Administration Système sous Linux (Ubuntu Server)

Grégory Morel

2018-2019

CPE Lyon

Cinquième partie

Réseau

Rappels : adressage IPv4

Classes d'adresses IPv4

Rappels

Adresse IPv4 = 4 octets séparés par des points : `n1.n2.n3.n4`

2 parties : net id et host id

2 hôtes ayant même *net id* communiquent directement ; sinon il faut un routage

Classe	net id	1ers bits	1ère adresse	Dernière adresse	Nb réseaux	Nb hôtes
A	1 octet	0	0.0.0.0	127.255.255.255	2^7	2^{24}
B	2 octets	10	128.0.0.0	191.255.0.0	2^{14}	2^{16}
C	3 octets	110	192.0.0.0	223.255.255.0	2^{21}	2^8
D[1]	indéf.	1110	224.0.0.0	239.255.255.255	2^{28} adresses	
E[2]	indéf.	1111	240.0.0.0	255.255.255.255	2^{28} adresses	

Obsolète, mais encore utilisé !

1. Adresses de multidiffusion *(multicast)*
2. Réservée pour un usage futur

Adresses privées

Chaque classe comporte une plage d'adresses non routables et réservées aux réseaux locaux / privés :

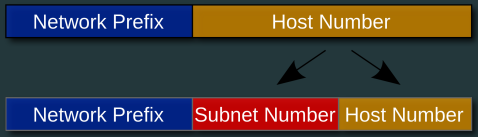
- `10.0.0.0` à `10.255.255.255`
- `172.16.0.0` à `172.31.255.255`
- `192.168.0.0` à `192.168.255.255`
- `127.0.0.1` est l'adresse de bouclage *(loopback)* : elle représente la machine elle-même

Par ailleurs,

- si l'*host id* ne contient que des `0`, l'adresse désigne le réseau lui-même
- si l'*host id* ne contient que des `1`, c'est une adresse de broadcast (permet d'envoyer une trame vers tous les hôtes du réseau)

Sous-réseaux

Niveau hiérarchique intermédiaire entre le réseau et les hôtes



Exemple

Une adresse de la classe B peut être vue comme 256 réseaux de 254 machines, plutôt que comme un seul réseau de 65 534 machines[1]

Pour communiquer entre elles, les machines doivent appartenir à un même réseau ou sous-réseau !

1. Chaque réseau ayant deux adreses réservées (`.0` et `.255`), on gagne en flexibilité d'adressage mais on perd en nombre de machines adressables (ici, 65024 vs 65534)

Masque de sous-réseau

Masque de sous-réseau

Masque binaire permettant de distinguer l'adresse de réseau et de sous-réseau de l'adresse de l'hôte dans une adresse IP :

Adresse réseau = Adresse IP &[1] Masque de sous-réseau

Calculer un masque de sous-réseau :

1. Déterminer N le nombre de machines dans le réseau, et ajouter 2
2. Trouver le plus petit p tel que $2^p \geq N$
3. Le masque de sous-réseau est constitué de $32 - p$ chiffres *1* suivis de p chiffres *0*

Il existe donc 32 masques de sous-réseau possibles[2]

1. Il s'agit ici du "ET" binaire
2. Autrefois, on évitait les deux masques constitués uniquement de 0 ou de 1

Notation CIDR (Classless Inter-Domain Routing)

Intérêt

- Notion de classes devenue obsolète
- Représentation compacte d'une plage d'adresses IP
- Diminue la taille des tables de routage

Notation : nombre de bits correspondant au sous-réseau dans l'adresse IP, précédés d'un "slash"

Exemple

La notation CIDR /19 fait référence au masque `11111111.11111111.11100000.00000000` soit 255.255.224.0

Exemple

Quelle est l'adresse de sous-réseau de la machine `91.198.174.2/19` ?

	Notation binaire	Notation décimale
	`01011011.11000110.10101110.00000010`	`91.198.174.2`
&	`11111111.11111111.11100000.00000000`	`255.255.224.0`
=	`01011011.11000110.10100000.00000000`	`91.198.160.0`

Quelle est l'adresse de l'hôte au sein de ce sous-réseau ?

	Notation binaire	Notation décimale
	`01011011.11000110.10101110.00000010`	`91.198.174.2`
&	`00000000.00000000.00011111.11111111`	`255.255.224.0`
=	`00000000.00000000.00001110.00000010`	`0.0.14.2`

Exemple

Comment subdiviser un 192.44.78.0/24 en 4 sous-réseaux ? Combien de machines seront adressables sur chaque sous-réseau ?

