCP. Cependant, pour garantir la sécurité du réseau, seules les machines dont l'adresse MAC est enregistrée par le service réseau et système peuvent recevoir une adresse via DHCP, et cette adresse est statique

Pour centraliser la gestion des serveurs DHCP sur chaque réseau, l'entreprise souhaite installer un serveur central capable d'accéder aux différents serveurs DHCP. Elle souhaite également gérer de manière centralisée les services (tels que les serveurs WEB et FTP) hébergés sur certaines machines des réseaux locaux.

L'objectif est d'avoir une supervision centralisée des serveurs DHCP et des services pour améliorer la gestion du réseau.

Architecture :



Cette architecture sera composée de 4 machines virtuelles (3 sur le vlan 1 et 3 sur le vlan 2) avec un système Debian 11 fonctionnant avec 512 Mo de RAM. En amont de ces machines, il y aura deux serveurs DHCP (vlan 1 et 2) et en amont de ces serveurs, il y aura le serveur central qui sera relié à internet.

Plan d'adressage IP :

Serveur Centrale		
Réseaux de connexion	Adresse IP	Gateway
192.168.122.0 /24	192.168.122.50	Non
10.20.1.0 /24	10.20.1.1	Oui
10.20.2.0 /24	10.20.2.1	Oui

Serveur DHCP 1		
Réseaux de connexion	Adresse IP	Plage d'adresse
10.20.1.0 /24	10.20.1.5	10.20.1.50 à 10.20.1.250

Serveur DHCP 2		
Réseaux de connexion	Adresse IP	Plage d'adresse
10.20.2.0 /24	10.20.2.5	10.20.2.50 à 10.20.2.250

Les 4 machines virtuelles se verront une adresse IP attribué selon le serveur DHCP 1 ou 2.

Mise en place de l'architecture technique :

L'objectif est de :

- Créer le serveur central ;
- Créer les deux serveurs DHCP ;
- Créer les 4 machines virtuelles ;
- Mettre en place la configuration réseau de chaque machine virtuelle ;
- Tester le fonctionnement de chaque machine ;
- Tester le fonctionnement de l'ensemble.

Création et configuration du serveur central :

Récupération et information générale

Après avoir récupéré sur le partage amu la machine Debian 11 légère, on note l'identifiant et mot de passe de connexion :

- Identifiant : sae203
- Mot de passe : sae203

Modification du nom interne et externe

Nous souhaitons changer le nom interne et externe de notre machine et l'appeler srv-central.
Pour changer le nom interne, il convient de rentrer la commande `sudo hostnamectl set-hostname srv-central`, l'utilisation du sudo est nécessaire.
Pour changer le nom externe, il convient de modifier le fichier /etc/hosts. Nous utiliserons nano pour modifier ce fichier, étant donné que c'est un fichier système il est nécessaire d'utiliser sudo : `sudo nano /etc/hosts`.
Actuellement notre fichier ressemble à ça :

```
127.0.0.1       localhost
127.0.1.1       sae203-base

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Il faut donc remplacer sae203-base par srv-central

Configuration des interfaces réseau

Actuellement notre serveur est relié à internet en DHCP via le réseau 192.168.122.0 /24 via l'interface ens3.
On veut rester sur même réseaux mais avoir l'adresse ip statique 192.168.122.50 /24. De même on veut installer, le Vlan 1 qui servira au serveur DHCP 1 et le Vlan 2 qui servira au serveur DHCP 2, les adresse de Vlan 1 et Vlan seront elle aussi attribués de manière statique.
Pour résumer :

Interface	Type d'interface	Adresse IP et masque	gateway
ens3	Ethernet	192.168.122.50 /24	192.168.122.1
ens3.1	VLAN 802.1Q ID: 1	10.20.1.1 /24	Non

ens3.1	VLAN 802.1Q ID: 1	10.20.2.1 /24	Non

Nous allons d'abord commencer par installer nos deux Vlan. Pour cela il faut charger le module noyau gérant le protocole 802.1Q\ :
Il convient de taper la commande `modprobe 8021q`
Ensuite il faut éditer le fichier /etc/modules en y ajoutant 8021q ce qui permettra au prochain démarrage de charger ce module :
Il convient de taper la commande `sudo nano /etc/modules`, l'utilisation de sudo est nécessaire. Une fois entré dans le fichier, il faut écrire 8021q.
Par la suite nous devons installer le paquet permettant de gérer les VLANs\ :
Il convient de taper la commande `sudo apt install vlan`, l'utilisation de sudo est nécessaire.
Enfin pour que les vlans et que l'adresse 192.168.122.50 /24 soit configuré, il faut éditer le fichier /etc/network/interfaces.
Il convient de taper la commande `sudo nano /etc/network/interfaces`, l'utilisation de sudo est nécessaire.
Une fois entrée dans le fichier on peut apercevoir que l'interface ens3 est configuré en DHCP comme ci-dessous :

