

SAE 2.03: INSTALLATION DE SERVICE RÉSEAUX

Réalisé par : TRAN Minh-Tue

Membres: TRAN Minh-Tue

SAINZ Miquel

SIDHOU Sidali

Partie 1: Création du Serveur Web / SSH avec Docker	3
Configuration Serveur Web /SSH	3
Installation de Docker	3
Mise en place d'un conteneur Debian	3
Installation Apache dans le container	3
Mise en place SSH	4
Capture de trames avec l'outil Wireshark	6
HTTP	6
Commentaire des trames HTTP	6
Détails supplémentaires de la requête HTTP	7
En résumé	9
SSH	10
Commentaire des trames SSH	10
En résumé	11
Modifications dans les trames HTTP	12
Partie 2: Configuration théorique du réseau	16
Configuration d'un réseau sur le Docker	16
Option	19
Récapitulatif des commandes	21
Partie 3: Configuration pratique du réseaux et déploiement du serveur web sur le réseau	22
TRAMES HTTP CAPTURÉES - client du client-network	29
Commentaires de trames	29
Trame 8 : Requête HTTP GET	29
Trame 12 : Réponse HTTP 200 OK	30
Conclusion pour trames capture client	30
TRAMES HTTP CAPTURÉES - clientwn du réseau server-network	31
Commentaires de trames:	32
Trame 9 : Requête HTTP GET	32
Trame 13 : Réponse HTTP 200 OK	33
Différences principales	33
3 Serveurs web sur 3 Containers docker	34
Démonstration avec mon réseau	35

Partie 1: Création du Serveur Web / SSH avec Docker

Configuration Serveur Web /SSH

Installation de Docker

- Installer le docker desktop via le [lien](#)
- Vérifier l'installation et la version de Docker via le terminal : *docker --version*

Mise en place d'un conteneur Debian

- Démarrer un conteneur Debian avec le port 80 mappé au port 8080 de votre machine hôte via la commande docker suivante:

```
docker run -dit --name webserver -p 8080:80 -p 2222:22 debian /bin/bash
```

Explication:

- -dit : Démarre le container en mode détaché (background) et interactif.
- --name webserver : Nom du container.
- -p 8080:80 : Mappe le port 8080 de la machine hôte au port 80 du container.
- -p 2222:22 : Mappe le port 2222 de la machine hôte au port 22 du container (pour SSH).

/bin/bash est nécessaire pour spécifier le shell à exécuter dans le container. Cela vous donne accès à une interface de ligne de commande interactive. Sans cela, le container pourrait démarrer et se terminer immédiatement car il n'aurait pas de processus actif pour le maintenir en cours d'exécution.

Installation Apache dans le container

- