

Dans ce sixième TP, nous allons voir comment mettre en place différents services réseau (serveur DHCP, serveur DNS, serveur web...) sur notre serveur.

### Préambule : le réseau sous VirtualBox

VirtualBox propose plusieurs modes permettant aux machines virtuelles de communiquer entre elles ou avec l'extérieur, résumés ci-dessous :

Aucune connexion	La VM voit qu'une carte réseau est présente, mais c'est comme si le câble était débranché
Réseau interne	Réseau limité aux VM : elles peuvent communiquer entre elles mais sont isolées de l'extérieur et ne peuvent communiquer avec l'hôte
Réseau privé hôte	Les VM communiquent entre elles et avec l'hôte, via une carte réseau virtuelle
NAT ( <b>mode par défaut</b> )	La VM accède à Internet via l'hôte, et n'est pas visible sur Internet
Réseau NAT	Id. NAT, mais les VM peuvent aussi communiquer entre elles
Accès par pont (bridge)	VirtualBox contourne la pile réseau de l'hôte et accède directement à la carte réseau choisie. Vu du réseau, une nouvelle machine apparaît
Pilotes génériques	Rarement utilisé ; permet d'offrir une architecture ouverte aux plug-ins

Le tableau suivant résume les différentes communications possibles selon le mode choisi :

Mode	VM > Hôte	VM < Hôte	VM1 <> VM2	VM > Extérieur	VM < Extérieur
Aucune connexion	-	-	-	-	-
Réseau interne	-	-	✓	-	-
Réseau privé hôte	✓	✓	✓	-	-
NAT	✓	Redirection	-	✓	Redirection
Réseau NAT	✓	Redirection	✓	✓	Redirection
Pont	✓	✓	✓	✓	✓

### Exercice 1. Adressage IP (rappels)

Vous administrez le réseau interne 172.16.0.0/23 d'une entreprise, et devez gérer un parc de 254 machines réparties en 7 sous-réseaux. La répartition des machines est la suivante :




- Sous-réseau 1 : 38 machines
- Sous-réseau 2 : 33 machines
- Sous-réseau 3 : 52 machines
- Sous-réseau 4 : 35 machines
- Sous-réseau 5 : 34 machines
- Sous-réseau 6 : 37 machines
- Sous-réseau 7 : 25 machines

Donnez, pour chaque sous-réseau, l'adresse de sous-réseau, l'adresse de broadcast (multidiffusion) ainsi que les adresses de la première et dernière machine **configurées** (précisez si vous utilisez du VLSM ou pas).

## Exercice 2. Préparation de l'environnement

Dans ce TP nous allons mettre en place un réseau rudimentaire constitué de seulement deux machines : un **serveur** et un **client** :

- le *serveur* a une connexion Internet, notamment pour télécharger les paquets nécessaires à l'installation des serveurs, et sert de passerelle au *client* ;
- les deux machines appartiennent à un réseau local ;
- le *client* a accès à Internet uniquement via le serveur ; il dispose d'une interface réseau qui recevra son adresse IP du serveur DHCP. Dans un premier temps, cette interface sera activée mais **débranchée**.

1. VM **éteintes**, utilisez les outils de configuration de VirtualBox pour mettre en place l'environnement décrit ci-dessus.
2. Démarrez le serveur et vérifiez que les interfaces réseau sont bien présentes. A quoi correspond l'interface appelée **lo** ?
3.  Dans les versions récentes, Ubuntu installe d'office le paquet **cloud-init** lors de la configuration du système. Ce paquet permet la configuration et le déploiement de machines dans le cloud via un script au démarrage. Nous ne nous en servons pas et sa présence interfère avec certains services (en particulier le changement de nom d'hôte) ; par ailleurs, vos machines démarreront plus rapidement. **Désinstallez complètement ce paquet (il faudra penser à le faire également sur le client ensuite.)**
4. Les deux machines **serveur** et **client** se trouveront sur le domaine **tpadmin.local**. A l'aide de la commande **hostnamectl** renommez le serveur (**le changement doit persister après redémarrage**, donc cherchez les bonnes options dans le manuel !). On peut afficher le nom et le domaine d'une machine avec les commandes **hostname** et/ou **dnsdomainname** ou en affichant le contenu du fichier **/etc/hostname**.
  -  Il se peut que l'ancien nom persiste dans le fichier **/etc/hosts**, associé à l'adresse IP **127.0.1.1** ; si c'est le cas, modifiez ce fichier à la main.
  -  ISC DHCP Server utilise le temps UTC, qui diffère de l'heure locale française. C'est pourquoi vous pourrez observer une différence entre la machine virtuelle et la machine hôte.