```
allow-hotplug ens3
iface ens3 inet dhcp
```

Il faut donc supprimer la ligne

```
iface ens3 inet dhcp
```

Et la remplacer par

```
iface ens3 inet static
    address 192.168.122.50/24
    gateway 192.168.122.1
```

Nous pouvons maintenant ajouter notre Vlan 1 et 2 :

```
auto ens3.1
iface ens3.1 inet static
      address 10.20.1.1
auto ens3.2
iface ens3.2 inet static
      address 10.20.2.1
```

Pour vérifier que tout soit bien pris en compte nous pouvons tester avec la commande `ip addr` ou `ip a`.

Vous devrez obtenir un résultat ressemblant celui-ci :

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
    link/ether 52:54:00:5c:2b:8c brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 192.168.122.50/24 brd 192.168.122.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe5c:2b8c/64 scope link
       valid_lft forever preferred_lft forever
3: ens3.1@ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue s
    link/ether 52:54:00:5c:2b:8c brd ff:ff:ff:ff:ff:ff
    inet 10.20.1.1/24 brd 10.20.1.255 scope global ens3.1
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe5c:2b8c/64 scope link
       valid_lft forever preferred_lft forever
4: ens3.2@ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue s
    link/ether 52:54:00:5c:2b:8c brd ff:ff:ff:ff:ff:ff
    inet 10.20.2.1/24 brd 10.20.2.255 scope global ens3.2
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe5c:2b8c/64 scope link
       valid_lft forever preferred_lft forever
```

Configuration des interfaces réseau est maintenant terminée.

Pour que les machines des réseaux 10.20.1.0/24 et 10.20.2.0/24 aient accès à l'extérieur, il est nécessaire d'activer le routage nat, qui permettra aux réseaux 10.20.1.0/24 et 10.20.2.0/24 de communiqué avec l'extérieur grâce au réseau 192.168.122.0.
Pour cela nous devrons utiliser la commande `sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o ens3`.
Pour mieux comprendre cette commande nous allons l'expliquer pas à pas :

- -t nat permet de spécifier la table à laquelle on s'intéresse dans notre cas la table nat
- Ici la chaine qui nous intéresse est POSTROUTING et -A permet de spécifier la chaine
- Ici la cible qui nous intéresse est MASQUERADE et -j permet de spécifier la cible
- Enfin -o permet de spécifier l'interface de sortie dans notre cas ens3.

Une fois cette commande entrée il est nécessaire d'entrée la commande `iptables -t nat -n -L – v`
Où :

- -n évite les recherches DNS inverse (on s'en fiche)
- -L permet de préciser la chaîne, mais comme ici on ne dit rien toutes les chaînes sont listées
- -v permet d'afficher tous les détails de la règle.

On devrait obtenir quelque chose comme ça :

```
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

   22  1558 MASQUERADE  all  --  *      ens3    0.0.0.0/0            0.0.0.0/0
```

Si tout est bon, il faut sauver la manipulation précédant au moyen de la commande `sudo iptables-save`, l'utilisation de sudo est nécessaire.
Normalement il devrait s'afficher quelque chose comme ça :

```
$ sudo iptables-save
# Generated by iptables-save v1.8.7 on Tue Mar 14 13:25:23 2023
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o ens3 -j MASQUERADE
COMMIT
# Completed on Tue Mar 14 13:25:23 2023
```

Pour la suite nous allons créer un fichier nommer iptables.rules.v4 placer dans le répertoire /etc :
Il convient d'entrer la commande `sudo touch /etc/iptables.rules.v4`, l'utilisation de sudo est nécessaire.
Dans ce fichier on y place le résultat précédent (suite à la commande sudo iptables-save).

