Machine	IP eth0	IP eth1
ADMIN	192.168.1.254/24	
DMZ	192.168.1.1/24	192.168.2.1/24
INTERNET	192.168.2.254/24	

Admin : 192.168.1.254
DMZ: 192.168.1.1 - 192.168.43.254
Internet: 192.168.43.1

A. Un rapide coup d'œil sur le logiciel Netfilter

Q1: Installation

La commande à exécuter pour installer `arptables` est différente selon l'OS
Sous Debian/Ubuntu :

```
apt update && apt install -y arptables
```

Q2: Tables et chaines

`arptables` ne fournit qu'une seule table et c'est `filter` avec les chaines suivantes :

» La chaine INPUT
» La chaine OUTPUT
» La chaine FORWARD

Q3: Paramètres `arptables`

Un certain nombre de paramètres qui sont liés aux données qu'on retrouve au niveau de la couche 2 du modèle TCP/IP *(couche MAC Ethernet)*
- source-mac
- destination-mac
- h-length
- opcode
- h-type
- proto-type

Q4: Règles par défaut

Pour refuser tous les messages entrant qui ne correspondent pas aux règles précédentes:

```
arptables -P INPUT DROP
```

Pour refuser tous les messages sortant qui ne correspondent pas aux règles précédentes:

```
arptables -P OUTPUT DROP
```

Pour refuser tous les messages quoi traversent la machine et qui ne correspondent pas aux règles précédentes:

```
arptables -P FORWARD DROP
```

Q5: Accepter tous les messages entrant d'eth2

```
arptables -A INPUT -i eth2 -j ACCPET
```

Q6 Accepter le trafic depuis le poste Admin

```
arptables -A INPUT -s 192.168.1.254 -j ACCEPT
```

B ebtables

Q1: Qu'est-ce qu'un bridge

File failed to load: https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/extensions/MathMenu.js rs cartes réseaux (réelles ou virtuelles) entre elles

Q2: Installation

La commande a éxecuter pour installer `ebtables` est différente selon l'OS
Sous Debian/Ubuntu :

```
apt update && apt install -y ebtables
```

Q3: Tables et chaines

`ebtables` fournit les tables suivantes (pour chaque table, ses chaines)

» `filter`
  › INPUT
  › OUTPUT
  › FORWARD

» `nat`
  › PREROUTING
  › OUTPUT
  › POSTROUTING

» `broute`
  › BROUTING

Q4: Paramètres `ebtables`

interface réelle via laquelle le paquet sera envoyé/reçu

» `logical-in`
» `logical-out`

Q5: filtrage basique

Soit `aa:bb:cc:dd:ee:ff` l'adresse `MAC` d'une machine qu'on veut accepter ...

```
ebtables -P INPUT DROP
ebtables -A INPUT -s aa:bb:cc:dd:ee:ff -j ACCEPT
ebtables -P OUTPUT ACCEPT
```

Q6: Anti-spoofing

```
ebtables -N VMS
ebtables -A FORWARD -p ip -i vnet+ -j VMS
ebtables -P VMS DROP
```

Puis rajouter toutes les machines connus ainsi que leurs adresses `MAC` :

```
ebtables -A VMS -p ip --ip-src <IP_MACHINE> -s <MAC_MACHINE> -j ACCEPT
```

Exemple:

```
ebtables -A VMS -p ip --ip-src 192.168.1.254 -s aa:bb:cc:dd:ee:ff -j ACCEPT
```

B. Docker et iptables

Q1 Nouvelles interface

Lorsqu'on lance le daemon Docker, ce dernier crée une interface virtuelle nommée `docker0`

Q2 Nouvelles interface - Particularité

`docker0` est le bridge par défaut de Docker. Lorsqu'on crée un conteneur, si on ne spécifie pas à quel réseau ce dernier doit appartenir, alors docker le met systématiquement dans `docker0`

Q3 Nouvelles règles `iptables`

Docker utilise les tables `nat` et la table `filter` et donc crée des règles dans ces deux tables

Il crée les entrés pour isoler les conteneurs. Et aussi pour leur permettre de traverser le réseau via le nat en utilisant l'ip de la machine hôte.

filter

En `FORWARD` et crée de nouvelles chaines : `DOCKER` , `DOCKER-ISOLATION-STAGE-1` , `DOCKER-ISOLATION-STAGE-2` et `DOCKER-USER`

```
Chain INPUT (policy ACCEPT 89 packets, 20049 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0    0 DOCKER-USER  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0    0 DOCKER-ISOLATION-STAGE-1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
    0    0 DOCKER     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT     all  --  docker0 docker0  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 94 packets, 9173 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER (1 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0    0 DOCKER-ISOLATION-STAGE-2  all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0    0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0    0 DROP       all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
    0    0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-USER (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0    0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

---

nat

En `PREROUTING` , `OUTPUT` et en `POSTROUTING`

