Linux

- Firewall / Pare-feu
- Hetzner Cloud - IPv6 Setup on Ubuntu 22.04
- Runcloud + Iptable + redirection + proxypass
- comment créer des alias de commandes temporaires et persistants ?

Firewall / Pare-feu

Installation d'un pare-feu

Installation de iptables

```
apt install iptables
```

Création du process et des règles

Je créer un fichier firewall qui va être exécuter au démarrage et gérer par systemctl.

Son but et d'indiquer quoi faire quand on start le process et quand on le stop et la gestion du status.

```
nano /etc/init.d/firewall
```

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          Firewall
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Firewall
# Description:       Firewall1
### END INIT INFO

IPT=/sbin/iptables
IP6T=/sbin/ip6tables

do_start() {
    # Efface toutes les regles en cours. -F toutes. -X utilisateurs
    $IPT -t filter -F
```

```
    $IPT -t filter -X
    $IPT -t nat -F
    $IPT -t nat -X
    $IPT -t mangle -F
    $IPT -t mangle -X
    $IP6T -t filter -F
    $IP6T -t filter -X
    $IP6T -t mangle -F
    $IP6T -t mangle -X
    # strategie (-P) par defaut : bloc tout l'entrant le forward et autorise le sortant
    $IPT -t filter -P INPUT DROP
    $IPT -t filter -P FORWARD DROP
    $IPT -t filter -P OUTPUT ACCEPT
    $IP6T -t filter -P INPUT DROP
    $IP6T -t filter -P FORWARD DROP
    $IP6T -t filter -P OUTPUT ACCEPT
    # Loopback
    $IPT -t filter -A INPUT -i lo -j ACCEPT
    $IPT -t filter -A OUTPUT -o lo -j ACCEPT
    $IP6T -t filter -A INPUT -i lo -j ACCEPT
    $IP6T -t filter -A OUTPUT -o lo -j ACCEPT
    # Permettre a une connexion ouverte de recevoir du trafic en entree
    $IPT -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
    $IP6T -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

    echo "firewall started [OK]"
}
# fonction qui arrete le firewall
do_stop() {
    # Efface toutes les regles
    $IPT -t filter -F
    $IPT -t filter -X
    $IPT -t nat -F
    $IPT -t nat -X
    $IPT -t mangle -F
    $IPT -t mangle -X
    $IP6T -t filter -F
    $IP6T -t filter -X
    $IP6T -t mangle -F
    $IP6T -t mangle -X
```

```
    # remet la strategie
    $IPT -t filter -P INPUT ACCEPT
    $IPT -t filter -P OUTPUT ACCEPT
    $IPT -t filter -P FORWARD ACCEPT
    $IP6T -t filter -P INPUT ACCEPT
    $IP6T -t filter -P OUTPUT ACCEPT
    $IP6T -t filter -P FORWARD ACCEPT
    #
    echo "firewall stopped [OK]"
}
# fonction status firewall
do_status() {
    # affiche les regles en cours
    clear
    echo Status IPV4
    echo -----------------------------------------------
    $IPT -L -n -v
    echo
    echo -----------------------------------------------
    echo
    echo status IPV6
    echo -----------------------------------------------
    $IP6T -L -n -v
    echo
}
case "$1" in
    start)
        do_start
        exit 0
    ;;
    stop)
        do_stop
        exit 0
    ;;
    restart)
        do_stop
        do_start
        exit 0
    ;;
    status)
```

```
        do_status
        exit 0
    ;;
    *)
        echo "Usage: /etc/init.d/firewall {start|stop|restart|status}"
        exit 1
    ;;
esac
```

Nous avons ici la configuration la plus bloquante (trafic sortant uniquement): Uniquement le trafic local et les connexion établie vers l'extérieur en IPv4 et IPv6.

Pour ajouter des autorisation, par exemple l'ICMP (ping), dans `do_start()` ajouter:

```
# ICMP
    $IPT -t filter -A INPUT -p icmp -j ACCEPT
    $IP6T -t filter -A INPUT -p ipv6-icmp -j ACCEPT
```

Votre `do_start()` ressemble à ça:

```
do_start() {
    # Efface toutes les regles en cours. -F toutes. -X utilisateurs
    $IPT -t filter -F
    $IPT -t filter -X
    $IPT -t nat -F
    $IPT -t nat -X
    $IPT -t mangle -F
    $IPT -t mangle -X
    $IP6T -t filter -F
    $IP6T -t filter -X
    $IP6T -t mangle -F
    $IP6T -t mangle -X
    # strategie (-P) par defaut : bloc tout l'entrant le forward et autorise le sortant
    $IPT -t filter -P INPUT DROP
    $IPT -t filter -P FORWARD DROP
    $IPT -t filter -P OUTPUT ACCEPT
    $IP6T -t filter -P INPUT DROP
    $IP6T -t filter -P FORWARD DROP
    $IP6T -t filter -P OUTPUT ACCEPT
    # Loopback
    $IPT -t filter -A INPUT -i lo -j ACCEPT
```

```
    $IPT -t filter -A OUTPUT -o lo -j ACCEPT
    $IP6T -t filter -A INPUT -i lo -j ACCEPT
    $IP6T -t filter -A OUTPUT -o lo -j ACCEPT
    # Permettre a une connexion ouverte de recevoir du trafic en entree
    $IPT -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
    $IP6T -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

 # ICMP
    $IPT -t filter -A INPUT -p icmp -j ACCEPT
    $IP6T -t filter -A INPUT -p ipv6-icmp -j ACCEPT

    echo "firewall started [OK]"
}
```

Voici une liste de règles souvent utile

```
    # ICMP
    $IPT -t filter -A INPUT -p icmp -j ACCEPT
    $IP6T -t filter -A INPUT -p ipv6-icmp -j ACCEPT

    # DNS:53
    $IPT -t filter -A INPUT -p tcp --dport 53 -j ACCEPT
    $IPT -t filter -A INPUT -p udp --dport 53 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 53 -j ACCEPT
    $IP6T -t filter -A INPUT -p udp --dport 53 -j ACCEPT

    #FTP:20+21 - FTP pasv: 10 090 - 10 100
    $IPT -t filter -A INPUT -p tcp --dport 20 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 20 -j ACCEPT
    $IPT -t filter -A INPUT -p tcp --dport 21 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 21 -j ACCEPT
    $IPT -t filter -A INPUT -p tcp --dport 10090:10100 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 10090:10100 -j ACCEPT

    # SSH:22
    # ATTENTION, indiques bien ton port personnalisé si tu l'as changé
    $IPT -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 22 -j ACCEPT

    # HTTP:80 (Serveur Web)
```

```
    $IPT -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 80 -j ACCEPT

    # HTTPS:443 (Serveur Web)
    $IPT -t filter -A INPUT -p tcp --dport 443 -j ACCEPT
    $IP6T -t filter -A INPUT -p tcp --dport 443 -j ACCEPT

    # SNMP:161 (Monitoring)
    $IPT -t filter -A INPUT -p udp --dport 161 -j ACCEPT
    $IP6T -t filter -A INPUT -p udp --dport 161 -j ACCEPT

    # SMTP:25 (Mail)
□$IPT -t filter -A INPUT -p tcp --dport 25 -j ACCEPT
□$IP6T -t filter -A INPUT -p tcp --dport 25 -j ACCEPT

    # POP3:110 (Mail)
□$IPT -t filter -A INPUT -p tcp --dport 110 -j ACCEPT
□$IP6T -t filter -A INPUT -p tcp --dport 110 -j ACCEPT

□# IMAP:143 (Mail)
□$IPT -t filter -A INPUT -p tcp --dport 143 -j ACCEPT
□$IP6T -t filter -A INPUT -p tcp --dport 143 -j ACCEPT
```

Il faut maintenant rendre exécutable de fichier

```
chmod +x /etc/init.d/firewall
```

> Les prochaines étapes vont rendre actif le pare-feu. Si vous êtes en SSH et que une règle n'autorise pas le SSH vous risquer de vous auto bloqué a la prochaine connexion. Assurez-vous d'avoir un accès directe à la machine pour gérer le cas ou vous seriez dans l'incapacité de vous reconnecter.

Premier démarrage du pare-feu

```
/etc/init.d/firewall start
```

Gestion par systemctl

```
update-rc.d firewall defaults
systemctl enable firewall
systemctl start firewall
systemctl status firewall
```

Cela devrais vous indiquer quelque chose du genre:

```
● firewall.service - LSB: Firewall
   Loaded: loaded (/etc/init.d/firewall; generated)
   Active: active (exited) since *** ****-**-** **:**:** ****; * ago
     Docs: man:systemd-sysv-generator(8)
  Process: ***** ExecStart=/etc/init.d/firewall start (code=exited, status=0/SUCCESS)

***. ** **:**:** monitor systemd[1]: Starting LSB: Firewall...
***. ** **:**:** monitor firewall[30462]: firewall started [OK]
***. ** **:**:** monitor systemd[1]: Started LSB: Firewall.
```

Hetzner Cloud - IPv6 Setup on Ubuntu 22.04

Hello,

I ordered a VPS Cloud from Hetzner, and it's true that once delivered, you need to configure IPv6.

For my setup, I'm using a CX22 - Cloud, running Ubuntu 22.04.

Start by updating your Cloud server:

```
apt update && apt upgrade -y
```

Next, navigate to the `50-cloud-init.yaml` file located in netplan:

```
nano /etc/netplan/50-cloud-init.yaml
```

Here is the default configuration:

```
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        eth0:
            addresses:
            - 2a01:XXX:XXX:XXX::1/64
            dhcp4: true
            match:
                macaddress: 96:XX:XX:XX:XX:XX
            nameservers:
                addresses:
                - 2a01:XXX:XXXX::add:1
                - 2a01:XXX:XXXX::add:2
```

```
        routes:
        -   on-link: true
            to: default
            via: fe80::1
        set-name: eth0
```

You can make a backup copy:

```
cp /etc/netplan/50-cloud-init.yaml /etc/netplan/50-cloud-init.yaml.bck
```

Now, on line 11, change `2a01:XXX:XXX:XXX::1/64` to `2a01:XXX:XXX:XXX::24/64`. For this example, I changed it to `::24`.

```
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        eth0:
            addresses:
            - 2a01:XXX:XXX:XXX::24/64 # change ::1 to ::24 here
            dhcp4: true
            match:
                macaddress: 96:XX:XX:XX:XX:XX
            nameservers:
                addresses:
                - 2a01:XXX:XXXX::add:1
                - 2a01:XXX:XXXX::add:2
            routes:
            -   on-link: true
                to: default
                via: fe80::1
            set-name: eth0
```

Save the changes and run the following command:

```
netplan try
```

If you see this message:

```
** (process:16923): WARNING **: 02:56:51.010: Permissions for /etc/netplan/50-cloud-init.yaml are too open.
Netplan configuration should NOT be accessible by others.
```

Press Enter, then run this command to fix the permissions issue:

```
sudo chmod 600 /etc/netplan/50-cloud-init.yaml
```

Then, rerun the command:

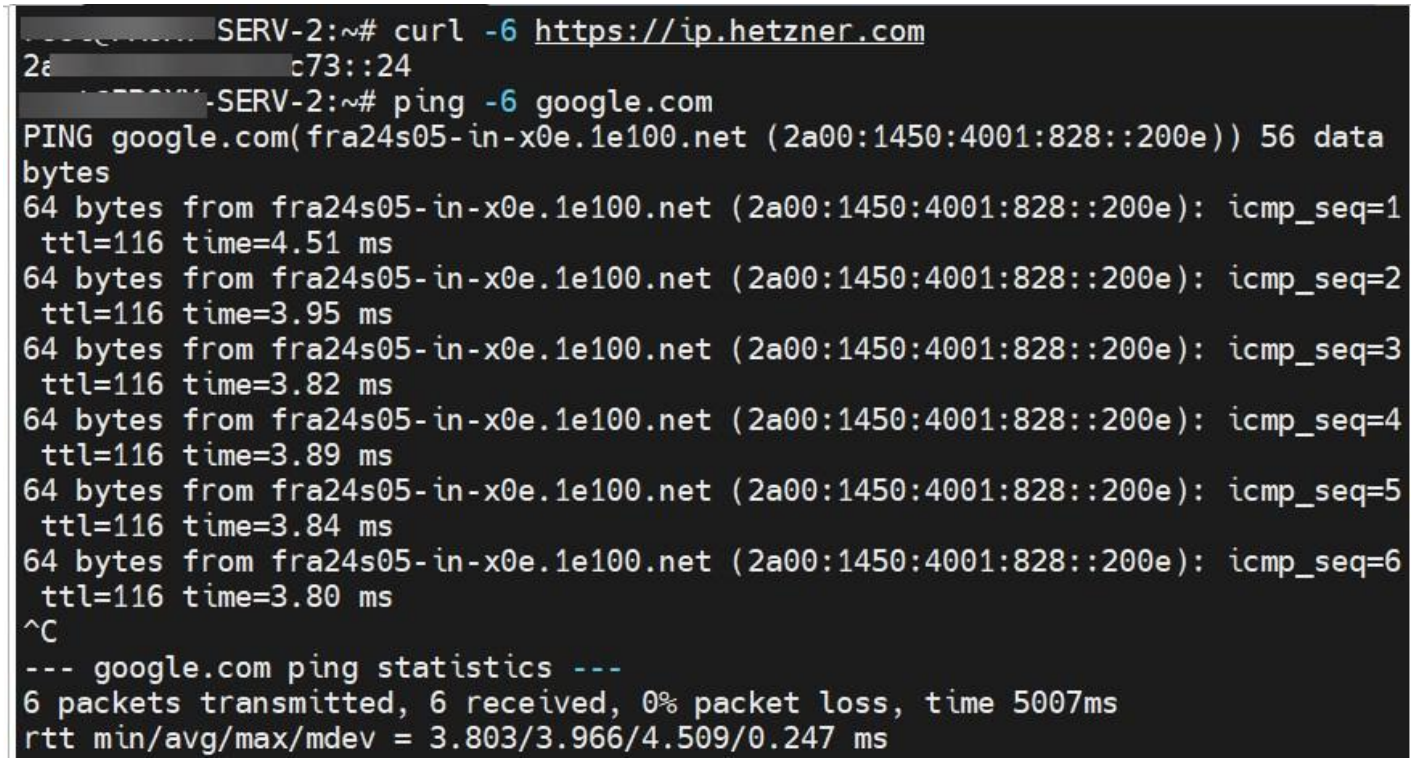
```
netplan try
```

Finally, apply the changes with the following command:

```
netplan apply
```

To verify everything is working, try these commands:

```
curl -6 https://ip.hetzner.com
ping -6 google.com
ip a
```



Runcloud + Iptable + redirection + proxypass

il faut desactiver firewalld de runcloud pour minimiser les soucis

Quand on met ce script

```
#!/bin/bash
sleep 2
echo "FIREWALL OK";

#Install iptables if you haven't already
#Alternatively use packet manager of your choice
#apt-get install iptables

#Allow all incoming traffic to begin with
sudo iptables -P INPUT ACCEPT
#Clean out any existing input rules. You may also remove the "INPUT" argument and run only "iptables -F" to clear all chains. When doing so, make sure there are no rules in other chains that you still need (list via "iptables -L"), for example Oracle cloud servers will have preset rules, which should not be removed.
sudo iptables -F INPUT

#Allow all internal connections
sudo iptables -A INPUT -i lo -j ACCEPT

#Allow continuing setup connections
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

#Allow ssh, adjust port if you run it on non-default

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT #SSH
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 34210 -j ACCEPT
```

```
sudo iptables -A INPUT -p udp --dport 22 -j ACCEPT #SSH
sudo iptables -A INPUT -p udp --dport 443 -j ACCEPT
sudo iptables -A INPUT -p udp --dport 34210 -j ACCEPT

# Ajouter les r�gles de redirection
sudo iptables -t nat -A PREROUTING -p udp --dport 50120 -j DNAT --to-destination 176.9.63.27:40120
sudo iptables -t nat -A PREROUTING -p tcp --dport 50120 -j DNAT --to-destination 176.9.63.27:40120
# Ajouter la r�gle de masquage
sudo iptables -t nat -A POSTROUTING -j MASQUERADE