Maintenant il faut le rendre automatique pour chaque démarrage de cette machine.
Pour cela nous allons créer un fichier nommé iptables dans le répertoire /etc/network/if-pre-up.d.
Il convient d'entrer la commande `sudo touch touch /etc/network/if-pre-up.d/iptables`, l'utilisation de sudo est nécessaire.

Avant de l'éditer nous allons lui attribuer les bons droits :
Il convient d'entrer la `commande sudo chmod +x /etc/network/if-pre-up.d/iptables`.

Une fois les bons droits attribués nous pouvons éditer ce fichier au moyen de la commande :
`sudo nano /etc/network/if-pre-up.d/iptables`
Est écris :

```
#!/bin/sh
/sbin/iptables-restore < /etc/iptables.rules.ipv4
```

Le serveur central est maintenant opérationnel

Création et configuration des deux serveurs DHCP :

On notera que les manipulations suivantes sont les mêmes pour serveur DHCP 1 et serveur DHCP2.

Récupération et information générale

Après avoir récupéré sur le partage amu la machine Debian 11 légère, on note l'identifiant et mot de passe de connexion :

- Identifiant : sae203
- Mot de passe : sae203

Modification du nom interne et externe

Nous souhaitons changer le nom interne et externe de nos machines et les appeler srv-dhcp-1 et srv-dhcp-2.
Pour changer le nom interne, il convient de rentrer la commande `sudo hostnamectl set-hostname srv-dhcp-<num>`, l'utilisation du sudo est nécessaire.
Pour changer le nom externe, il convient de modifier le fichier /etc/hosts. Nous utiliserons nano pour modifier ce fichier, étant donné que c'est un fichier système il est nécessaire d'utiliser sudo : `sudo nano /etc/hosts`.
Actuellement notre fichier ressemble à ça :

```
127.0.0.1       localhost
127.0.1.1       sae203-base

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Il faut donc remplacer `sae203-base` par le nom de nos machines srv-dhcp-1 et srv-dhcp-2.

Configuration des interfaces réseau

Actuellement nos deux serveurs sont reliés à internet en DHCP via le réseau 192.168.122.0 /24 via l'interface ens3.
Nous voulons toujours utiliser cette interface mais aussi créer les interfaces Vlan1 pour srv-dhcp-1 et Vlan 2 pour srv-dhcp-2.
Pour résumer :

srv-dhcp-1			
Interface	Type d'interface	Adresse IP et masque	Gateway
ens3	Ethernet	dhcp	
ens3.1	VLAN 802.1Q ID: 1	10.20.1.5 /24	10.20.1.1

srv-dhcp-2			
Interface	Type d'interface	Adresse IP et masque	Gateway
ens3	Ethernet	dhcp	
ens3.1	VLAN 802.1Q ID: 1	10.20.2.5 /24	10.20.2.1

Pour rendre cela effectif des modifications dans le fichier /etc/network/interfaces sont nécessaire.

Il convient de taper la commande `sudo nano /etc/network/interfaces`, l'utilisation de sudo est nécessaire.
Une fois entré dans le fichier on devrait voir quelque chose comme ça :

```
allow-hotplug ens3
iface ens3 inet dhcp
```

Nous n'allons pas toucher ces deux lignes mais on ajoute en-dessous de manière à obtenir ce résultat :
Pour srv-dhcp-1

```
allow-hotplug ens3
iface ens3 inet dhcp
auto ens3.1
iface ens3.1 inet static
      address 10.20.1.5 /24
      gateway 10.20.1.1
```

Pour srv-dhcp-2