$\Rightarrow$ On a besoin de $\log_2(4) = 2$ bits pour distinguer les sous-réseaux. Les 4 sous-réseaux sont donc :

- `192.44.78.0/26`
- `192.44.78.64/26`
- `192.44.78.128/26`
- `192.44.78.198/26`

$\Rightarrow$ Il reste 6 bits pour adresser les machines dans chaque réseau, soit $2^6 - 2 = 62$ adresses possibles

On veut subdiviser le réseau 192.168.1.0/25 en sous-réseaux de 50 machines. Quel est le masque de sous-réseau ?

$\Rightarrow 2^5 < 50 + 2 \leq 2^6$ d'où $p = 6$. Le masque de sous-réseau est donc `11111111.11111111.11111111.11000000` soit 255.255.255.192

IPv6

Les adresses IPv4 sont arrivées à saturation le 3 février 2011 :

- mauvaise gestion initiale
- multiplication de la demande des particuliers
- explosion des dispositifs mobiles, des objets connectés...

⇒ une solution possible est le NAT *(Network Address Translation)*, qui permet à des machines d'un sous-réseau privé de communiquer avec le reste d'Internet
⇒ autre solution : passage à IPv6 :

- adresses sur 128 bits (chaque humain peut en posséder des milliards de milliards)
- notation : 8 groupes de 2 octets écrits en hexa et séparés par des : (ex. : `2001:0e36:2ed9:d4f0:021b:(0000):(0000):f81b`)

Encore peu répandu en 2018

Configuration réseau sous Ubuntu

Nomenclature des interfaces réseau

Classiquement, les interfaces réseau étaient nommées `eth0`, `eth1`… par le noyau

Problème

Les noms pouvaient changer après un redémarrage de la machine ou une modification de matériel (par exemple, `eth0` devient `eth1` et devient autorisée par le pare-feu !)

Une solution est de nommer les interfaces d'*après leur adresse MAC*. Cependant :

⇒ nécessite un répertoire root accessible en écriture (généralement pas le cas)
⇒ les adresses MAC ne sont pas toujours fixes (ex. : machines virtuelles)

Solution

Predictable Network Interface Names : le nom est choisi par le BIOS en fonction de l'emplacement sur la carte mère (ex. : `enp0s3`, pour *ethernet network peripheral 0 serial 3*)

Utilitaires de configuration réseau sous Ubuntu

- Jusqu'à Debian 9 : paquets `net-tools` et `wireless-tools` (`ifconfig`, `route`, `arp`, `netstat`…)
- Depuis Debian 9 : dépréciés[1] et remplacés respectivement par `iproute2` et `iw` ; syntaxe des différentes commandes unifiée :

Utilisation	Commande net-tools	Commande iproute2
Adressage	`ifconfig`	`ip addr`, `ip link`
Routage	`route`	`ip route`
Résolution d'adresses	`arp`	`ip neigh`
VLAN	`vconfig`	`ip link`
Tunneling	`iptunnel`	`ip tunnel`
Multicast	`ipmaddr`	`ip maddr`
Statistiques	`netstat`	`ss`

1. `net-tools` n'est plus installé par défaut avec la version Desktop d'Ubuntu 18.10, mais l'est toujours dans la version Server

Lister les interfaces

- jusqu'à la couche 2 / liaison (adresses MAC) : `ip l[ink] [show]`)
- jusqu'à la couche 3 / réseau (adresses IP) : `ip a[ddr] [show]`)

```
$ ip -4 a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
      valid_lft 79785sec preferred_lft 79785sec
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    inet 192.168.100.1/24 brd 192.168.100.255 scope global enp0s8
      valid_lft forever preferred_lft forever
```

1. Pour avoir des infos constructeur : `lshw -class network` (lshw = *list hardware*)

Configurer une interface réseau

Activer une interface : `ip link set enp0s3 up`

Désactiver une interface : `ip link set enp0s3 down`

Attribuer d'une adresse IP automatique par DHCP : `dhclient enp0s3`

Attribuer une adresse IP : `ip addr add 192.168.1.100/24 dev enp0s3`

Supprimer une adresse IP : `ip addr del 192.168.1.100/24 dev enp0s3`

Supprimer toute la configuration d'une interface : `ip addr flush enp0s3`

La configuration avec ces outils est `temporaire`!

