

# Installation Services Réseaux

Le 15 Juin 2025

COLLOT Grégoire  
MARECAILLE-HENAUT Mattieu  
SIMSEK Dilara

# Sommaire

<b>Sommaire</b>	<b>2</b>
<b>Introduction</b>	<b>3</b>
<b>Préambule :</b>	<b>4</b>
Installation de notre image Debian	4
<b>Serveur Apache</b>	<b>5</b>
Mise en place d'un conteneur	5
Mise en place d'Apache	6
Mise en place d'un container client	7
<b>Apache - Analyse des trames WireShark:</b>	<b>8</b>
Analyse d'établissement d'une connexion apache sans redirection de port.	8
Analyse d'établissement d'une connexion apache avec redirection de port.	9
Analyse d'établissement d'une connexion entre deux réseaux.	11
<b>Serveur SSH</b>	<b>12</b>
Mise en place du container SSH	12
Mise en place du serveur SSH	12
Connexion externe vers le serveur SSH	13
<b>SSH - Analyse des trames WireShark</b>	<b>15</b>
Analyse de l'établissement d'une connexion SSH sans redirection de port.	15
Analyse de l'établissement d'une connexion SSH avec redirection de port.	16
Analyse d'établissement d'une connexion SSH entre deux réseaux.	17
<b>Configuration théorique du réseau</b>	<b>18</b>
Topologie du réseau	18
Plan d'adressage	19
<b>Docker Compose</b>	<b>20</b>
<b>Conclusion</b>	<b>22</b>

# Introduction

Nous sommes étudiants en première année de BUT Informatique à l'IUT de Maubeuge.

Dans le cadre de la ressource 2.03 : Installation de services réseaux, il nous a été demandé d'installer, de configurer et de déployer sur un réseau deux services, Web et SSH.

Nous avons utilisé Docker afin de pouvoir déployer et tester nos différents services durant ce projet.

Ainsi, nous allons voir la création et la configuration d'un serveur web sous Apache et voir comment nous pouvons le déployer pour qu'il soit accessible depuis deux réseaux différents. Nous continuons avec le déploiement d'un service SSH, nécessitant les mêmes contraintes.

Ayant utilisé Docker pour nos différents tests, nous avons réalisé une conception théorique d'une architecture réseau multi-réseaux pour une application concrète de nos déploiement de service.

Enfin, nous allons voir la possibilité de déployer plusieurs services en même temps et de manière automatique avec l'utilisation de Docker Compose.

## Préambule :

Toutes commande de terminale seront de couleur verte.

Nous avons utilisé un ordinateur sous debian pour l'utilisation de docker. Si vous souhaitez utiliser un autre OS, nous vous recommandons de suivre la documentation officielle de Docker.

<https://docs.docker.com/engine/install/>

### Étape 1 : Ajout du dépôt contenant Docker.

# Add Docker's official GPG key:

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=\$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]

https://download.docker.com/linux/debian \

\$(. /etc/os-release && echo "\$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

### Étape 2 : Installer Docker

sudo apt-get install docker-ce docker-ce-cli containerd.io

docker-buildx-plugin docker-compose-plugin

### Étape 3 : Vérifier la bonne installation

sudo docker run hello-world

Un retour console "Hello World" apparaît.

```
dilarasimsek@MacBook-Air-de-Dilara ~ % docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

## Installation de notre image Debian

Pour mettre en place nos différent containers, il faut tout d'abord télécharger une image de Debian avec la commande suivante :

docker pull debian

Vous pouvez ensuite vous assurez du bon téléchargement de l'image et voir les autres images présentes pour votre docker.

docker images

```
dilarasimsek@MacBook-Air-de-Dilara ~ % docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
debian        latest   bd73076dc2cd   2 weeks ago    181MB
hello-world    latest   0b6a027b5cf3   4 months ago   20.4kB
```

# Serveur Apache

## Mise en place d'un conteneur

Vous avez plusieurs options possibles pour mettre en place votre container.

Voici les deux commandes afin de mettre en place notre container Docker avec ou sans redirection de port (Apache utilise le port 80 par défaut).

`docker run --name server -ti debian /bin/bash` ← Sans redirection de port

`docker run --name server -p 8080:80 -ti debian /bin/bash` ← Avec redirection de port

Explication des paramètre de la commande run de docker :

- `--name server` ← Attribue un nom à notre container, ici "server".
- `-p 80:8080` ← Permet d'attribuer le port que l'on souhaite.
- `-ti` ← Permet d'indiquer que nous souhaitons interagir avec notre console (Terminal Interactive).
- `debian` ← correspond au nom de l'image que l'on souhaite utiliser.
- `/bin/bash` ← Récupération de la console bash du container.
- `--env http_proxy="http://proxy-rech.uphf.fr:3128"` ← permet de mettre en place un proxy.

Pour arrêter votre container, taper simplement `exit`.

Nous pouvons voir les différents containers mis en place grâce à `docker ps -a`.

```
dilarasimsek@MacBook-Air-de-Dilara ~ % docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
f5693530784d   hello-world "/hello"   17 minutes ago   Exited (0) 13 minutes ago   -   mystifying_lehmann
a4c124076a67   debian    "/bin/bash" 5 days ago     Exited (0) 5 days ago     -   client
944a2cd2b086   debian    "/bin/bash" 5 days ago     Exited (0) 19 minutes ago  -   server
```

Vous pouvez lancer un de vos containers avec `docker start -i nomducontainer`.

Pour supprimer un container, il suffit d'utiliser `docker rm nomducontainer`.

## Mise en place d'Apache

Nous venons de mettre en place notre container qui va accueillir notre serveur Apache. Il faut maintenant installer les applications nécessaires.

Tout d'abord nous devons mettre à jour notre fichier source :

```
apt update
```

Nous allons avoir besoin de plusieurs outils pour utiliser notre serveur Apache.

- Apache2 est le paquet contenant Apache.
- Net-tools va nous permettre de récupérer l'adresse IP de votre machine.
- inetutils-ping permet de réaliser des pings vers des adresses ip.

Vous pouvez ainsi installer tous les packages nécessaire avec la commande suivante :

```
apt install apache2 net-tools inetutils-ping -y
```

Une fois les packages installés, nous allons venir lancer apache avec la ligne de commande :

```
service apache2 start
```

```
root@a3e9751d0b83:/# service apache2 start
Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2.
Set the 'ServerName' directive globally to suppress this message
.
```

Si vous ne savez plus si votre serveur apache est allumé vous pouvez utiliser `service apache2 status`.

Si vous avez effectué une modification des fichiers apache veuillez à bien redémarrer votre serveur pour que les modifications soient actives : `service apache2 restart`

Enfin, si vous souhaitez éteindre votre serveur apache : `service apache2 stop`

Pour réaliser nos différents tests, vous pouvez récupérer l'adresse IP de vos machines avec `ifconfig`.

Vous pouvez ainsi réaliser un test de ping entre votre machine et votre container server.