```
allow-hotplug ens3
iface ens3 inet dhcp

auto ens3.2
iface ens3.2 inet static
      address 10.20.2.5 /24
      gateway 10.20.2.1
```

Tout est opérationnel nous pouvons tester et voir si les nouveaux paramètres on bien été pris en compte avec `ip addr` ou `ip a`.

Fixer la gateway

Sur chaque serveur dhcp, nous avons configuré l'interface réseau ens3 en utilisant le protocole DHCP pour que celle-ci soit automatiquement attribuée une adresse sur le réseau 192.168.122.0/24, étant donné que c'est le principal trunk physique.
Cependant, lors de l'exécution de la commande `ip route`, nous avons remarqué que la route par défaut était définie sur 192.168.122.1, ce qui n'est pas souhaitable.
En réalité, l'interface physique ens3 est seulement censée servir de support aux VLANs 1 et 2. Cependant, puisque ens3 est nécessairement activée avant les interfaces VLAN (ens3.1 ou ens3.2 selon la machine), elle impose sa route par défaut.
Pour résoudre ce problème, nous devons ajouter dans le fichier de configuration du client DHCP (/etc/dhcp/dhclient.conf) une section dédiée à l'interface ens3, qui prend en paramètre :

- Ne demande que l'adresse IP
- Le masque et la MTU au serveur DHCP

- Sans demander la route par défaut ni l'adresse des serveurs DNS.

Pour cela il convient de rentrer la commande `sudo nano /etc/dhcp/dhclient.conf`, l'utilisation de sudo est nécessaire.
Une fois entrée dans le fichier.
Nous allons ajouter :

```
interface "ens3" {
	request subnet-mask, broadcast-address, interface-mtu;
}
```

Dès que la modification de ce fichier est terminée, il faut redémarrer ce service par le biais de la commande :

```
systemctl restart networking.service
```

La simple utilisation de `ip route` nous permet de voir que les paramètres ont bien été pris en compte :

```
default via 10.20.1.1 dev ens3.1 onlink
10.20.1.0/24 dev ens3.1 proto kernel scope link src 10.20.1.5
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.101
```

Configuration de la distribution d'adresse via dhcp avec dnsmasq

Avant tout il faut installer dnsmasq sur chaque serveur :
Il convient de taper la commande `sudo apt install dnsmasq`, l'utilisation de sudo est nécessaire.

Nous allons maintenant configurer dnsmasq de manière qu'il distribue une adresse parmi une plage d'adresses définie aux machines clients relié à chaque serveur dhcp.
Pour résumer :

Hôte	Plage d'adresse	Durée des baux
srv-dhcp-1	10.20.1.50 à 10.20.1.150	12h
srv-dhcp-2	10.20.2.50 à 10.20.2.150	12h

Pour chaque serveur dhcp, il faut créer un fichier dns.conf situé dans le répertoire /etc/dnsmasq.d :
Il convient de taper la commande `sudo touch /etc/dnsmasq.d/range.conf`, l'utilisation de sudo est nécessaire.

Une fois que ce fichier est créé nous pouvons l'éditer :
Il convient de taper la commande `sudo nano /etc/dnsmasq.d/range.conf`, l'utilisation de sudo est nécessaire.

Maintenant que nous sommes entrées dans ce fichier il faut écrire :

- Pour srv-dhcp-1

```
dhcp-range = 10.20.1.50, 10.20.1.150, 255.255.255.0, 12h
```

- Pour srv-dhcp-2

```
dhcp-range = 10.20.2.50, 10.20.2.150, 255.255.255.0, 12h
```

Optionnel :
Il est possible d'attribuer des adresses ip static au client, pour cela il faut créer un fichier /etc/dnsmasq.d/host.conf et l'éditer de la manière suivante, pour cela on rentre la commande `sudo nano /etc/dnsmasq.d/host.conf`, où l'utilisation de sudo est nécessaire.

```
dhcp-host = <mac>,<nom_de_la_machine>,<ip_choisie>
```

Nos deux serveurs dhcp sont opérationnels.
Pour tester la connectivité réseau des serveurs srv-dhcp-1 et srv-dhcp-2, il est nécessaire d'effectuer un ping vers une adresse extérieure tout en résolvant un nom d'hôte pour tester également le fonctionnement du serveur DNS.
Par exemple, en exécutant la commande `ping www.univ-amu.fr`, le ping doit réussir. De plus, le résultat d'une commande `traceroute <ip>` sur la même machine doit afficher les deux premiers hôtes qui ont relayé la requête : le premier étant le gateway du serveur DHCP avec l'adresse IP 10.20.1.1, qui correspond à srv-central, suivi de l'adresse IP 192.168.122.1 qui est le gateway de srv-central.

Ajouter la ligne à décommenter dans /etc/sysctl.conf
Net.ipv4.ip_forward=1

Création des 4 machines clientes

On notera que les manipulations suivantes sont les mêmes pour serveur client-1-1, client-1-2, client-2-1, et client-2-2.

Récupération et information générale

Après avoir récupéré sur le partage amu la machine Debian 11 légère, on note l'identifiant et mot de passe de connexion :

- Identifiant : sae203
- Mot de passe : sae203

Modification du nom interne et externe

Nous souhaitons changer le nom interne et externe de nos machines et les appeler client-1-1, client-1-2, client-2-1, et client-2-2.
Pour changer le nom interne, il convient de rentrer la commande `sudo hostnamectl set-hostname client-<num>-<num>`, l'utilisation du sudo est nécessaire.
Pour changer le nom externe, il convient de modifier le fichier /etc/hosts. Nous utiliserons nano pour modifier ce fichier, étant donné que c'est un fichier système il est nécessaire d'utiliser sudo : `sudo nano /etc/hosts`.

Actuellement notre fichier ressemble à ça :