Netplan

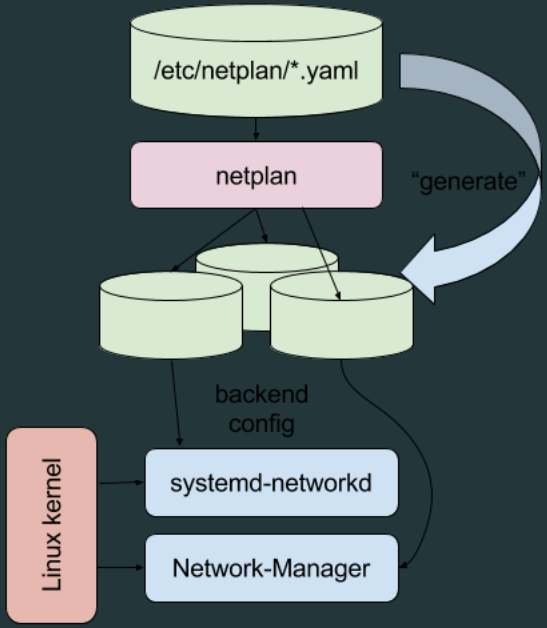
Avant Ubuntu 17.10, on configurait un réseau avec le paquet `ifupdown` et le fichier `/etc/network/interfaces`.

Depuis Ubuntu 17.10 : Netplan (fichiers de configuration au format YAML stockés dans `/etc/netplan`[1]).

Version	Renderer	Fichier
Desktop	NetworkManager	`01-network-manager-all.yaml`
Server	networkd	`01-netcfg.yaml`
Cloud	networkd	`50-cloud-init.yaml`

1. Par ordre d'importance, on peut trouver ces fichiers dans `/lib/netplan`, `/etc/netplan` et `/run/netplan`. Les fichiers sont traités dans l'ordre numérique, et un fichier remplace la configuration d'un fichier précédent pour une même interface

Netplan


/etc/netplan/*.yaml
netplan
“generate”
backend
config
Linux kernel
systemd-networkd
Network-Manager

Netplan

2 types de renderer :

- `networkd` (surtout version Server / valeur par défaut)
- `NetworkManager` (surtout version Desktop)

Quelques commandes à connaître :

- `netplan try` : essaie une configuration et revient en arrière en l'absence de confirmation
- `netplan [--debug] apply` : applique une configuration
- `systemctl restart systemd-networkd` : relancer le service

```
network :
  version : 2
  renderer : networkd
  ethernets :
    enp0s3 :
      dhcp4 : true
```

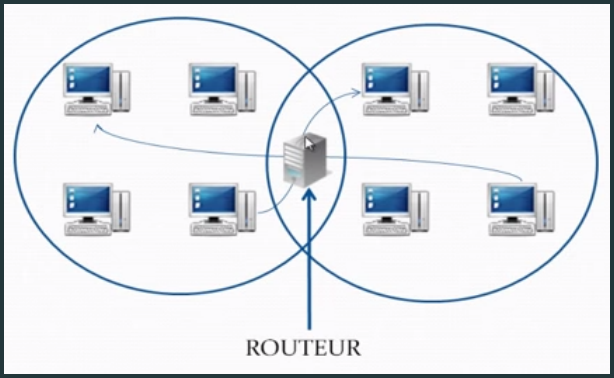
```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      addresses:
        - 10.10.10.2/24
```

```
network:
  version: 2
  renderer: networkd
  wifis:
    wlp2s0b1:
      dhcp4: yes
      access-points:
        "SSID_du_WiFi":
          password: "**********"
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      gateway4: 192.168.0.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

Routage

Rappel : routeur = machine servant d'interface entre des réseaux et assurant le transit des paquets



Un routeur a autant d'interfaces réseau que de réseaux auxquels il est connecté. Quand un routeur est connecté à plus de deux réseaux, on utilise une table de routage pour savoir où envoyer un paquet.

Visualiser la table de routage :

```
ip r[oute] [list]
```

Les anciennes commandes (`route`,`netstat -r`) sont cependant plus lisibles :