### Exercice 3. Installation du serveur DHCP

Un serveur DHCP permet aux ordinateurs clients d'obtenir automatiquement une configuration réseau (adresse IP, serveur DNS, passerelle par défaut...), pour une durée déterminée. Ainsi, dans notre cas, l'interface réseau de **client** doit être configurée automatiquement par **serveur**. Le réseau local **tpadmin.local** a pour adresse **192.168.100.0/24** (on aurait pu choisir une autre adresse; attention, 192.168.1.0/24 est souvent réservée, par exemple par votre FAI).

1. Sur le serveur, installez le paquet **isc-dhcp-server**. La commande **systemctl status isc-dhcp-server** devrait vous indiquer que le serveur n'a pas réussi à démarrer, ce qui est normal puisqu'il n'est pas encore configuré (en particulier, il n'a pas encore d'adresses IP à distribuer).
2. Un serveur DHCP a besoin d'une IP statique. Attribuez de manière **permanente** l'adresse IP 192.168.100.1 à **l'interface réseau du réseau interne**. Vérifiez que la configuration est correcte.
3. La configuration du serveur DHCP se fait via le fichier **/etc/dhcp/dhcpd.conf**. Faites une sauvegarde du fichier existant sous le nom **dhcpd.conf.bak** puis éditez le fichier **dhcpd.conf** avec les informations suivantes :

```
default-lease-time 120;
max-lease-time 600;
authoritative;                                #DHCP officiel pour notre réseau
option broadcast-address 192.168.100.255;      #informe les clients de l'adresse de broadcast
option domain-name "tpadmin.local";           #tous les hôtes qui se connectent au
                                              #réseau auront ce nom de domaine

subnet 192.168.100.0 netmask 255.255.255.0 { #configuration du sous-réseau 192.168.100.0
    range 192.168.100.100 192.168.100.240;    #pool d'adresses IP attribuables
    option routers 192.168.100.1;              #le serveur sert de passerelle par défaut
    option domain-name-servers 192.168.100.1; #le serveur sert aussi de serveur DNS
}
```

A quoi correspondent les deux premières lignes ?

💡 Les valeurs indiquées sur ces deux lignes sont faibles, afin que l'on puisse voir constituer quelques logs durant ce TP. Dans un environnement de production, elles sont beaucoup plus élevées !

4. Editez le fichier **/etc/default/isc-dhcp-server** afin de spécifier l'interface sur laquelle le serveur doit écouter.
5. Validez votre fichier de configuration avec la commande **dhcpd -t** puis redémarrez le serveur DHCP (avec la commande **systemctl restart isc-dhcp-server**) et vérifiez qu'il est actif.
6. Notre serveur DHCP est configuré ! Passons désormais au client. Si vous avez suivi le sujet du TP 1, le client a été créé en *clonant* la machine virtuelle du serveur. Par conséquent, son nom d'hôte est toujours **serveur**. Vérifiez que la carte réseau du client est **débranchée**, puis démarrez le client (il est possible qu'il mette un certain temps à démarrer : ceci est dû à l'absence de connexion Internet). Comme pour le serveur, désinstallez ensuite **cloud-init**, puis modifiez le nom de la machine (elle doit s'appeler **client.tpadmin.local**).  
💡 Pour empêcher la latence au démarrage si une machine n'a pas de connexion Internet, on peut ajouter la ligne **optional: true** dans son fichier de configuration **/etc/netplan/00-installer-config.yaml**  
💡 Il est possible que les commandes **sudo** prennent désormais un certain temps à s'exécuter sur le client. C'est parce que **sudo** utilise le fichier **/etc/hosts** qui contient l'ancien nom de la machine. Il faut modifier ce fichier à la main si vous rencontrez ce problème.
7. La commande **tail -f /var/log/syslog** affiche de manière continue les dernières lignes du fichier de log du système (dès qu'une nouvelle ligne est écrite à la fin du fichier, elle est affichée à l'écran). Lancez cette commande sur le **serveur**, puis connectez la carte réseau du **client** et observez les logs sur le serveur. Expliquez à quoi correspondent les messages **DHCPDISCOVER**, **DHCPPOFFER**, **DHCPREQUEST**, **DHCPACK**. Vérifiez que le client reçoit bien une adresse IP de la plage spécifiée précédemment.
8. Que contient le fichier **/var/lib/dhcp/dhcpd.leases** sur le **serveur**, et qu'affiche la commande **dhcp-lease-list** ?
9. Vérifiez que les deux machines peuvent communiquer via leur adresse IP, à l'aide de la commande **ping**.