```
127.0.0.1       localhost
127.0.1.1       sae203-base

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Il faut donc remplacer `sae203-base` par le nom de nos machines.

Configuration des interfaces réseau

Actuellement nos deux serveurs sont reliés à internet en DHCP via le réseau 192.168.122.0 /24 via l'interface ens3.
Nous voulons toujours utiliser cette interface mais aussi avoir une adresse en donné à nos machines par nos deux serveurs DHCP.

Client-1-1			
Interface	Type d'interface	Adresse IP et masque	Gateway
ens3	Ethernet	dhcp	
ens3.1	VLAN 802.1Q ID: 1	dhcp	10.20.1.1

Client-1-2			
Interface	Type d'interface	Adresse IP et masque	Gateway
ens3	Ethernet	dhcp	
ens3.1	VLAN 802.1Q ID: 1	dhcp	10.20.1.1

Client-2-1			
Interface	Type d'interface	Adresse IP et masque	Gateway
ens3	Ethernet	dhcp	
ens3.2	VLAN 802.1Q ID: 1	dhcp	10.20.2.1

Client-2-2			
Interface	Type d'interface	Adresse IP et masque	Gateway
ens3	Ethernet	dhcp	
ens3.2	VLAN 802.1Q ID: 1	dhcp	10.20.2.1

Pour rendre cela effectif des modifications dans le fichier /etc/network/interfaces sont nécessaire.
Il convient de taper la commande `sudo nano /etc/network/interfaces`, l'utilisation de sudo est nécessaire.
Une fois entré dans le fichier on devrait voir quelque chose comme ça :

```
allow-hotplug ens3
iface ens3 inet dhcp
```

Nous n'allons pas toucher ces deux lignes mais on ajoute en-dessous de manière à obtenir ce résultat :
Pour client-1-1 et client-1-2 qui appartient à srv-dhcp-1

```
allow-hotplug ens3
iface ens3 inet dhcp
auto ens3.1
iface ens3 inet dhcp
```

Pour client-2-1 et client-2-2 qui appartiennent à srv-dhcp-2

```
allow-hotplug ens3
iface ens3 inet dhcp

auto ens3.2
iface ens3 inet dhcp
```

Tout est opérationnel nous pouvons tester et voir si les nouveaux paramètres on bien été pris en compte avec `ip addr` ou `ip a`.

Fixer la gateway

A appliquer à chaque client :
Sur chaque client, nous avons configuré l'interface réseau ens3 en utilisant le protocole DHCP pour que celle-ci soit automatiquement attribuée une adresse sur le réseau 192.168.122.0/24, étant donné que c'est le principal trunk physique.
Cependant, lors de l'exécution de la commande `ip route`, nous avons remarqué que la route par défaut était définie sur 192.168.122.1, ce qui n'est pas souhaitable.
En réalité, l'interface physique ens3 est seulement censée servir de support aux VLANs 1 et 2. Cependant, puisque ens3 est nécessairement activée avant les interfaces VLAN (ens3.1 ou ens3.2 selon la machine), elle impose sa route par défaut.
Pour résoudre ce problème, nous devons ajouter dans le fichier de configuration du client DHCP (/etc/dhcp/dhclient.conf) une section dédiée à l'interface ens3, qui prend en paramètre :

- Ne demande que l'adresse IP
- Le masque et la MTU au serveur DHCP
- Sans demander la route par défaut ni l'adresse des serveurs DNS.