```
$ route
Kernel IP routing table
Destination     Gateway   Genmask         Flags   Iface
default         10.0.2.2  0.0.0.0         UG      enp0s3
10.0.2.0        0.0.0.0   255.255.255.0   U       enp0s3
10.0.2.2        0.0.0.0   255.255.255.255 UH      enp0s3
192.168.100.0   0.0.0.0   255.255.255.0   U       enp0s8
```

Ajouter une passerelle :

```
ip r add default via 192.168.1.1
```

Outils réseau

`ping <IP> | <nom d'hôte>` : vérifie si une machine distante répond

host

La commande `host` permet d'effectuer des requêtes DNS, notamment pour convertir des noms d'hôte en adresse IP et réciproquement :

```
$ host www.wikipedia.fr
www.wikipedia.fr has address 78.109.84.114
$ host 78.109.84.114
114.84.109.78.in-addr.arpa domain name pointer
wikimedia2.typhon.net.
```

host

On peut aussi obtenir les serveurs DNS qui gèrent un domaine :

```
$ host -t NS wikipedia.fr
wikipedia.fr name server b.dns.gandi.net.
wikipedia.fr name server c.dns.gandi.net.
wikipedia.fr name server a.dns.gandi.net.
```

Ou les serveurs de messagerie [1] pour ce domaine :

```
$ host -t MX wikipedia.fr
wikipedia.fr mail is handled by 50 fb.mail.gandi.net.
wikipedia.fr mail is handled by 10 spool.mail.gandi.net.
```

`host` affiche une réponse par ligne et est donc bien adapté aux scripts ; il est approprié pour des réponses simples et rapides

1. Les nombres correspondent aux priorités ; le plus petit nombre a la plus grande priorité

dig

La commande `dig` permet de réaliser des tâches similaires, mais est plus complet :

```
$ dig wikipedia.fr + short
78.109.84.114
$ dig NS wikipedia.fr + short
a.dns.gandi.net.
c.dns.gandi.net.
b.dns.gandi.net.
$ dig MX wikipedia.fr + short
50 fb.mail.gandi.net.
10 spool.mail.gandi.net.
$ dig -x 8.8.8.8 +short
google-public-dns-a.google.com.
```

nslookup

`nslookup` est historiquement le premier outil développé pour effectuer des requêtes DNS. Il est très utilisé mais moins adapté aux scripts.

```
$ nslookup wikipedia.fr
Server:     127.0.0.53
Address:    127.0.0.53#53

Non-authoritative answer:
Name:    wikipedia.fr
Address: 78.109.84.114
$ nslookup -type=mx wikipedia.fr
Server: 127.0.0.53
Address: 127.0.0.53#53