```
ping x.x.x.x
```

## Mise en place d'un container client

Si vous souhaitez tester votre connexion à votre serveur Apache, vous pouvez mettre en place un container client :

```
docker run --name server -ti debian /bin/bash
```

Vous devez installer deux paquets, Lynx permet d'avoir un navigateur web dans votre console. Le deuxième permet de réaliser des pings entre deux machines.

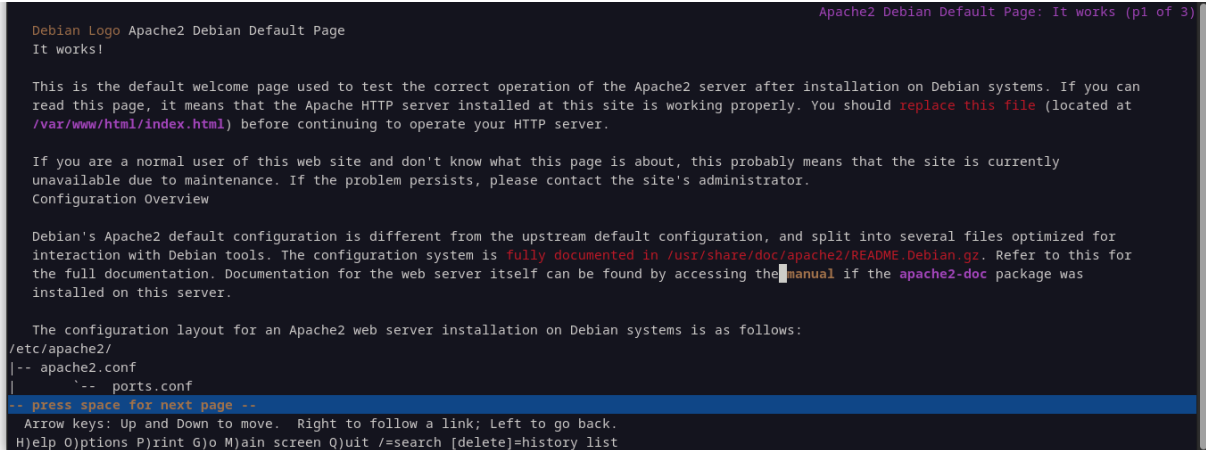
```
apt update  
apt install lynx inetutils-ping -y
```

Une fois les paquets installés vous pouvez réaliser un ping vers votre serveur pour directement voir la page d'accueil de votre serveur Apache avec lynx.

```
lynx 172.17.0.2
```

Par ailleurs, si vous utilisez une machine qui utilise un autre réseau, vous pouvez vous connecter grâce à l'adresse IP qui hoste votre container debian. Il faut ajouter le port qui redirige votre flux vers le container. Ici nous avons utilisé le port 8080.

```
lynx 196.168.5.215:8080
```



```
Debian Logo Apache2 Debian Default Page                                     Apache2 Debian Default Page: It works (p1 of 3)  
It works!  
  
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can  
read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at  
/var/www/html/index.html) before continuing to operate your HTTP server.  
  
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently  
unavailable due to maintenance. If the problem persists, please contact the site's administrator.  
Configuration Overview  
  
Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for  
interaction with Debian tools. The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for  
the full documentation. Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was  
installed on this server.  
  
The configuration layout for an Apache2 web server installation on Debian systems is as follows:  
/etc/apache2/  
|-- apache2.conf  
|   |-- ports.conf  
|-- press space for next page --  
Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.  
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

# Apache - Analyse des trames WireShark:

## Analyse d'établissement d'une connexion Apache sans redirection de port.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.17.0.4	TCP	76	47098 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=31420
2	0.000012841	172.17.0.1	172.17.0.4	TCP	76	[TCP Retransmission] 47098 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
3	0.000052504	172.17.0.4	172.17.0.1	TCP	76	80 → 47098 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM
4	0.000055756	172.17.0.4	172.17.0.1	TCP	76	[TCP Retransmission] 80 → 47098 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=
5	0.000082975	172.17.0.1	172.17.0.4	TCP	68	47098 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3142999874 TSecr=2
6	0.000084694	172.17.0.1	172.17.0.4	TCP	68	[TCP Dup ACK 5#1] 47098 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=
7	0.000215863	172.17.0.1	172.17.0.4	HTTP	510	GET / HTTP/1.1
8	0.000219027	172.17.0.1	172.17.0.4	TCP	510	[TCP Retransmission] 47098 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=
9	0.000269692	172.17.0.4	172.17.0.1	TCP	68	80 → 47098 [ACK] Seq=1 Ack=443 Win=64768 Len=0 TSval=2926158153 TSecr
10	0.000271829	172.17.0.4	172.17.0.1	TCP	68	[TCP Dup ACK 9#1] 80 → 47098 [ACK] Seq=1 Ack=443 Win=64768 Len=0 TSva
11	0.001710483	172.17.0.4	172.17.0.1	HTTP	3448	HTTP/1.1 200 OK (text/html)
12	0.001719443	172.17.0.4	172.17.0.1	TCP	3448	[TCP Retransmission] 80 → 47098 [PSH, ACK] Seq=1 Ack=443 Win=64768 Le
13	0.001769640	172.17.0.1	172.17.0.4	TCP	68	47098 → 80 [ACK] Seq=443 Ack=3381 Win=63232 Len=0 TSval=3142999876 TS
14	0.001774386	172.17.0.1	172.17.0.4	TCP	68	[TCP Dup ACK 13#1] 47098 → 80 [ACK] Seq=443 Ack=3381 Win=63232 Len=0
30	5.007456497	172.17.0.4	172.17.0.1	TCP	68	80 → 47098 [FIN, ACK] Seq=3381 Ack=443 Win=64768 Len=0 TSval=29261631
31	5.007482764	172.17.0.4	172.17.0.1	TCP	68	[TCP Retransmission] 80 → 47098 [FIN, ACK] Seq=3381 Ack=443 Win=64768
32	5.007789533	172.17.0.1	172.17.0.4	TCP	68	47098 → 80 [FIN, ACK] Seq=443 Ack=3382 Win=64128 Len=0 TSval=314300048
33	5.007806592	172.17.0.1	172.17.0.4	TCP	68	[TCP Retransmission] 47098 → 80 [FIN, ACK] Seq=443 Ack=3382 Win=64128
34	5.007873133	172.17.0.4	172.17.0.1	TCP	68	80 → 47098 [ACK] Seq=3382 Ack=444 Win=64768 Len=0 TSval=2926163161 TS
35	5.007891487	172.17.0.4	172.17.0.1	TCP	68	[TCP Dup ACK 34#1] 80 → 47098 [ACK] Seq=3382 Ack=444 Win=64768 Len=0

On remarque qu'il y a une tentative de connexion, cela est visible avec le SYN sur le port 80 et un enchaînement de SYN-ACK et de ACK qui montre que la connexion est correctement établie en utilisant le 3-Way Handshake.

1	0.000000000	172.17.0.1	172.17.0.4	TCP	76	47098 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3142999874 TSecr=0 WS=128
2	0.000012841	172.17.0.1	172.17.0.4	TCP	76	[TCP Retransmission] 47098 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3142999874 TSecr=0 WS=128
3	0.000052504	172.17.0.4	172.17.0.1	TCP	76	80 → 47098 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2926158153 TSecr=3142999874 WS=128
4	0.000055756	172.17.0.4	172.17.0.1	TCP	76	[TCP Retransmission] 80 → 47098 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2926158153 TSecr=3142999874 WS=128
5	0.000082975	172.17.0.1	172.17.0.4	TCP	68	47098 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3142999874 TSecr=2926158153
6	0.000084694	172.17.0.1	172.17.0.4	TCP	68	[TCP Dup ACK 5#1] 47098 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3142999874 TSecr=2926158153

## Remarque :

On peut voir aussi que la machine envoie des paquets TCP retransmission car il attend une réponse du serveur et ne la reçoit pas assez rapidement, il renvoie donc sa demande jusqu'au moment où le serveur lui répond.