Pour cela il convient de rentrer la commande `sudo nano /etc/dhcp/dhclient.conf`, l'utilisation de sudo est nécessaire.
Un fois entrée dans le fichier.
Nous allons ajouter :

```
interface "ens3" {
      request subnet-mask, broadcast-address, interface-mtu;
}
```

Dès que la modification de ce fichier est terminée, il faut redémarrer ce service par le biais de la commande :

```
systemctl restart networking.service
```

La simple utilisation de `ip route` nous permet de voir que les paramètres ont bien été pris en compte.

Nos clients sont opérationnels.
Pour tester la connectivité réseau des clients, il est nécessaire d'effectuer un ping vers une adresse extérieure tout en résolvant un nom d'hôte pour tester également le fonctionnement du serveur DNS.
Par exemple, en exécutant la commande `ping www.univ-amu.fr`, le ping doit réussir. De plus, le résultat d'une commande `traceroute` sur la même machine doit afficher les trois premiers hôtes qui ont relayé la requête : le premier étant le gateway du serveur DHCP avec l'adresse IP 10.20.1.5, qui correspond à srv-dhcp-1, suivi de l'adresse IP 10.20.1.1, qui correspond à srv-central, suivi de l'adresse IP 192.168.122.1 qui est le gateway de srv-central.

Mise en place des commandes de gestion DHCP

Au cours de cette partie nous allons développer 4 commandes en python qui seront sur notre serveur central :

- Commande add-dhcp-client
  Permets d'ajouter un client DHCP à partir des informations suivantes :
  - Adresse MAC de la machine ;
  - Adresse IP attribuée.
  - Nom ou IP du serveur DHCP
- Commande remove-dhcp-client
  Supprimer un client DHCP à partir de son adresse MAC et du nom ou de l'IP du serveur DHCP.
- Commande check-dhcp
  Vérification de l'intégrité de la configuration DHCP, et optionnellement correction de celle-ci.
- Commande list-dhcp
  Affichage formatté de la configuration DHCP de tous les réseaux ou d'un réseau particulier.

Pour arriver à un tel résultat nous allons commencer par créer 3 sous-fonctions qui seront utilisé par ces 4 fonctions :

- La fonction validation
  Cette fonction permet de vérifier si une adresse mac et ou ip est correcte
- La fonction config
  Cette fonction permet de charger un fichier de configuration s'il existe et ou d'en créer un. Également elle permet de sélectionner un de nos serveurs dhcp à partir d'une adresse ip du réseau des serveurs dhcp.
- La fonction dhcp
  Cette fonction regroupe toutes les fonctions qui seront utiles à nos 4 commandes

Mise en place

Mise en place de clés ssh

Comme introduit nos 4 commandes seront sur notre serveur central et feront appel à des fichiers de configurations qui sont sur nos deux serveurs dhcp. Pour que le serveur central est accès au deux serveurs dhcp, nous allons utiliser ssh via des clés. Pour cela il faut commencer par générer une paire de clés.
Depuis le serveur central, il convient de renter la commande : ssh-keygen, il est vivement recommandé de mettre une pass-phrase.
Une paire de clés publique et privée sont générées dans le répertoire .ssh.
Nous allons à présent envoyer la clé publique vers nos serveurs dhcp. Pour cela nous allons utiliser scp qui permet d'envoyer des fichiers entre deux machines. Scp s'utilise de la manière suivante : `scp <source_du_fichier> <nom_hote>@<adresse_de_la_machine_de_destination> :<destination>`. Pour le serveur dhcp 1, Il convient de rentrer la commande : scp .ssh/id_rsa.pub

sae203@10.20.1.5:.ssh. Il faut répater la manipulation pour le serveur dhcp 2. Une fois les clés arrivées sur les deux serveurs dhcp il faut les renommé à l'aide de la commande : mv .ssh/id_rsa.pub .ssh/autorizedkeys.

Le serveur central a maintenant accès aux deux serveurs dhcp sans mot de passe.

Mise en place de l'environnement python

Nos 4 commandes seront développées en python avec des bibliothèques qu'il faut installer. C'est ce que nous allons faire dans cette partie.

Il faut tout d'abord commencer par installer python sur le serveur central. Le paquet s'appelle python3-pip, il faut alors rentrer la commande `sudo apt install pyhton3-pip`, l'utilisation de sudo est nécessaire.

A présent nous allons installer la bibliothèque fabric. Pour cela il faut entrer la commande `pip3 install fabric`, ici l'utilisation de sudo n'est pas nécessaire étant donné que les commandes seront exécutées par l'utilisateur sae203.