Non-authoritative answer:
wikipedia.fr mail exchanger = 50 fb.mail.gandi.net.
wikipedia.fr mail exchanger = 10 spool.mail.gandi.net.
```

Netfilter / Iptables

Netfilter

Module du noyau Linux ($\geq$ 2.4) permettant de filtrer et manipuler les paquets réseau qui passent dans le système

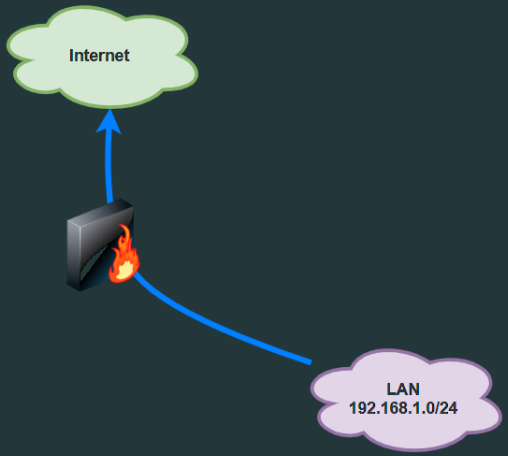
Pour info...

Il est prévu à terme que Netfilter soit (au moins en partie) remplacé par *nftables*, mais qui est encore en développement à ce jour.

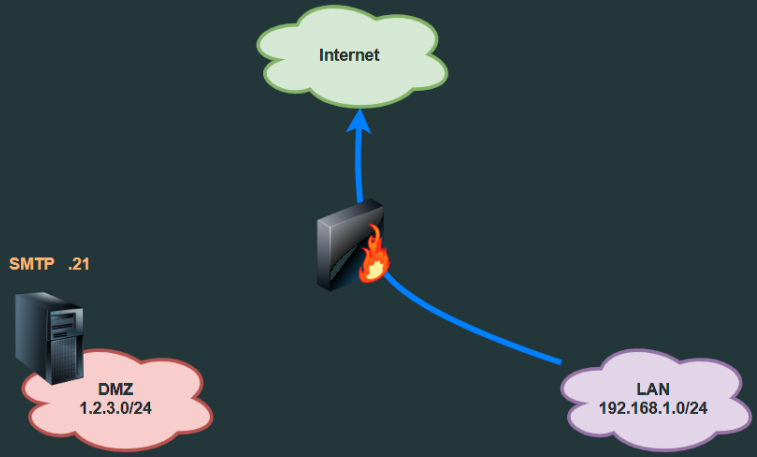
Netfilter est un *framework* ; il s'utilise via des utilitaires :

- iptables : bas niveau, pas toujours simple d'utilisation
- ufw (uncomplicated firewall) : alternative simplifiée à iptables
- Shorewall : une alternative à ufw

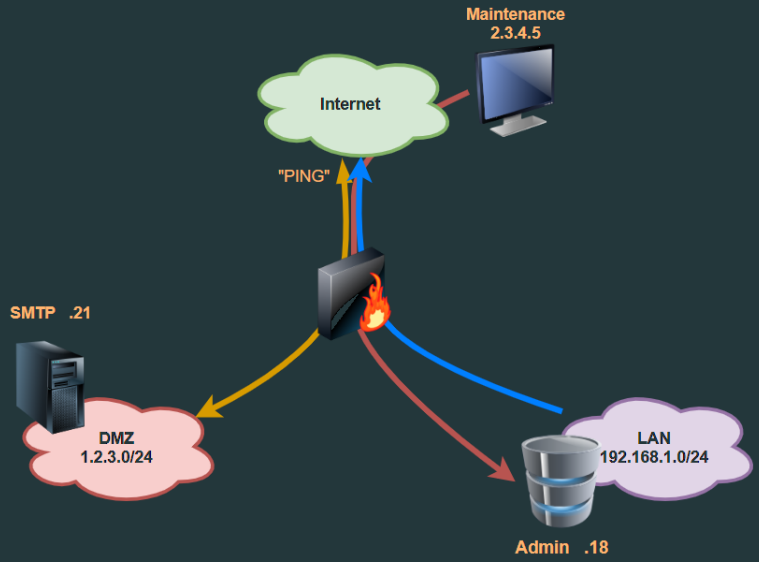
Netfilter



Netfilter



Netfilter


Maintenance
2.3.4.5
Internet
"PING"
SMTP .21
DMZ
1.2.3.0/24
LAN
192.168.1.0/24
Admin .18

4 règles à configurer pour ce réseau :

1. Autoriser les utilisateurs du LAN à accéder à Internet
2. Autoriser tout le monde à accéder au serveur mail
3. Autoriser le pare-feu à pinguer sur Internet
4. Autoriser une exception pour accéder au serveur sur le LAN

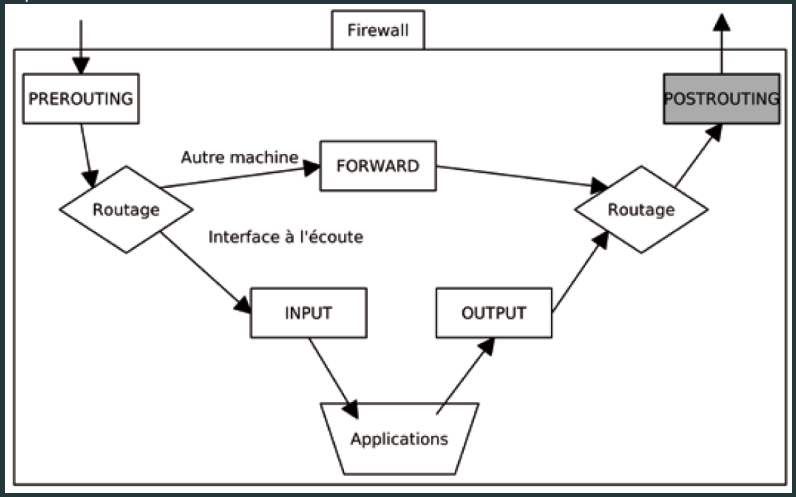
3 catégories :

paquets passant par le pare-feu (Règles 1 et 2) => `FORWARD`
paquets émis par le pare-feu (Règle 3) => `OUTPUT`
paquets à destination du pare-feu (Règle 4) => `INPUT`

On peut en rajouter 2 autres :

`PREROUTING` : traitement dès réception (ex. : modif. d'adresse de destination)
`POSTROUTING` : traitement avant émission (ex. : modif. d'adresse source)

Vie d'un paquet :



Principe de Netfilter : chaque paquet, entrant ou sortant, suit une ou plusieurs suites de règles appelées chaînes.

Principe des chaînes de règles

- Les règles sont lues dans l'ordre
- Dès qu'une règle est remplie, les suivantes sont ignorées
- Une règle est remplie si tous ses critères sont remplis
- Si un paquet passe toutes les règles, une règle par défaut peut être appliquée

Iptables

Pour gérer les chaînes, on utilise `iptables`.

- `iptables -vL` : liste les chaînes de règles
- `iptables -F` : supprime toutes les chaînes de règles ⚠
- `iptables -X` : supprime les chaînes définies par l'utilisateur

Lorsque le pare-feu n'est pas configuré, tout le trafic passe dans toutes les directions (`policy ACCEPT`) :

```
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Exemple

Pour interdire tous les paquets en provenance de `192.168.1.11` :

```
iptables -A INPUT -s 192.168.1.11 -j DROP
```

- `-A INPUT` : ajoute une règle à la chaîne `INPUT`
- `-s` : source du paquet
- `-j DROP` : cible (*jump*) de la règle, si elle est satisfaite

Attention !

Lorsque la politique (*policy*) par défaut sur `INPUT` est `ACCEPT`, la dernière règle est souvent `-j REJECT` pour tout interdire sauf les règles précédentes. Avec `-A INPUT`, la nouvelle règle est placée à la suite et n'aura donc aucun effet.

Autres exemples

Pour interdire les entrées par `enp0s3` :

```
iptables -A INPUT -i enp0s3 -j DROP
```

Pour interdire le protocole ICMP (*ping*) en entrée :

```
iptables -A INPUT -p icmp[1] -j DROP
```

Pour interdire les connexions entrantes à destination du port 80 :

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

Pour interdire toutes les connexions sauf celle de 10.0.0.1 :

```
iptables -A INPUT -s ! 10.0.0.1 -j DROP
```

Pour loguer les événements :

```
iptables -A INPUT -m limit --limit 5/min -j LOG
--log-prefix "iptables denied: " --log-level 7
```

1. Voir `/etc/protocols` pour les autres protocoles

Les chaînes sont regroupées en tableaux (ou *tables*). Il en existe 3 :

- `filter` : tableau par défaut ; chaînes de filtrage pour accepter, refuser, ignorer un paquet
- `nat` : chaînes de modification des adresses IP ou des ports sources ou destinataires
- `mangle` : chaînes permettant de modifier certains paramètres à l'intérieur des paquets IP

Ex. : *MASQUERADING* (= NAT source) : autoriser les machines avec une IP privée à accéder à Internet

```
$ iptables -t nat -a POSTROUTING -s 192.168.1.0/24 -o enp0s3 -j MASQUERADE
```

Important

Les règles sont transmises dynamiquement au noyau, et sont perdues au redémarrage de la machine ! Il faut donc penser à les sauvegarder puis les restaurer.

iptables propose les commandes `iptables-save` et `iptables-restore`, mais qui sont peu pratiques.

Le moyen le plus simple est d'utiliser le paquet `iptables-persistent` :

```
$ sudo netfilter-persistent save
...
$ sudo netfilter-persistent reload
```

UFW : Uncomplicated Firewall

Front-end pour NetFilter

`ufw enable / disable`	Active / Désactive le pare-feu
`ufw status [verbose]`	Affiche le statut du pare-feu
`ufw allow / deny [règle]`	Autorise / Refuse une connexion
`ufw logging on / off`	Active / Désactive la journalisation
`ufw app list`	Liste les services qui ont des règles `ufw`
`ufw app info APP`	Affiche les règles de *APP*
`ufw [--dry-run] règle`	Affiche les changements impliqués par *règle* sans les appliquer

Exemples :

- `ufw default allow` : autorise le trafic entrant selon les règles par défaut
- `ufw deny 80` : bloque le port 80
- `ufw deny http` : bloque le service HTTP
- `ufw deny apache` : bloque le service Apache