On retrouve des erreurs du type [TCP ACKed unseen segment] et [TCP Previous segment not captured] d'après la doc se serait principalement des erreurs liées à la capture de la trame.

Pour plus d'information :

- <https://osqa-ask.wireshark.org/questions/46134/tcp-acked-unseen-segment/>
- <https://www.chappell-university.com/post/wireshark-expert-explained-acked-segment-th-at-wasn-t-captured>

## Analyse d'établissement d'une connexion Apache avec redirection de port.



1	0.000000000	127.0.0.1	127.0.0.1	TCP	76 37322 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1367693502 T...
2	0.000026234	127.0.0.1	127.0.0.1	TCP	76 8080 → 37322 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=1...
3	0.000046017	127.0.0.1	127.0.0.1	TCP	68 37322 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1367693503 TSecr=1367693502
4	0.000278270	127.0.0.1	127.0.0.1	HTTP	526 GET / HTTP/1.1
5	0.000298827	127.0.0.1	127.0.0.1	TCP	68 8080 → 37322 [ACK] Seq=1 Ack=459 Win=65152 Len=0 TSval=1367693503 TSecr=1367693...
6	0.000448711	172.17.0.1	172.17.0.2	TCP	76 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1822407360 TSec...
7	0.000462638	172.17.0.1	172.17.0.2	TCP	76 [TCP Retransmission] 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM ...
8	0.000467594	172.17.0.1	172.17.0.2	TCP	76 [TCP Retransmission] 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM ...
9	0.000503747	172.17.0.2	172.17.0.1	TCP	76 80 → 33030 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1711...
10	0.000531970	172.17.0.2	172.17.0.1	TCP	76 [TCP Retransmission] 80 → 33030 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460...
11	0.000563546	172.17.0.1	172.17.0.2	TCP	68 33030 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1822407360 TSecr=1711494688
12	0.000565625	172.17.0.1	172.17.0.2	TCP	68 [TCP Dup ACK 11#1] 33030 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=182240736...
13	0.000762578	172.17.0.1	172.17.0.2	HTTP	526 GET / HTTP/1.1
14	0.000774569	172.17.0.1	172.17.0.2	TCP	526 [TCP Retransmission] 33030 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=458 TSval=...
15	0.000851664	172.17.0.2	172.17.0.1	TCP	68 80 → 33030 [ACK] Seq=1 Ack=459 Win=64768 Len=0 TSval=1711494688 TSecr=1822407360
16	0.000855824	172.17.0.2	172.17.0.1	TCP	68 [TCP Dup ACK 15#1] 80 → 33030 [ACK] Seq=1 Ack=459 Win=64768 Len=0 TSval=1711494...
17	0.002579123	172.17.0.2	172.17.0.1	HTTP	3448 HTTP/1.1 200 OK (text/html)
18	0.002587484	172.17.0.2	172.17.0.1	TCP	3448 [TCP Retransmission] 80 → 33030 [PSH, ACK] Seq=1 Ack=459 Win=64768 Len=3380 TSv...
19	0.002627848	172.17.0.1	172.17.0.2	TCP	68 33030 → 80 [ACK] Seq=459 Ack=3381 Win=63232 Len=0 TSval=1822407362 TSecr=171149...
20	0.002631893	172.17.0.1	172.17.0.2	TCP	68 [TCP Dup ACK 19#1] 33030 → 80 [ACK] Seq=459 Ack=3381 Win=63232 Len=0 TSval=1822...
21	0.002732518	127.0.0.1	127.0.0.1	HTTP	3448 HTTP/1.1 200 OK (text/html)

On remarque qu'il y a une tentative de connexion, cela est visible avec le SYN sur le port 8080 et un enchaînement de SYN-ACK et de ACK qui montre que la connexion est correctement établie en utilisant le 3-Way Handshake.

6	0.000448711	172.17.0.1	172.17.0.2	TCP	76 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1822407360 TSecr=1711494688
7	0.000462638	172.17.0.1	172.17.0.2	TCP	76 [TCP Retransmission] 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1822407360 TSecr=1711494688
8	0.000467594	172.17.0.1	172.17.0.2	TCP	76 [TCP Retransmission] 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1822407360 TSecr=1711494688
9	0.000503747	172.17.0.2	172.17.0.1	TCP	76 80 → 33030 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1711494688 TSecr=1822407360
10	0.000531970	172.17.0.2	172.17.0.1	TCP	76 [TCP Retransmission] 80 → 33030 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1711494688 TSecr=1822407360
11	0.000563546	172.17.0.1	172.17.0.2	TCP	68 33030 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1822407362 TSecr=1711494688

Cependant nous pouvons voir l'utilisation de la redirection de port. En effet, les différentes demandes entre les deux machines passent par le port 37322 <-> 8080 <-> 33030 <-> 80.

```

76 37322 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1367693502 TSecr=1367693502
76 8080 → 37322 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=1367693502 TSecr=1367693502
68 37322 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1367693503 TSecr=1367693502
526 GET / HTTP/1.1
68 8080 → 37322 [ACK] Seq=1 Ack=459 Win=65152 Len=0 TSval=1367693503 TSecr=1367693502
76 33030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1822407360 TSecr=1711494688

```

Comme pour la connexion sans redirection de port on retrouve des Tcp retransmission pour s'assurer de l'intégrité et du bon envoi des fichiers.

### Remarque :

Au début des échanges, des paquets "doublons" [dup ACK] sont transmis entre les deux machines.

Ceci est dû au fait que les échanges mettent parfois trop de temps à envoyer un ACK, ce qui entraîne une renonciation de ces données et un renvoi des paquets. Cependant, durant cette période, vu que la machine à reçu un ACK, le deuxième ACK envoyé pour confirmer le même paquet est donc transformé en [dup ACK].