Mise en place de l'utilisation du sudo sans mot de passe

Lorsque nous utiliserons nos 4 commandes, nous serons amenées à avoir besoin d'utiliser les droits root sur nos deux serveurs dhcp, c'est pourquoi nous allons rendre l'utilisation de sudo disponible sans avoir besoin d'utiliser de mot de passe.

Attention la modification du fichier /etc/sudoers est risquée c'est pourquoi nous allons le modifier à l'aide de visudo. En effet une mauvaise manipulation et l'utilisation de sudo n'est plus disponibles.

Depuis nos deux serveurs dhcp :
Entrer la commande sudo visudo. Une fois entrée dans le fichier se rendre à la ligne :

```
%sudo ALL=(ALL:ALL) ALL
```

Et la remplacer par :

```
%sudo ALL=(ALL:ALL) NOPASSWD: ALL
```

La mise en place est maintenant terminée

Développement des 3 sous-fonctions

Afin d'avoir d'un environnement graphique, il est recommandé de développer ces sous-fonctions depuis une machine graphique ayant un logiciel python comma Spyder.

Fonction validation

Il faut commencer par créer un fichier nommé validation.py
Cette fonction à deux définitions :

Définition valid_ip

Prends en paramètre une chaine de caractère ip_address et renvoi True si l'adresse ip est valide et False sinon. Cette définition utilise la bibliothèque ipaddress.

Définition valid_mac

Prend en paramètre une chaine de caractère mac_address et renvoi True si l'adresse mac et valide et False sinon. Cette définition utilise la bibliothèque re.

Fonction config

Il faut commencer par créer un fichier nommé config.py
Cette fonction à deux définitions :

Définition load_config

Prends en paramètre une chaine de caractère représentant le chemin d'un fichier de configuration et un boolean True ou False. Cette définition permet de renvoyer s'il existe le contenu d'un fichier de configuration, s'il n'existe pas et que le boolean et à True un fichier de configuration minimale est créé dans le cas contraire si le boolean est à False la fonction renvoi un message d'erreur. Également s'il n'est pas possible de charger le fichier de configuration un message d'erreur est affiché. Cette définition utilise les bibliothèques os et yaml.

Définition get_dhcp_server

Prends en paramètre une chaine de caractère ip et un dictionnaire représentant un fichier de configuration. Cette définition renvoie un le réseau de l'adresse ip s'il est trouvé dans le fichier de configuration, dans le cas contraire la définition renvoie None. Cette définition utilise la bibliothèque ipaddress.

Fonction dhcp

Il faut commencer par créer un fichier nommé dhcp.py
Cette fonction à cinq définitions :
Trois fonctions utilisent les bibliothèques fabric et getpass, il faut alors les importer au début du fichier.

Définition ip_other_mac_exists

Cette définition prend en paramètre une connexion ssh , une chaine de caractère représentant une adresse ip, une chaine de caractère représentant une adresse mac, et un dictionnaire représentant un fichier de configuration. Cette définition cherche à l'aide de la connexion ssh le fichier /etc/dnsmaq.d/host.conf compare les adresses mac et ip pris en paramètre et du fichier. Si l'adresse ip existe dans le fichier mais avec une adresse mac différente la définition renvoie True dans une autre situation, elle renvoie False.

Définition mac_exists

Cette définition prend en paramètre une connexion ssh , une chaine de caractère représentant une adresse mac, et un dictionnaire représentant un fichier de configuration. Cette définition cherche à l'aide de la connexion ssh le fichier /etc/dnsmaq.d/host.conf compare l'adresse mac prise en paramètre existe dans le fichier. Si telle est le cas la définition renvoie True sinon False.

Définition dhcp_add

Cette définition prend en paramètre une chaine de caractère qui représente une adresse ip, une deuxième chaine de caractère qui représente une adresse mac, une troisième qui représente l'adresse ip du serveur dhcp concernée et un dictionnaire représentant un fichier de configuration. Cette définition ajoute une adresse ip static selon certaines conditions d'un fichier de configuration.

Définition dhcp_remove

Cette définition prend en paramètre une chaine de caractère qui représente une adresse mac, une troisième qui représente l'adresse ip du serveur dhcp concernée et un dictionnaire représentant un fichier de configuration. Cette définition supprime une adresse ip static selon certaines conditions d'un fichier de configuration.

