ENSEMBLE DE COMMANDES DE RÉSEAU LINUX

Introduction

Dans cet article, je vous présente un **ensemble de commandes de réseau Linux** que j'ai l'habitude d'utiliser afin de me permettre de configurer, maintenir, dépanner et gérer le réseau de mes serveurs Linux.

Les fichiers

Je vous liste d'abord des fichiers qui peuvent vous être utile lors de vos manipulations réseau.

- `/etc/hosts` : associer des adresses IP à des noms d'hôtes
- `/etc/networks` : associer des noms d'hôtes à des adresses IP
- `/etc/services` : noms des services tcp/udp ainsi que leur numéro de port

Les commandes

ping

Sans surprise, j'utilise la fameuse commande `ping` , elle est souvent exploitée pour tester la connectivité entre deux systèmes sur un réseau local (LAN) ou un réseau étendu (WAN). Pour information cette commande utilise le protocole ICMP (Internet Control Message Protocol) pour communiquer avec les nœuds d'un

réseau.

Vous indiquerez dans la commande simplement une adresse IP ou un nom d'hôte :

```
$ ping 192.168.1.3

PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.636 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=6.72 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.327 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.352 ms
64 bytes from 192.168.1.3: icmp_seq=5 ttl=64 time=0.265 ms
^C
--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.265/1.661/6.726/2.535 ms
```

Voici les options disponibles que je suis susceptible d'utiliser :

- `-c` : nombre de paquet
- `-i` : intervalle de relance

arp

J'utilise l'outil `arp` pour traduire les adresses IP en adresses physique. Ci-dessous quelques exemples d'utilisation :

Afficher l'adresse MAC de l'ip 192.168.1.3 :

```
$ arp 192.168.1.3

Address                  HWtype  HWaddress           Flags Mask            Iface
node01                   ether   02:42:ac:11:00:37   C                     ens3
```

Afficher la table ARP afin de connaître les adresses MAC des machines de votre réseau :

```
$ arp
```

```
Address                  HWtype  HWaddress           Flags Mask            Iface
172.17.0.120             ether   02:42:ac:11:00:78   C                     ens3
172.17.0.67              ether   02:42:ac:11:00:43   C                     ens3
172.17.0.1               ether   02:42:f9:85:98:f5   C                     ens3
172.17.0.130             ether   02:42:ac:11:00:82   C                     ens3
node01                   ether   02:42:ac:11:00:37   C                     ens3
172.16.0.47              ether   06:fe:47:9d:e6:b8   C                     ens3
```

Ajouter une entrée dans la table :

```
$ arp -s 192.168.1.3 02:42:ac:11:00:37
```

traceroute

Commande qui m'est très utile pour découvrir la source de blocage d'un paquet, puisqu'elle permet de suivre le chemin complet de votre système local à un autre système réseau. Elle affiche le nombre de sauts (adresses IP du routeur) dans le chemin emprunté pour atteindre le serveur final.

```
$ traceroute google.com

traceroute to google.com (216.58.215.46), 30 hops max, 60 byte packets
1  _gateway (192.168.0.1)  1.060 ms  2.942 ms  2.894 ms
2  10.44.64.1 (10.44.64.1)  22.544 ms  22.936 ms  22.843 ms
3  213-245-252-89.rev.numericable.fr (213.245.252.89)  22.817 ms  22.786 ms  23.535 m
4  172.19.132.146 (172.19.132.146)  23.969 ms  23.942 ms  23.985 ms
5  72.14.196.224 (72.14.196.224)  23.905 ms  23.853 ms  23.764 ms
6  108.170.244.193 (108.170.244.193)  23.684 ms 108.170.245.1 (108.170.245.1)  31.061
7  108.170.235.15 (108.170.235.15)  26.616 ms 72.14.237.93 (72.14.237.93)  27.103 ms
8  par21s17-in-f14.1e100.net (216.58.215.46)  26.457 ms  26.373 ms  26.276 ms
```

ip

Avant j'avais tendance à utiliser la commande `ifconfig` mais comme cette commande est de moins en moins maintenue , il est conseillé d'utiliser maintenant la commande `ip` permettant d'afficher et manipuler le routage, les périphériques réseaux et les interfaces.