10. Modifiez la configuration du serveur pour que l'interface réseau du client reçoive l'IP statique 192.168.100.20 :

```
deny unknown-clients;      #empêche l'attribution d'une adresse IP à une
                           #station dont l'adresse MAC est inconnue du serveur

host client {
    hardware ethernet XX:XX:XX:XX:XX:XX;    #remplacer par l'adresse MAC
    fixed-address 192.168.100.20;
}
```

Vérifiez que la nouvelle configuration a bien été appliquée sur le client (éventuellement, désactivez puis réactivez l'interface réseau pour forcer le renouvellement du bail DHCP, ou utilisez la commande `dhclient -v`).

## Exercice 4. Donner un accès à Internet au client

A ce stade, le client est juste une machine sur notre réseau local, et n'a aucun accès à Internet. Pour remédier à cette situation, on va se servir de la machine **serveur** (qui, elle, a un accès à Internet via son autre carte réseau) comme d'une passerelle.

1. La première chose à faire est d'autoriser l'**IP forwarding** sur le serveur (désactivé par défaut, étant donné que la plupart des utilisateurs n'en ont pas besoin). Pour cela, il suffit de décommenter la ligne `net.ipv4.ip_forward=1` dans le fichier `/etc/sysctl.conf`. Pour que les changements soient pris en compte immédiatement, il faut saisir la commande `sudo sysctl -p /etc/sysctl.conf`.

💡 Vérifiez avec la commande `sysctl net.ipv4.ip_forward` que la nouvelle valeur a bien été prise en compte.

2. Ensuite, il faut autoriser la traduction d'adresse source (*masquerading*) en ajoutant la règle `iptables` suivante :  
`sudo iptables --table nat --append POSTROUTING --out-interface enp0s3 -j MASQUERADE`

Vérifiez à présent que vous arrivez à « pinguer » une adresse IP (par exemple `1.1.1.1`) depuis le client.

A ce stade, le client a désormais accès à Internet, mais il sera difficile de surfer : par exemple, il est même impossible de pinguer `www.google.com`. C'est parce que nous n'avons pas encore configuré de **serveur DNS** pour le client.

## Exercice 5. Installation du serveur DNS

De la même façon qu'il est plus facile de retenir le nom d'un contact plutôt que son numéro de téléphone, il est plus simple de mémoriser le nom d'un hôte sur un réseau (par exemple `www.cpe.fr`) plutôt que son adresse IP (`178.237.111.223`).

Dans les premiers réseaux, cette correspondance, appelée **résolution de nom**, se faisait via un fichier nommé `hosts` (présent dans `/etc` sous Linux<sup>1</sup>). L'inconvénient de cette méthode est que lorsqu'un nom ou une adresse IP change, il faut modifier les fichiers `hosts` de toutes les machines !

Par conséquent, avec l'avènement des réseaux à grande échelle, ce système n'était plus viable, et une autre solution, automatisée et centralisée cette fois, a été mise au point : DNS (Domain Name Server). Généralement, le serveur DNS utilisé est soit celui mis à disposition par le fournisseur d'accès à Internet, soit un DNS public (comme celui de Google : `8.8.8.8`, ou celui de Cloudflare : `1.1.1.1`).

Il est aussi très commun d'utiliser un serveur DNS privé, interne à l'organisation, afin de pouvoir résoudre les noms des machines locales. Pour les requêtes extérieures, le serveur DNS privé passe alors la main à un DNS externe.

Il existe de nombreux serveurs DNS, mais le plus commun sous UNIX est **Bind9** (Berkeley Internet Name Daemon v.9).