```
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1   328 DOCKER     all  --  *      *       0.0.0.0/0            0.0.0.0/0            ADDRTYPE match dst-type LOCAL

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

Chain OUTPUT (policy ACCEPT 6 packets, 348 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0    0 DOCKER     all  --  *      *       0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT 6 packets, 348 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0    0 MASQUERADE  all  --  *      !docker0  172.17.0.0/16        0.0.0.0/0

Chain DOCKER (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0    0 RETURN     all  --  docker0 *       0.0.0.0/0            0.0.0.0/0
```

Q4: règles `arptables` ou `ebtables`

Docker n'utilise ni `arptables` ni `ebtables` et donc ne crée que des règles `iptables`

Q5:

Il n'existe pas de moyen de suspendre l'utilisation d'iptables par docker.
le choix d'utiliser ou pas `iptables` doit se faire avant le démarrage du deamon Docker.
Pour désactiver l'utilisation d' `iptables` par Docker, il faut rajouter le paramètre `--iptables=false`

selon l'OS:

» OpenRC :
Il faut rajouter un paramètre au lancement du service `init.d` docker dans `/etc/init.d/docker`

```
# This is the pid file created/managed by start-stop-daemon
DOCKER_SSD_PIDFILE=/var/run/$BASE-ssd.pid
DOCKER_LOGFILE=/var/log/$BASE.log
DOCKER_OPTS= # IL FAUT RAJOUTER ICI --iptables=false
```

```
DOCKER_DESC="Docker"
```

ou bien si votre système le prend en charge, dans le fichier `/etc/default/docker`

» SystemD
Il faut créer le fichier `/etc/systemd/system/docker.service.d/iptables.conf` avec le contenu suvant :

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H fd:// --iptables=false
```

Q6

Oui, cette méthode permet de désactiver complétement l'utilisation d'iptables de manière permanente.

Avec docker network

Q1:

Oui de nouvelles interfaces sont apparus :
- vethee5s4e1@if24: (virtual ethernet)
- veth1011384@if26: (virtual ethernet)
- vethee5e95d@if28: (virtual ethernet)
- br-a8cf550bd964: (bridge)

Q2:

Ces interfaces correspondent aux conteneurs crée. Une interface virtuelle pour chaque conteneur. Et le bridge correspond au réseau partagé par les conteneurs : `docker-compose_nodes`

```
$> docker network ls
```

```
NETWORK ID          NAME                      DRIVER              SCOPE
ffb924e21bb0        bridge                    bridge              local
a8cf550bd964        docker-compose_nodes      bridge              local
2c652c4d384e        host                      host                local
ccd107c0d2fe        none                      null                local
```

Q3 Nouvelles règles `iptables`

`filter`

```
Chain INPUT (policy ACCEPT 79 packets, 11368 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER-USER  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DOCKER-ISOLATION-STAGE-1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      br-a8cf550bd964  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
    0     0 DOCKER     all  --  *      br-a8cf550bd964  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  br-a8cf550bd964 !br-a8cf550bd964  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  br-a8cf550bd964 br-a8cf550bd964  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
    0     0 DOCKER     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 docker0  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 85 packets, 8019 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  !br-a8cf550bd964 br-a8cf550bd964  0.0.0.0/0            172.18.0.2           tcp dpt:80
    0     0 ACCEPT     tcp  --  !br-a8cf550bd964 br-a8cf550bd964  0.0.0.0/0            172.18.0.3           tcp dpt:22
    0     0 ACCEPT     tcp  --  !br-a8cf550bd964 br-a8cf550bd964  0.0.0.0/0            172.18.0.4           tcp dpt:22

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER-ISOLATION-STAGE-2  all  --  br-a8cf550bd964 !br-a8cf550bd964  0.0.0.0/0            0.0.0.0/0
    0     0 DOCKER-ISOLATION-STAGE-2  all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      br-a8cf550bd964  0.0.0.0/0            0.0.0.0/0
```

```
    0   0 DROP       all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
    0   0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-USER (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0   0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

nat

```
Chain PREROUTING (policy ACCEPT 2 packets, 746 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1   328 DOCKER     all  --  *      *       0.0.0.0/0            0.0.0.0/0            ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT 2 packets, 746 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 74 packets, 5141 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER     all  --  *      *       0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT 74 packets, 5141 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      !br-a8cf550bd964  172.18.0.0/16        0.0.0.0/0
    0     0 MASQUERADE  all  --  *      !docker0  172.17.0.0/16        0.0.0.0/0
    0     0 MASQUERADE  tcp  --  *      *       172.18.0.2           172.18.0.2           tcp dpt:80
    0     0 MASQUERADE  tcp  --  *      *       172.18.0.3           172.18.0.3           tcp dpt:22
    0     0 MASQUERADE  tcp  --  *      *       172.18.0.4           172.18.0.4           tcp dpt:22