Affichage d'informations

```
### Afficher des informations sur toutes les interfaces réseau
ip a

### N'afficher que de l'IPv4
ip  -4 a
### N'afficher que de l'IPv6
ip  -6 a

### Ne montre que l'interface ens3
ip a show ens3

## Affiche uniquement les interfaces à l'état UP
ip link ls up
```

Assigner/Supprimer une adresse IP à une interface réseau

```
### Assigner l'adresse 192.168.1.3 avec le masque sous réseau 255.255.255.0 à l'inter
ip a add 192.168.1.3/255.255.255.0 dev ens3
## ou
ip a add 192.168.1.3/24 dev ens3

### Supprimer l'adresse IP 192.168.1. de l'interface ens3
ip a del 192.168.1.3/24 dev ens3
```

Changer l'état d'une interface en UP ou DOWN

```
### Désactiver l’état du périphérique ens3
ip link set dev ens3 down

### Rétablir l’état du périphérique ens3
ip link set dev ens3 up
```

route

J'utilise la commande `route` afin d'afficher ou de manipuler la table de routage IP d'un système Linux. Elle est principalement utilisée pour définir un chemin de route statique dans les tables de route. Voici mes différents cas d'utilisation.

Afficher la table de routage IP

```
$ route

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.17.0.1      0.0.0.0         UG    0      0        0 ens3
10.32.0.0       *               255.240.0.0     U     0      0        0 weave
172.17.0.0      *               255.255.0.0     U     0      0        0 ens3
172.18.0.0      *               255.255.255.0   U     0      0        0 docker0
```

Ajouter un itinéraire réseau à la table de routage

```
$ sudo route add -net 192.168.1.3 netmask 255.255.255.0 gw 192.168.1.1 dev eth0
## ou
sudo route add -net 192.168.1.3/24 gw 192.168.1.1 dev eth0
```

Supprimer une entrée de route spécifique de la table de routage.

```
$ sudo route del -net 192.168.1.3/24 gw 192.168.1.1 dev eth0
```

nslookup

Je me sers de cette commande afin d'interroger le serveur DNS dans le but de traduire une adresse IP en un nom de domaine, ou inversement.

```
$ nslookup google.com

Server:         62.210.16.6
Address:        62.210.16.6#53

Non-authoritative answer:
Name:   google.com
Address: 216.58.213.142
```

dig

dig (Domain Information Groper) est un outil flexible pour interroger des informations liées au DNS telles que l'enregistrement A, CNAME, l'enregistrement MX, etc.

```
$ dig facebook.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57188
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;facebook.com.                  IN      A

;; ANSWER SECTION:
facebook.com.           84      IN      A       157.240.21.35

;; Query time: 1 msec
;; SERVER: 62.210.16.6#53(62.210.16.6)
;; WHEN: Mon Sep 23 11:39:32 UTC 2019
;; MSG SIZE  rcvd: 57
```

netstat

La commande `netstat` m'offre un moyen simple d'examiner chacune de mes connexions réseau et de mes sockets ouverts.

Écouter les ports et les programmes

Par exemple, la commande suivante affiche tous les ports TCP en mode d'écoute et les programmes en cours d'écoute.

```
$ sudo netstat -lntp

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      995/
tcp6       0      0 :::22                   :::*                    LISTEN      995/
```

L'état `LISTEN` signifie que le programme écoute et attend une connexion, mais vous pourriez aussi avoir l'état `ESTABLISHED` lorsqu'une connexion est déjà établie.

> **Information**
>
> Remplacez l'option `-t` par un `-u` pour examiner les ports UDP.