1. Sur le serveur, commencez par installer **bind9**, puis assurez-vous que le service est bien actif.
2. A ce stade, Bind n'est pas configuré et ne fait donc pas grand chose. L'une des manières les simples de le configurer est d'en faire un serveur cache : il ne fait rien à part mettre en cache les réponses de serveurs externes à qui il transmet la requête de résolution de nom.

💡 Le binaire (= programme) installé avec le paquet **bind9** ne s'appelle ni **bind** ni **bind9** mais **named**...

Nous allons donc modifier son fichier de configuration : **/etc/bind/named.conf.options**. Dans ce fichier, décommentez la partie **forwarders**, et à la place de **0.0.0.0**, renseignez les IP de DNS publics comme **1.1.1.1** et **8.8.8.8** (en terminant à chaque fois par un point virgule). Redémarrez le serveur **bind9**.

3. Sur le client, retentez un ping sur **www.google.fr**. Cette fois ça devrait marcher ! On valide ainsi la configuration du DHCP effectuée précédemment, puisque c'est grâce à elle que le client a trouvé son serveur DNS.
4. Sur le client, installez le navigateur en mode texte **lynx** et essayez de surfer sur **fr.wikipedia.org** (bienvenue dans le passé...)

## Exercice 6. Configuration du serveur DNS pour la zone **tpadmin.local**

L'intérêt d'un serveur DNS privé est principalement de pouvoir résoudre les noms des machines du réseau local. Pour l'instant, il est impossible de pinguer par leur nom **client** depuis **serveur** et inversement.

1. Modifiez le fichier **/etc/bind/named.conf.local** et ajoutez les lignes suivantes :

```
zone "tpadmin.local" IN {
    type master;                // c'est un serveur maître
    file "/etc/bind/db.tpadmin.local"; // lien vers le fichier de définition de zone
};
```

2. Créez une copie appelée **db.tpadmin.local** du fichier **db.local**. Ce fichier est un fichier configuration typique de DNS, constitué d'**enregistrements DNS** (cf. cours). Dans le nouveau fichier, remplacez toutes les références à **localhost** par **tpadmin.local**, et l'adresse **127.0.0.1** par l'adresse IP du serveur.

💡 La ligne **root.tpadmin.local.** indique en fait une adresse mail du responsable technique de cette zone, où le symbole **@** est remplacé par le premier point. **Attention également à ne pas oublier le point final, qui représente la racine DNS** ; on ne le met pas dans les navigateurs, mais il est indispensable dans les fichiers de configuration DNS !

💡 Le champ **serial** doit être incrémenté à **chaque** modification du fichier. Généralement, on lui donne pour valeur la date suivie d'un numéro sur deux chiffres, par exemple **2019031401**.

3. Maintenant que nous avons configuré notre fichier de zone, il reste à configurer le fichier de zone *inverse*, qui permet de convertir une adresse IP en nom.

Commencez par rajouter les lignes suivantes à la fin du fichier **named.conf.local** :

```
zone "100.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192.168.100";
};
```

---

1. On le trouve aussi sous Windows, dans **C:\Windows\System32\drivers\etc**

Créez ensuite le fichier `db.192.168.100` à partir du fichier `db.127`, et modifiez le de la même manière que le fichier de zone. Sur la dernière ligne, faites correspondre l'adresse IP avec celle du serveur (Attention, il y a un petit piège!).

4. Utilisez les utilitaires `named-checkconf` et `named-checkzone` pour valider vos fichiers de configuration :

```
$ named-checkconf named.conf.local
$ named-checkzone tpadmin.local /etc/bind/db.tpadmin.local
$ named-checkzone 100.168.192.in-addr.arpa /etc/bind/db.192.168.100
```

Modifiez le fichier `/etc/systemd/resolved.conf` et décommentez la section DNS.

5. Redémarrer le serveur Bind9. Vous devriez maintenant être en mesure de « pinguer » les différentes machines du réseau.

**Pour les plus rapides :**

## **Exercice 7. Installation d'un serveur web**

Installez et configurez un serveur web de votre choix (Apache ou nginx), et configurez le serveur pour qu'il réponde aussi au nom de `www`.

## **Exercice 8. Installation d'un serveur de temps**

Installez et configurez un serveur de temps (paquet `ntp`).