## Analyse d'établissement d'une connexion Apache entre deux réseaux.

47 35.143739384	192.168.5.182	TCP	80	51798	8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460	192.168.5.215	TCP	80	51798	8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
40 35.143810152	192.168.5.182	TCP	80	51798	80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460	192.168.5.215	TCP	80	51798	80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
40 35.143816662	192.168.5.182	TCP	80	51798	80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460	192.168.5.215	TCP	80	51798	80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
50 35.143861208	192.168.5.182	TCP	76	80	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460	192.168.5.215	TCP	76	80	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460
53 35.143889335	192.168.5.215	TCP	76	8080	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460	192.168.5.182	TCP	76	8080	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460
53 35.148089111	192.168.5.182	TCP	68	51798	8080 [ACK] Seq=1 Ack=1 Win=131712 Len=0	192.168.5.215	TCP	68	51798	8080 [ACK] Seq=1 Ack=1 Win=131712 Len=0
54 35.148134110	192.168.5.182	TCP	68	51798	80 [ACK] Seq=1 Ack=1 Win=131712 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=1 Ack=1 Win=131712 Len=0
55 35.148140611	192.168.5.182	TCP	68	51798	80 [ACK] Seq=1 Ack=1 Win=131712 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=1 Ack=1 Win=131712 Len=0
56 35.151530224	192.168.5.182	HTTP	308	GET / HTTP/1.0		192.168.5.215	HTTP	308	GET / HTTP/1.0	
57 35.151575947	192.168.5.182	HTTP	308	GET / HTTP/1.0		192.168.5.215	HTTP	308	GET / HTTP/1.0	
58 35.151582929	192.168.5.182	TCP	308	51798	80 [PSH, ACK] Seq=1 Ack=1 Win=65024 Len=0	192.168.5.215	TCP	308	51798	80 [PSH, ACK] Seq=1 Ack=1 Win=65024 Len=0
59 35.151583648	192.168.5.182	TCP	68	80	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.215	TCP	68	80	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0
60 35.151585059	192.168.5.182	TCP	68	80	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.215	TCP	68	80	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0
61 35.151671661	192.168.5.215	TCP	68	8080	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.182	TCP	68	8080	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0
62 35.153193477	192.168.5.215	TCP	2964	80	51798 [PSH, ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.182	TCP	2964	80	51798 [PSH, ACK] Seq=1 Ack=241 Win=65024 Len=0
63 35.153202353	192.168.5.215	TCP	2964	51798	80 [PSH, ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.182	TCP	2964	51798	80 [PSH, ACK] Seq=1 Ack=241 Win=65024 Len=0
64 35.153213800	192.168.5.215	TCP	1516	8080	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.182	TCP	1516	8080	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=0
65 35.153267423	192.168.5.215	TCP	1516	8080	51798 [PSH, ACK] Seq=1 Ack=241 Win=65024 Len=0	192.168.5.182	TCP	1516	8080	51798 [PSH, ACK] Seq=1 Ack=241 Win=65024 Len=0
66 35.153281753	192.168.5.215	HTTP	515	HTTP/1.1 200 OK (text/html)		192.168.5.182	HTTP	515	HTTP/1.1 200 OK (text/html)	
67 35.153281753	192.168.5.215	HTTP	515	HTTP/1.1 200 OK (text/html)		192.168.5.182	HTTP	515	HTTP/1.1 200 OK (text/html)	
68 35.153297423	192.168.5.215	HTTP	515	HTTP/1.1 200 OK (text/html)		192.168.5.182	HTTP	515	HTTP/1.1 200 OK (text/html)	
69 35.153456990	192.168.5.215	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=0 Len=0	192.168.5.182	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=0 Len=0
70 35.153458802	192.168.5.215	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=0 Len=0	192.168.5.182	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=0 Len=0
71 35.153458802	192.168.5.215	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=0 Len=0	192.168.5.182	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=0 Len=0
72 35.158196134	192.168.5.182	TCP	68	51798	8080 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	8080 [ACK] Seq=241 Ack=3345 Win=0 Len=0
73 35.158196966	192.168.5.182	TCP	68	51798	8080 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	8080 [ACK] Seq=241 Ack=3345 Win=0 Len=0
74 35.158271022	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
75 35.158276492	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
76 35.158280120	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
77 35.158280120	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
78 35.158280120	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
79 35.158370272	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
80 35.158370272	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
81 35.158959215	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
82 35.158959215	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
83 35.158959215	192.168.5.182	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [ACK] Seq=241 Ack=3345 Win=0 Len=0
84 35.163341146	192.168.5.182	TCP	68	51798	80 [FIN, ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [FIN, ACK] Seq=241 Ack=3345 Win=0 Len=0
85 35.163341146	192.168.5.182	TCP	68	51798	80 [FIN, ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [FIN, ACK] Seq=241 Ack=3345 Win=0 Len=0
86 35.163341146	192.168.5.182	TCP	68	51798	80 [FIN, ACK] Seq=241 Ack=3345 Win=0 Len=0	192.168.5.215	TCP	68	51798	80 [FIN, ACK] Seq=241 Ack=3345 Win=0 Len=0
87 35.163384271	192.168.5.182	TCP	68	80	51798 [ACK] Seq=3345 Ack=241 Win=65024 Len=0	192.168.5.215	TCP	68	80	51798 [ACK] Seq=3345 Ack=241 Win=65024 Len=0
88 35.163384271	192.168.5.182	TCP	68	80	51798 [ACK] Seq=3345 Ack=241 Win=65024 Len=0	192.168.5.215	TCP	68	80	51798 [ACK] Seq=3345 Ack=241 Win=65024 Len=0
89 35.163397157	192.168.5.215	TCP	68	8080	51798 [ACK] Seq=3345 Ack=241 Win=65024 Len=0	192.168.5.182	TCP	68	8080	51798 [ACK] Seq=3345 Ack=241 Win=65024 Len=0
90 35.164042616	2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data		2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data	
91 35.164042616	2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data		2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data	
92 35.164124576	2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data		2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data	
93 35.164124576	2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data		2602:f03f:3:ff03::6a	TLSv1.2	127	Application Data	

Nous pouvons retrouver ce que nous avons vu précédemment avec l'échange de deux machines en utilisant une redirection de port.

80 51798	8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2945963129 TSecr=0 SACK_PERM
80 51798	80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2945963129 TSecr=0 SACK_PERM
80 [TCP Retransmission] 51798	80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2945963129 TSecr=0 SACK_PERM
76 80	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=469563707 TSecr=2945963129 WS=128
76 [TCP Retransmission] 80	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=469563707 TSecr=2945963129 WS=128
76 8080	51798 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=469563707 TSecr=2945963129 WS=128
68 51798	8080 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2945963232 TSecr=469563707
68 51798	80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2945963232 TSecr=469563707

Le 3-Way Handshake ainsi que la redirection de port sont bien en place.

Suite à cela notre client récupère la trame HTTP pour avoir le site internet.

192.168.5.215	192.168.5.182	TCP	1516	8080	51798 [ACK] Seq=1 Ack=241 Win=65024 Len=1448 TSval=469563717 TSecr=2945963236 [TCP segment of a reassembled PDU]
192.168.5.215	192.168.5.182	TCP	1516	8080	51798 [PSH, ACK] Seq=1449 Ack=241 Win=65024 Len=1448 TSval=469563717 TSecr=2945963236 [TCP segment of a reassembled PDU]
172.17.0.2	192.168.5.182	HTTP	515	HTTP/1.1 200 OK (text/html)	
172.17.0.2	192.168.5.182	TCP	515	[TCP Retransmission] 80	51798 [PSH, ACK] Seq=289 Ack=241 Win=65024 Len=447 TSval=469563717 TSecr=2945963236
192.168.5.215	192.168.5.182	HTTP	515	HTTP/1.1 200 OK (text/html)	
172.17.0.2	192.168.5.182	TCP	68	80	51798 [FIN, ACK] Seq=3344 Ack=241 Win=65024 Len=0 TSval=469563717 TSecr=2945963236

# Serveur SSH

## Mise en place du container SSH

Comme pour le container Apache, vous avez la possibilité d'ajouter une redirection de port, attention le service SSH utilise le port 22 par défaut.

```
docker run --name serverssh -ti debian /bin/bash ← Sans redirection de port  
docker run --name serverssh -p 8080:22 -ti debian /bin/bash ← Avec redirection de port
```

## Mise en place du serveur SSH

Nous allons avoir besoin de plusieurs paquets pour l'utilisation de notre serveur.

- ssh paquet contenant notre service ssh
- nano est un éditeur de fichier pour terminaux.

```
apt update  
apt install ssh nano inetutils-ping -y
```

Une fois nos paquets installés, nous allons créer un utilisateur pour nos connexions externes.

```
adduser userssh
```

Suivez les indications, il vous demandera un mot de passe et des informations sur l'utilisateur. N'hésitez pas à rajouter des droits à vos utilisateurs si vous le souhaitez.