Afficher les statistiques par protocole

Par défaut, les statistiques sont affichées pour les protocoles TCP, UDP, ICMP et IP. Le paramètre `-s` peut être utilisé pour spécifier cet ensemble de protocoles.

```
### Statistique de tous les protocoles
netstat -s

### Statistique que pour le protocole TCP
netstat -st

### Statistique que pour le protocole TCP
netstat -su
```

nmap

L'utilitaire `nmap` me permet de vérifier le port ouvert sur un serveur.

```
$ nmap google.com

Starting Nmap 6.40 ( http://nmap.org ) at 2018-07-12 09:23 BST
Nmap scan report for google.com (172.217.166.78)
Host is up (0.0036s latency).
rDNS record for 172.217.166.78: bom05s15-in-f14.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 4.92 seconds
```

On peut aussi scanner une plage d'IP :

```
### Scanner par masque sous réseau (va scanner de l'ip 192.168.1.0 à 192.168.1.255)
nmap 192.168.1.0/24

### Scanner la plage IP de 192.168.1.1 à 192.168.1.200
nmap 192.168.1.1-200
```

tcpdump

Dans le cas d'une utilisation d'une interface graphique j'aurai plus tendance à utiliser l'analyseur de paquets libre et gratuit wireshark mais en ligne de commande j'utilise l'outil `Tcpdump`, qui est largement utilisé pour capturer et analyser les paquets TCP/IP transmis ou reçus sur un réseau ou sur une interface spécifique.

Voici les options que j'utilise le plus souvent :

- `-i` : utiliser une interface réseau
- `-A` : Voir le contenu d'un paquet IP
- `port` : Filtrage par port
- `host` : adresse de destination et/ou source
- `dst` : adresse de destination
- `src` : adresse source
- `-w` : capturer les paquets dans un fichier qui pourra être analysé plus tard
- `-r` : lire le paquet capturer

Ci-dessous quelques exemples d'utilisation :

```
### écouter le port http (80) sur l'interface ens3 et voir le contenu du
tcpdump -A -i ens3 port http
```

```
### Affiche tous les paquets en provenance de 192.168.1.2 vers 192.168.1.3 sur le po
tcpdump src host 192.168.1.2 and dst host 192.168.1.3 and port 22 and tcp

### stocker la capture dans le fichier capture.tdp
tcpdump -w capture.tdp

### lire la capture dans le fichier capture.tdp
tcpdump -v -r capture.tdp
```

UFW

UFW est un nouvel outil de configuration simplifié en ligne de commande qui est une alternative à l'outil iptables. Il est par défaut sur les distributions Debian et Ubuntu Linux et est utilisé pour ajouter/supprimer/modifier/réinitialiser les règles de filtrage de paquets du pare-feu de votre système.

Avant de rajouter des règles, il faut d'abord vérifier le statut de l'outil UFW à l'aide de la commande suivante:

```
$ sudo ufw status

Status: active
```

S'il n'est pas activé alors lancez la commande suivante :

```
$ sudo ufw enable
```

Par défaut, UFW bloquera toutes les connexions entrantes et autorisera toutes les connexions sortantes. Cela signifie que toute personne essayant d'accéder à votre serveur ne pourra pas se connecter à moins que vous n'ouvriez spécifiquement un port, tandis que toutes les applications et tous les services exécutés sur votre serveur pourront accéder au monde extérieur.

Autoriser et refuser les connexions :

```
### Autoriser un protocole ou une ip
sudo ufw allow ssh
sudo ufw allow 80/tcp

### Autoriser l'accès en sortie à un serveur ssh
ufw allow out 22/tcp


### Autoriser l'accès en entré (de l'extérieur) à notre serveur en ssh
ufw allow in 22/tcp

## Seul l'ip 192.168.1.3 est autorisée à accéder à notre serveur en ssh
ufw allow from 192.168.1.3 to any port 5789

### Autoriser une plage de port
sudo ufw allow 1000:2000/tcp
```

Pour refuser vous remplacez `"allow"` par `"deny"`. N'oubliez pas de charger vos nouvelles règles avec la commande ci-dessous :

```
$ ufw reload
```

Voici la commande pour vérifier l'état actuel de votre firewall :

```
$ ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22                         ALLOW IN    Anywhere
22 (v6)                    ALLOW IN    Anywhere (v6)
```

Chaque règle possède un numéro que vous pouvez lister avec la commande suivante :

```
$ ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)
```

Supprimer simplement une règle d'après son numéro :

```
$ sudo ufw delete [numéro]
```

Conclusion

Voici les commandes réseau Linux que j'utilise habituellement, nous aurons peut-être l'occasion de voir plus en détails certaines d'entre elles. N'hésitez pas aussi à me dire dans l'espace commentaire quel outil utilisez-vous le plus !