Chain DOCKER (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  br-a8cf550bd964 *       0.0.0.0/0            0.0.0.0/0
    0     0 RETURN     all  --  docker0 *       0.0.0.0/0            0.0.0.0/0
    0     0 DNAT       tcp  --  !br-a8cf550bd964 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80 to:172.18.0.2:80
    0     0 DNAT       tcp  --  !br-a8cf550bd964 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:2222 to:172.18.0.3:22
    0     0 DNAT       tcp  --  !br-a8cf550bd964 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:2002 to:172.18.0.4:22
```

Q4: règles `arptables` ou `ebtables`

Docker n'utilise ni `arptables` ni `ebtables` et donc ne crée que des règles `iptables`

Q5

Docker rajoute des règles `iptables` pour permettre aux conteneurs de sortir en OUTPUT via le nat
et de rediriger les ports exposés vers les conteneurs (en nat aussi)

Q6: Sans `iptables` :

Impossible de joindre les conteneurs, car il n'y a plus de règles pour autoriser le forwarning et le nattage. Les conteneurs n'ont plus accès au réseau.

Q7:

Attention, ceci est une configuration minimale qui permet uniquement l'accès inter-conteneur du même réseau, et l'accès des conteneurs vers internet.

```
iptables -A FORWARD -j DOCKER-USER
iptables -A DOCKER-USER -p all -d 172.18.0.0/16 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A DOCKER-USER -p all -s 172.18.0.0/16 -j ACCEPT
iptables -t nat -A POSTROUTING -p all -s 172.18.0.0/16 -j MASQUERADE
```

filter

```
Chain DOCKER-USER (1 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             172.18.0.0/16        ctstate RELATED,ESTABLISHED
ACCEPT     all  --  172.18.0.0/16        anywhere
```

nat

```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.18.0.0/16        anywhere
```

c. Aller plus loin

```
iptables -A INPUT -i eth1,eth0 -m state --state ESTABLISHED,RELATED -d 192.168.1.1 -j ACCEPT
iptables -A INPUT -i eth1,eth0 -m state --state NEW -p tcp -s 192.168.1.254 -d 192.168.1.1 --dport 2002 -j ACCEPT
iptables -A INPUT -i eth1,eth0 -m state --state NEW -p tcp -s 192.168.1.254 -d 192.168.1.1 --dport 2222 -j ACCEPT
iptables -A INPUT -i eth1,eth0 -m state --state NEW -p tcp -d 192.168.1.1 --dport 80 -j ACCEPT
iptables -A INPUT -i eth1,eth0 -d 192.168.1.1 -j DROP
iptables -A INPUT -i eth1,eth0 -d 172.18.0.0/16 -j DROP
```

Question Bonus:

Admettant que l'on veuille rediriger temporairement le trafique qui arrive sur les ports 80 et 443 vers une autre machine dont l'adresse est 192.168.1.27

il suffit de rajouter les règles de forward/nat adéquate dans la chaine DOCKER-USER puis la flusher une fois le service rétabli.

```
#!/bin/bash
LISTEN_PORT=80
DEST_PORT=80
DEST_IP=192.168.1.27


enable(){
    iptables -t nat -N CUSTOM_PREROUTING
    iptables -t nat -N CUSTOM_POSTROUTING
    iptables -t nat -I PREROUTING 1 -j CUSTOM_PREROUTING
    iptables -t nat -I POSTROUTING 1 -j CUSTOM_POSTROUTING
    iptables -t nat -A CUSTOM_PREROUTING -i eno1 -p tcp --dport $LISTEN_PORT -j DNAT --to $DEST_IP:$DEST_PORT
    iptables -t nat -A CUSTOM_POSTROUTING -p tcp -o eno1 -j MASQUERADE

    iptables -N CUSTOM_FORWARD
    iptables -A DOCKER-USER -j CUSTOM_FORWARD
    iptables -A CUSTOM_FORWARD -p tcp -d $DEST_IP --dport $DEST_PORT -j ACCEPT
    iptables -A CUSTOM_FORWARD -p tcp -s $DEST_IP --sport $DEST_PORT -j ACCEPT
}

disable(){
    iptables -t nat -F CUSTOM_PREROUTING
    iptables -t nat -D PREROUTING -j CUSTOM_PREROUTING
    iptables -t nat -X CUSTOM_PREROUTING

    iptables -t nat -F CUSTOM_POSTROUTING
    iptables -t nat -D POSTROUTING -j CUSTOM_POSTROUTING
    iptables -t nat -X CUSTOM_POSTROUTING

    iptables -F CUSTOM_FORWARD
    iptables -D DOCKER-USER -j CUSTOM_FORWARD
    iptables -X CUSTOM_FORWARD
}

if [ "$1" = "enable" ];
then
    enable
else
    disable
fi
```