Il faut maintenant ajouter cet utilisateur à la liste des utilisateurs autorisés à se connecter au serveur SSH.

Pour ce faire, nous allons modifier le fichier sshd\_config.

```
nano /etc/ssh/sshd_config
```

Vous pouvez rajouter à la fin du fichier la ligne suivante :

```
AllowUsers userssh
```

Vous sauvegardez le fichier avec "CTRL + S" et vous pouvez le fermer avec "CTRL + X".

Nous pouvons enfin lancer notre serveur SSH.

```
service ssh start
```

Si votre serveur est déjà lancé lors de la modification du fichier, n'oubliez pas de le redémarrer pour que les changements s'effectuent.

```
service ssh restart
```

```
root@a3e9751d0b83:/# service ssh start  
Starting OpenBSD Secure Shell server: sshd.
```

Votre serveur SSH est maintenant opérationnel.

## Connexion externe vers le serveur SSH

Pour vous connecter depuis une machine en lan il suffit d'utiliser :

```
ssh userSSH@172.17.0.2
```

Si vous utilisez une machine externe au réseau lan, utilisez l'adresse IP de la machine hôte du container docker. N'oubliez pas d'effectuer la redirection de port.

```
ssh -p 8080 userssh@192.168.5.215
```

Confirmer l'ajout de la clé de sécurité avec **yes**.

```
duschkoll@duschkoll-debian:~$ ssh usersssh@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ED25519 key fingerprint is SHA256:8bd2F8h65drz7uvzael2c9aacXX4PY2+/P5iw9GhVFo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '172.17.0.2' (ED25519) to the list of known hosts.
usersssh@172.17.0.2's password:
Linux a3e9751d0b83 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Vous avez maintenant accès à votre machine debian à travers le SSH