Définition dhcp_list

Cette définition prend en paramètre une chaine de caractères serveur représentant l'ip d'un serveur dhcp et et un dictionnaire représentant un fichier de configuration. Cette définition ouvre une connexion ssh pour afficher le contenu du fichier /etc/dnsmasq.d/host.conf.

Développement des 4 commandes

Pour utiliser ces quatre commandes nous devront importer nos 4 sous-fonctions ainsi que la bibliothèque sys qui sert à récupérer les arguments saisis en ligne de commande.
Également dans toutes nos fonctions nous définirons une définition qui se lancera automatiquement.

Aussi il peut être judicieux de créer un fichier de configuration même si nos fonctions le créeront, il préférable de créer le fichier suivant :
Nous l'appellerons configuration_min, pour cela entrez la commande `nano configuration_min` et y ajouter :

```
dhcp_hosts_cfg: /etc/dnsmasq.d/host.conf
user: sae203
dhcp-servers:
 10.20.1.5: 10.20.1.0/24
 10.20.2.5: 10.20.2.0/24
```

Fonction add-dhcp-client

Pour cela il faut commencer par créer un fichier add-dhcp-client.py.
Cette fonction sert à ajouter un hôte dhcp ou à modifier l'adresse ip d'un hôte. L'hôte est ajouté si l'adresse mac n'est pas déjà utilisée et que l'adresse ip n'est pas déjà utilisée. Si l'adresse mac et déjà utilisée on changera seulement l'adresse ip de l'hôte si l'ip choisie n'est pas déjà attribué à un autre hôte.

Utilisation et appel de cette fonction :

```
add-dhcp-client.py <mac_address> <ip_address>
```

Fonction remove-dhcp-client

Pour cela il faut commencer par créer un fichier remove-dhcp-client.py.
Cette fonction sert à supprimer un hôte dhcp à partir de son adresse mac.

Utilisation et appel de cette fonction :

```
remove-dhcp-client.py <mac_address>
```

Fonction list-dhcp

Pour cela il faut commencer par créer un fichier list-dhcp.py.
Cette fonction sert à afficher la liste des hôtes enregistrée dans un serveur dhcp. Elle prend un paramètre optionnel qui est une adresse qui sert à déterminer sur quel serveur affiché les hôtes. Si aucun argument n'est donné on affiche tous les hôtes de tous les serveurs dhcp enregistrée dans le fichier de configuration.

Utilisation et appel de cette fonction :
Appel avec la liste de tous les serveurs dhcp :

```
list-dhcp.py
```

Appel avec la liste d'un serveur dhcp spécifique :

```
list-dhcp.py <ip_adress>
```

Fonction check-dhcp:

Pour cela il faut commencer par créer un fichier check-dhcp.py.
Cette fonction sert à vérifier la bonne configuration du fichier dhcp. Pour cela cette fonction contrôle s'il y a deux fois la présence de la même adresse mac ou ip. Dans le cas où une ou plusieurs erreurs seraient avéré cette fonction retourne les lignes où les erreurs sont présentes. Cette fonction prend un paramètre optionnel qui est une adresse ip. Dans le cas où l'argument est pris en paramètre la fonction en contrôle que le serveur dhcp sur le même réseau que l'adresse ip, si aucun argument n'est donné la fonction vérifie la configuration de tous les serveurs dhcp enregistrée dans le fichier de configuration.

Utilisation et appel de cette fonction :
Appel avec la liste de tous les serveurs dhcp :

```
check-dhcp.py
```

Appel avec la liste d'un serveur dhcp spécifique :

```
check-dhcp.py <ip_adress>
```

Rendre les commandes utilisables / exécutables

Actuellement nos fonctions sont placées dans /home/sae203. C'est pourquoi nous allons rendre ce répertoire exécutable. Pour cela il convient d'entrer la commande : `export PATH :"/home/sae203 :$PATH"`. On peut vérifier que /home/sae203 a bien été ajouté à PATH à l'aide de la commande : echo $PATH. Maintenant il faut rendre les permissions d'exécution à nos fichiers il faut alors entrer la commande : `chmod +x /home/sae203/<nom_du_fichier>.py`

Nos fonctions sont maintenant exécutables et notre réseau et enfin près à l'utilisation