# Activer l'IP forwarding temporairement
sudo sysctl -w net.ipv4.ip_forward=1

sudo iptables -P INPUT DROP

#Block all forwarding
#sudo iptables -P FORWARD DROP

#Allow all outgoing
sudo iptables -P OUTPUT ACCEPT
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
sudo iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP
```

Nous ne pouvons plus ping alors il faut modifier le resolv.conf pour mettre des nameserver autre que celui present mais apres un reboot ça disparait.

Pour ajouter des lignes permanentes dans `/etc/resolv.conf` vous pourriez installer le `resolvconf` ligne de commande:

```
sudo apt install resolvconf
```

alors tu dois éditer `/etc/resolvconf/resolv.conf.d/head` et ajoutez les lignes permanentes dont vous avez besoin. Par exemple:

```
nameserver 8.8.8.8
nameserver 1.1.1.1
```

et enfin exécutez ces commandes :

```
sudo resolvconf --enable-updates
sudo resolvconf -u
```

Un petit script pour reset l'iptables au cas ou

```
#!/bin/bash

# Vider toutes les règles
sudo iptables -F
sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -t raw -F

# Supprimer toutes les chaînes personnalisées
sudo iptables -X
sudo iptables -t nat -X
sudo iptables -t mangle -X
sudo iptables -t raw -X

# Réinitialiser toutes les politiques par défaut à ACCEPT
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT

# Vérifier les règles
sudo iptables -L -v -n
sudo iptables -t nat -L -v -n
```

Et ça devrait etre bon vérifié si ping google.com focntionne bien et que rundcloud agent fonctionne bien et est détecter par le panel rundcloud.

comment créer des alias de commandes temporaires et persistants ?

Lister les alias sous Linux

Pour lister les alias actuellement présents sur une machine Linux, ce n'est pas bien compliqué ! À partir du prompt du Terminal, il suffit d'exécuter cette commande :

```
alias
```

Comment créer un alias ?

Avant de créer un alias, vous devez savoir qu'il y a deux types d'alias :

- **Les alias temporaires** (valident dans la session en cours uniquement)
- **Les alias permanents** (qui seront toujours présents après redémarrage de la machine)

A. Créer un alias temporaire

Commençons par aborder la création d'un alias temporaire. La syntaxe pour créer un alias temporaire est la suivante :

```
alias <nom de l'alias>="<commande associée à l'alias>"
```

Prenons pour exemple la commande ci-dessous dont l'objectif est de lire en temps réel les dernières lignes du fichier de log "/var/log/auth.log" (lié à l'authentification sous Linux) :

```
sudo tail -f /var/log/auth.log
```

On pourrait associer cette commande à l'alias suivant :

```
authlog
```

Dans ce cas, l'alias suivant doit être créé :

```
alias authlog="sudo tail -f /var/log/auth.log"
```

Ensuite, il suffit d'exécuter la nouvelle commande pour appeler la commande complète :

```
authlog
```

Ce qui est cool, c'est que l'on peut profiter de l'autocomplétion lors de la saisie d'un alias, comme avec une commande classique, pour gagner encore plus de temps !

Le problème de cet alias temporaire, c'est qu'il est valide uniquement pour la session en cours...

B. Créer un alias persistant

Pour conserver un alias après redémarrage du système, ou après une déconnexion puis un reconnexion à une session Linux, il convient de **créer des alias persistants**. Pour cela, l'alias doit être **ajouté au fichier de configuration de votre profil shell**. Par défaut, Debian utilise Bash donc ce sera au sein du fichier "**~/.bashrc**.

Désormais, à partir d'un éditeur de texte tel que nano ou vim, il va falloir éditer ce fichier pour ajouter l'alias :

```
SRV-DEB-1: nano ~/.bashrc
```

Dans ce fichier, l'alias doit être ajouté en **respectant la même syntaxe que pour créer un alias temporaire**. La seule différence c'est que la commande est ajoutée dans un fichier plutôt que de l'exécuter dans le Terminal. On ajoute tout simplement le contenu à la fin du fichier. **On peut ajouter autant d'alias que l'on souhaite (un par ligne).**

```
# Alias pour lire le contenu du fichier auth.log
alias authlog="sudo tail -f /var/log/auth.log"
```

On enregistre et on ferme le fichier : le tour est joué !

on check nos alias

```
source ~/.bashrc
```

L'occasion de lister les alias avec la commande évoquée précédemment afin de voir qu'il est bien présent dans la liste !