```
usersssh@a3e9751d0b83:~$ ls ../../
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
usersssh@a3e9751d0b83:~$ ls ../
usersssh
```

# SSH - Analyse des trames WireShark

## Analyse de l'établissement d'une connexion SSH sans redirection de port.

5	4.787137623	172.17.0.1	172.17.0.2	TCP	76	55240 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3646336027 TSecr=...
6	4.787150998	172.17.0.1	172.17.0.2	TCP	76	[TCP Retransmission] 55240 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM ...
7	4.787193085	172.17.0.2	172.17.0.1	TCP	76	22 → 55240 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3694...
8	4.787197742	172.17.0.2	172.17.0.1	TCP	76	[TCP Retransmission] 22 → 55240 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460...
9	4.787230457	172.17.0.1	172.17.0.2	TCP	68	55240 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3646336027 TSecr=3694309636
10	4.787233197	172.17.0.1	172.17.0.2	TCP	68	[TCP Dup ACK 9#1] 55240 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3646336027...
11	4.788026176	172.17.0.1	172.17.0.2	SSHv2	108	Client: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u5)
12	4.788034950	172.17.0.1	172.17.0.2	TCP	108	[TCP Retransmission] 55240 → 22 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=40 TSval=3...
13	4.788063224	172.17.0.2	172.17.0.1	TCP	68	22 → 55240 [ACK] Seq=1 Ack=41 Win=65152 Len=0 TSval=3694309637 TSecr=3646336028
14	4.788067165	172.17.0.2	172.17.0.1	TCP	68	[TCP Dup ACK 13#1] 22 → 55240 [ACK] Seq=1 Ack=41 Win=65152 Len=0 TSval=36943096...
15	4.814053179	172.17.0.2	172.17.0.1	SSHv2	108	Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u6)
16	4.814060197	172.17.0.2	172.17.0.1	TCP	108	[TCP Retransmission] 22 → 55240 [PSH, ACK] Seq=1 Ack=41 Win=65152 Len=40 TSval=...
17	4.814117745	172.17.0.1	172.17.0.2	TCP	68	55240 → 22 [ACK] Seq=41 Ack=41 Win=64256 Len=0 TSval=3646336054 TSecr=3694309663
18	4.814121811	172.17.0.1	172.17.0.2	TCP	68	[TCP Dup ACK 17#1] 55240 → 22 [ACK] Seq=41 Ack=41 Win=64256 Len=0 TSval=3646336...
19	4.814598637	172.17.0.1	172.17.0.2	SSHv2	1628	Client: Key Exchange Init
20	4.814609672	172.17.0.1	172.17.0.2	TCP	1628	[TCP Retransmission] 55240 → 22 [PSH, ACK] Seq=41 Ack=41 Win=64256 Len=1560 TSv...
21	4.817123971	172.17.0.2	172.17.0.1	SSHv2	1204	Server: Key Exchange Init
22	4.817130598	172.17.0.2	172.17.0.1	TCP	1204	[TCP Retransmission] 22 → 55240 [PSH, ACK] Seq=41 Ack=1601 Win=64128 Len=1136 T...
23	4.857528702	172.17.0.1	172.17.0.2	TCP	68	55240 → 22 [ACK] Seq=1601 Ack=1177 Win=64128 Len=0 TSval=3646336098 TSecr=36943...
24	4.857569429	172.17.0.1	172.17.0.2	TCP	68	[TCP Dup ACK 23#1] 55240 → 22 [ACK] Seq=1601 Ack=1177 Win=64128 Len=0 TSval=364...
25	4.958422775	172.17.0.1	172.17.0.2	SSHv2	1276	Client: Diffie-Hellman Key Exchange Init
26	4.958432720	172.17.0.1	172.17.0.2	TCP	1276	[TCP Retransmission] 55240 → 22 [PSH, ACK] Seq=1601 Ack=1177 Win=64128 Len=1208...
27	5.001515631	172.17.0.2	172.17.0.1	TCP	68	22 → 55240 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=3694309851 TSecr=36463...
28	5.001523360	172.17.0.2	172.17.0.1	TCP	68	[TCP Dup ACK 27#1] 22 → 55240 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=369...
29	5.007819584	172.17.0.2	172.17.0.1	SSHv2	1632	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=316)
30	5.007827616	172.17.0.2	172.17.0.1	TCP	1632	[TCP Retransmission] 22 → 55240 [PSH, ACK] Seq=1177 Ack=2809 Win=64128 Len=1564...
31	5.007871892	172.17.0.1	172.17.0.2	TCP	68	55240 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=3646336248 TSecr=36943...
32	5.007876852	172.17.0.1	172.17.0.2	TCP	68	[TCP Dup ACK 31#1] 55240 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=364...

Dans cette section de trames WireShark, on peut voir qu'il y a un début de communication TCP entre le client et le serveur SSH, cela se manifeste grâce au paquet SYN, SYN-ACK, ACK sur le port 22. Un 3-Way Handshake est donc présent.

Une fois la connexion établie, les machines échangent la Clé de connexion.

172.17.0.1	172.17.0.2	SSHv2	1628	Client: Key Exchange Init
172.17.0.1	172.17.0.2	TCP	1628	[TCP Retransmission] 55240 → 22 [PSH, ACK] Seq=41 Ack=41 Win=64256 Len=1560 TSval=3646336055 TSecr=3694309663
172.17.0.2	172.17.0.1	SSHv2	1204	Server: Key Exchange Init
172.17.0.2	172.17.0.1	TCP	1204	[TCP Retransmission] 22 → 55240 [PSH, ACK] Seq=41 Ack=1601 Win=64128 Len=1136 TSval=3694309666 TSecr=3646336055
172.17.0.1	172.17.0.2	TCP	68	55240 → 22 [ACK] Seq=1601 Ack=1177 Win=64128 Len=0 TSval=3646336098 TSecr=3694309666
172.17.0.1	172.17.0.2	TCP	68	[TCP Dup ACK 23#1] 55240 → 22 [ACK] Seq=1601 Ack=1177 Win=64128 Len=0 TSval=3646336098 TSecr=3694309666
172.17.0.1	172.17.0.2	SSHv2	1276	Client: Diffie-Hellman Key Exchange Init
172.17.0.1	172.17.0.2	TCP	1276	[TCP Retransmission] 55240 → 22 [PSH, ACK] Seq=1601 Ack=1177 Win=64128 Len=1208 TSval=3646336198 TSecr=3694309666
172.17.0.2	172.17.0.1	TCP	68	22 → 55240 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=3694309851 TSecr=3646336198
172.17.0.2	172.17.0.1	TCP	68	[TCP Dup ACK 27#1] 22 → 55240 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=3694309851 TSecr=3646336198
172.17.0.2	172.17.0.1	SSHv2	1632	Server: Diffie-Hellman Key Exchange Reply, New Keys
172.17.0.2	172.17.0.1	TCP	1632	[TCP Retransmission] 22 → 55240 [PSH, ACK] Seq=1177 Ack=2809 Win=64128 Len=1564 TSval=3694309857 TSecr=3646336198
172.17.0.1	172.17.0.2	TCP	68	55240 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=3646336248 TSecr=3694309857
172.17.0.1	172.17.0.2	TCP	68	[TCP Dup ACK 31#1] 55240 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=3646336248 TSecr=3694309857

Après cet échange, on peut observer la communication entre le client et le serveur.



## Analyse de l'établissement d'une connexion SSH avec redirection de port.

3	4.267023274	192.168.5.215	172.17.0.2	TCP	76 53448 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1859798329 TSecr=0 WS=128
4	4.267052942	192.168.5.215	172.17.0.2	TCP	76 [TCP Retransmission] 53448 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1859798329 TSecr=0 WS=128
5	4.267174046	172.17.0.2	192.168.5.215	TCP	76 22 → 53448 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=4147273467 TSecr=1859798329 WS=128
6	4.267197110	172.17.0.2	192.168.5.215	TCP	76 [TCP Retransmission] 22 → 53448 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=4147273467 TSecr=1859798329 WS=128
7	4.267234151	192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1859798329 TSecr=4147273467
8	4.267295514	192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 7#1] 53448 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1859798329 TSecr=4147273467
9	4.268066896	192.168.5.215	172.17.0.2	SSHv2	168 Client: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u6)
10	4.268081524	192.168.5.215	172.17.0.2	TCP	168 [TCP Retransmission] 53448 → 22 [PSH, ACK] Seq=41 Ack=41 Win=65536 Len=0 TSval=1859798330 TSecr=4147273467
11	4.268094254	172.17.0.2	192.168.5.215	TCP	68 22 → 53448 [ACK] Seq=1 Ack=41 Win=65152 Len=0 TSval=4147273510 TSecr=1859798330
12	4.268095198	172.17.0.2	192.168.5.215	TCP	68 [TCP Dup ACK 11#1] 22 → 53448 [ACK] Seq=1 Ack=41 Win=65152 Len=0 TSval=4147273510 TSecr=1859798330
13	4.310214142	172.17.0.2	192.168.5.215	SSHv2	168 Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u6)
14	4.310235977	172.17.0.2	192.168.5.215	TCP	168 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=41 Ack=41 Win=65536 Len=1568 TSval=1859798373 TSecr=4147273510
15	4.311366395	192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=41 Ack=1 Win=65536 Len=0 TSval=1859798372 TSecr=4147273510
16	4.310235538	192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 15#1] 53448 → 22 [ACK] Seq=41 Ack=1 Win=65536 Len=0 TSval=1859798372 TSecr=4147273510
17	4.311386335	192.168.5.215	172.17.0.2	SSHv2	1628 Client: Key Exchange Init
18	4.311386335	192.168.5.215	172.17.0.2	TCP	1628 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=41 Ack=41 Win=65536 Len=1568 TSval=1859798373 TSecr=4147273510
19	4.311435854	172.17.0.2	192.168.5.215	TCP	68 22 → 53448 [ACK] Seq=41 Ack=1601 Win=64128 Len=0 TSval=4147273511 TSecr=1859798373
20	4.311439958	172.17.0.2	192.168.5.215	TCP	68 [TCP Dup ACK 19#1] 22 → 53448 [ACK] Seq=41 Ack=1601 Win=64128 Len=0 TSval=4147273511 TSecr=1859798373
21	4.314027361	172.17.0.2	192.168.5.215	SSHv2	1204 Server: Key Exchange Init
22	4.314027361	172.17.0.2	192.168.5.215	TCP	68 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=41 Ack=1601 Win=64128 Len=1136 TSval=4147273514 TSecr=1859798373
23	4.355989888	192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=1601 Ack=1177 Win=65536 Len=0 TSval=1859798418 TSecr=4147273514
24	4.355994167	192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 23#1] 53448 → 22 [ACK] Seq=1601 Ack=1177 Win=65536 Len=0 TSval=1859798418 TSecr=4147273514
25	4.481671750	192.168.5.215	172.17.0.2	SSHv2	1276 Client: Diffie-Hellman Key Exchange Init
26	4.481681742	192.168.5.215	172.17.0.2	TCP	1276 [TCP Retransmission] 53448 → 22 [PSH, ACK] Seq=1601 Ack=1177 Win=65536 Len=0 TSval=1859798543 TSecr=4147273514
27	4.524056579	172.17.0.2	192.168.5.215	TCP	68 22 → 53448 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=4147273724 TSecr=1859798543
28	4.524056579	172.17.0.2	192.168.5.215	TCP	68 [TCP Dup ACK 27#1] 22 → 53448 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=4147273724 TSecr=1859798543
29	4.530614332	172.17.0.2	192.168.5.215	SSHv2	1632 Server: Diffie-Hellman Key Exchange Reply, New Keys
30	4.530614332	172.17.0.2	192.168.5.215	TCP	1632 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=1177 Ack=2809 Win=64128 Len=1564 TSval=4147273730 TSecr=1859798543
31	4.530675584	192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=1859798592 TSecr=4147273730
32	4.530681489	192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 31#1] 53448 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=1859798592 TSecr=4147273730

Dans cette section de trames WireShark, nous observons les différentes étapes observées lors de l'établissement d'une connexion SSH entre 2 machines (192.168.5.215 et 172.17.0.2).

Bien que nous utilisons la redirection de port pour effectuer notre connexion SSH, celle-ci n'est pas visible étant donné que nous nous connectons depuis la machine qui host le container Docker ayant le service SSH.

Nous pouvons retrouver tout d'abord la connexion entre les 2 machines, le client envoie un paquet SYN pour initier la connexion sur le port 22, par la suite on peut voir que le serveur répond avec un SYN-ACK en acceptant la connexion et enfin on peut noter que le client renvoie un ACK finalisant le 3-Way Handshake.

76	53448 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1859798329 TSecr=0 WS=128
76	[TCP Retransmission] 53448 → 22 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1859798329 TSecr=0 WS=128
76	22 → 53448 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=4147273467 TSecr=1859798329 WS=128
76	[TCP Retransmission] 22 → 53448 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=4147273467 TSecr=1859798329 WS=128
68	53448 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1859798329 TSecr=4147273467
68	[TCP Dup ACK 7#1] 53448 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1859798329 TSecr=4147273467

On peut retrouver l'échange de clé SSH pour autoriser la connexion entre les deux machines.

172.17.0.2	192.168.5.215	TCP	108 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=1 Ack=41 Win=65152 Len=0 TSval=4147273510 TSecr=1859798330
192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=41 Ack=41 Win=65536 Len=0 TSval=1859798372 TSecr=4147273510
192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 15#1] 53448 → 22 [ACK] Seq=41 Ack=41 Win=65536 Len=0 TSval=1859798372 TSecr=4147273510
192.168.5.215	172.17.0.2	SSHv2	1628 Client: Key Exchange Init
192.168.5.215	172.17.0.2	TCP	1628 [TCP Retransmission] 53448 → 22 [PSH, ACK] Seq=41 Ack=41 Win=65536 Len=1568 TSval=1859798373 TSecr=4147273510
172.17.0.2	192.168.5.215	TCP	68 22 → 53448 [ACK] Seq=41 Ack=1601 Win=64128 Len=0 TSval=4147273511 TSecr=1859798373
172.17.0.2	192.168.5.215	TCP	68 [TCP Dup ACK 19#1] 22 → 53448 [ACK] Seq=41 Ack=1601 Win=64128 Len=0 TSval=4147273511 TSecr=1859798373
172.17.0.2	192.168.5.215	SSHv2	1204 Server: Key Exchange Init
172.17.0.2	192.168.5.215	TCP	1204 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=41 Ack=1601 Win=64128 Len=1136 TSval=4147273514 TSecr=1859798373
192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=1601 Ack=1177 Win=65536 Len=0 TSval=1859798418 TSecr=4147273514
192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 23#1] 53448 → 22 [ACK] Seq=1601 Ack=1177 Win=65536 Len=0 TSval=1859798418 TSecr=4147273514
192.168.5.215	172.17.0.2	SSHv2	1276 Client: Diffie-Hellman Key Exchange Init
192.168.5.215	172.17.0.2	TCP	1276 [TCP Retransmission] 53448 → 22 [PSH, ACK] Seq=1601 Ack=1177 Win=65536 Len=1208 TSval=1859798543 TSecr=4147273514
172.17.0.2	192.168.5.215	TCP	68 22 → 53448 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=4147273724 TSecr=1859798543
172.17.0.2	192.168.5.215	TCP	68 [TCP Dup ACK 27#1] 22 → 53448 [ACK] Seq=1177 Ack=2809 Win=64128 Len=0 TSval=4147273724 TSecr=1859798543
172.17.0.2	192.168.5.215	SSHv2	1632 Server: Diffie-Hellman Key Exchange Reply, New Keys
192.168.5.215	172.17.0.2	TCP	1632 [TCP Retransmission] 22 → 53448 [PSH, ACK] Seq=1177 Ack=2809 Win=64128 Len=1564 TSval=4147273730 TSecr=1859798543
192.168.5.215	172.17.0.2	TCP	68 53448 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=1859798592 TSecr=4147273730
192.168.5.215	172.17.0.2	TCP	68 [TCP Dup ACK 31#1] 53448 → 22 [ACK] Seq=2809 Ack=2741 Win=64128 Len=0 TSval=1859798592 TSecr=4147273730
192.168.5.215	172.17.0.2	SSHv2	84 Client: New Keys
192.168.5.215	172.17.0.2	TCP	84 [TCP Retransmission] 53448 → 22 [PSH, ACK] Seq=2809 Ack=2741 Win=65536 Len=16 TSval=1859798641 TSecr=4147273730
172.17.0.2	192.168.5.215	TCP	68 22 → 53448 [ACK] Seq=2741 Ack=2825 Win=64128 Len=0 TSval=4147273779 TSecr=1859798641
172.17.0.2	192.168.5.215	TCP	68 [TCP Dup ACK 35#1] 22 → 53448 [ACK] Seq=2741 Ack=2825 Win=64128 Len=0 TSval=4147273779 TSecr=1859798641



## Analyse d'établissement d'une connexion SSH entre deux réseaux.

2	0.000055766	AzureWaveTec_99:19...	ARP	44	192.168.5.215 is at dc:f5:05:99:19:4f
3	0.007690792	192.168.5.92	192.168.5.215	TCP	68 57554 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
4	0.007752630	192.168.5.92	172.17.0.2	TCP	68 57554 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
5	0.007759325	192.168.5.92	172.17.0.2	TCP	68 [TCP Retransmission] 57554 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
6	0.007815750	172.17.0.2	192.168.5.92	TCP	68 22 → 57554 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.007818804	172.17.0.2	192.168.5.92	TCP	68 [TCP Retransmission] 22 → 57554 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
8	0.007830039	192.168.5.215	192.168.5.92	TCP	68 8080 → 57554 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
9	0.019604950	192.168.5.92	192.168.5.215	TCP	56 57554 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
10	0.019655954	192.168.5.92	172.17.0.2	TCP	56 57554 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0
11	0.019662162	192.168.5.92	172.17.0.2	TCP	56 [TCP Dup ACK 10#1] 57554 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0
12	0.050590543	172.17.0.2	192.168.5.92	SSHv2	96 Server: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u6)
13	0.050597234	172.17.0.2	192.168.5.92	TCP	96 [TCP Retransmission] 22 → 57554 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=40
14	0.050635396	192.168.5.215	192.168.5.92	TCP	96 8080 → 57554 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=40
15	0.052089250	192.168.5.92	192.168.5.215	TCP	96 57554 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=40
16	0.052132200	192.168.5.92	172.17.0.2	SSHv2	96 Client: Protocol (SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u6)
17	0.052139297	192.168.5.92	172.17.0.2	TCP	96 [TCP Retransmission] 57554 → 22 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=40
18	0.052206655	172.17.0.2	192.168.5.92	TCP	56 22 → 57554 [ACK] Seq=41 Ack=41 Win=64256 Len=0
19	0.052209514	172.17.0.2	192.168.5.92	TCP	56 [TCP Dup ACK 18#1] 22 → 57554 [ACK] Seq=41 Ack=41 Win=64256 Len=0
20	0.052222065	192.168.5.215	192.168.5.92	TCP	56 8080 → 57554 [ACK] Seq=41 Ack=41 Win=64256 Len=0
21	0.210980998	192.168.5.92	192.168.5.215	TCP	1616 57554 → 8080 [PSH, ACK] Seq=41 Ack=41 Win=65536 Len=1560
22	0.211019736	192.168.5.92	172.17.0.2	SSHv2	1616 Client: Key Exchange Init
23	0.211025847	192.168.5.92	172.17.0.2	TCP	1616 [TCP Retransmission] 57554 → 22 [PSH, ACK] Seq=41 Ack=41 Win=65536 Len=1560
24	0.211090910	172.17.0.2	192.168.5.92	SSHv2	1192 Server: Key Exchange Init
25	0.211093878	172.17.0.2	192.168.5.92	TCP	1192 [TCP Retransmission] 22 → 57554 [PSH, ACK] Seq=41 Ack=1601 Win=63744 Len=1136
26	0.211141762	192.168.5.215	192.168.5.92	TCP	1192 8080 → 57554 [PSH, ACK] Seq=41 Ack=1601 Win=63744 Len=1136
27	0.413179846	192.168.5.92	192.168.5.215	TCP	56 57554 → 8080 [ACK] Seq=1601 Ack=1177 Win=64512 Len=0
28	0.413221091	192.168.5.92	172.17.0.2	TCP	56 57554 → 22 [ACK] Seq=1601 Ack=1177 Win=64512 Len=0

Dans cette section de trames WireShark, on peut observer une tentative de connexion entre un client local (192.168.5.92) et un serveur distant (172.17.0.2) avec une redirection de port sur une machine intermédiaire (192.168.5.215) via le port 8080.

3	0.007690792	192.168.5.92	192.168.5.215	TCP	68 57554 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
4	0.007752630	192.168.5.92	172.17.0.2	TCP	68 57554 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
5	0.007759325	192.168.5.92	172.17.0.2	TCP	68 [TCP Retransmission] 57554 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
6	0.007815750	172.17.0.2	192.168.5.92	TCP	68 22 → 57554 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
7	0.007830039	192.168.5.215	192.168.5.92	TCP	68 8080 → 57554 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
8	0.019604950	192.168.5.92	192.168.5.215	TCP	56 57554 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
9	0.019655954	192.168.5.92	172.17.0.2	TCP	56 57554 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0
10	0.019662162	192.168.5.92	172.17.0.2	TCP	56 [TCP Dup ACK 10#1] 57554 → 22 [ACK] Seq=1 Ack=1 Win=65536 Len=0

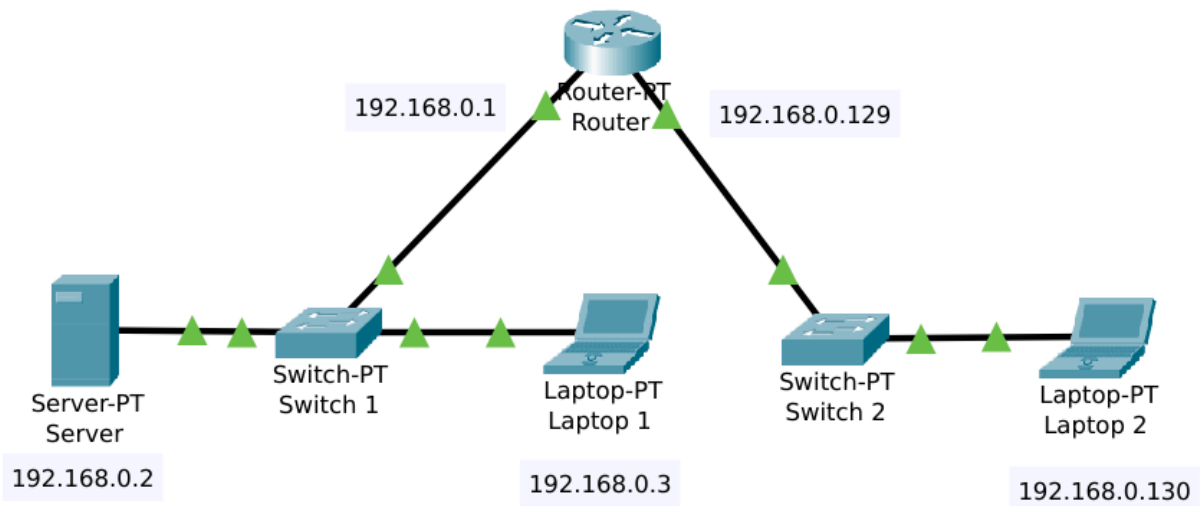
Au début, on peut observer une connexion TCP sur le port redirigé, avec l'enchaînement SYN, SYN-ACK, ACK.

Par la suite, on peut voir un échange des versions de SSH des deux machines. Et on peut observer qu'à chaque échange entre le client et le serveur, les paquets passent obligatoirement par la machine intermédiaire : elle joue le rôle de pont entre deux réseaux différents. Le client pense communiquer directement avec la machine intermédiaire, mais en réalité celle-ci transmet tout au serveur SSH.

## Configuration théorique du réseau

### Topologie du réseau

Afin de pouvoir faire communiquer nos machines entre les réseaux, nous avons décidé de mettre deux switch reliés à un routeur. Ainsi nous pouvons brancher nos machines sur nos switchs en fonction des besoins.



## Plan d'adressage

Pour nos tests nous avons besoin que de deux réseau, c'est pour celà que nous utilisons un masque de réseau /25 (255.255.255.128).

On pourra, au maximum, connecter 125 machines par réseau vu que des IP sont réservées pour l'adresse réseau, le broadcast et qu'une adresse est donnée pour le routeur.

Routeur ↔ Switch n°1	192.168.0.1 /25
PC Server	192.168.0.2 /25
PC Client n°1	192.168.0.3 /25
Routeur ↔ Switch n° 2	192.168.0.129 /25
PC Client n°2	192.168.0.130 /25

# Docker Compose

Pour établir nos containers, nous allons créer une image de notre serveur contenant Apache.

```
docker commit server server_setup_sae203
```

Nous allons mettre en place notre compose.yaml afin de créer nos 3 serveurs.

Pour ceci, créer un dossier.

A l'intérieur de celui-ci veuillez créer le fichier compose.yaml, dans celui-ci veuillez entrer les lignes suivantes :

services:

```
sae_server1:
  image: server_setup_sae203
  tty: true
  ports:
    - "8001:80"
```

```
sae_server2:
  image: server_setup_sae203
  tty: true
  ports:
    - "8002:80"
```

```
sae_server3:
  image: server_setup_sae203
  tty: true
  ports:
    - "8003:80"
```

tty:true permet que vos containers une fois mise en place, reste en fonctionnement.

Ouvrez un terminal dans le dossier où vous avez créé le fichier compose.yaml.

Entrer la commande suivante pour créer vos containers et les lancer.

```
docker compose up
```

Pour montrer que nos serveur sont bien différents de chacun, nous allons remplacer les fichiers index.html de chaque serveur par un autre index customisé

```
docker compose cp index1.html sae_server1:/var/www/html/index.html
docker compose cp index2.html sae_server2:/var/www/html/index.html
docker compose cp index3.html sae_server3:/var/www/html/index.html
```

Il faut maintenant démarrer le service apache sur nos trois containers.  
Répéter cette action 3 fois en changeant le nom du container.

```
docker compose exec sae_server1 bash
service apache2 start
exit
```

Il vous suffit maintenant d'utiliser votre navigateur pour accéder à vos serveurs.



Serveur Number 1



Serveur Number 2



Serveur Number 3

## Conclusion

Durant ce projet, nous avons pu appliquer les notions vues en R2.05 tels que l'adressage IP, le fonctionnement de sous-réseaux.

Ainsi nous avons pu mettre en place un serveur web Apache dans un conteneur Docker, accessible depuis différentes machines sur des réseaux distincts.

Grâce à la mise en place du service SSH, nous avons pu accéder de façon sécurisée à nos containers.

De plus, l'étude des paquets réseau avec Wireshark a fourni une compréhension détaillée des communications, ainsi que des effets liés aux modifications de ports.

Nous vous remercions pour le temps que vous nous avez accordé en lisant ce compte rendu,

COLLOT Grégoire,  
MARECAILLE-HAINAUT Mattieu,  
SIMSEK Dilara.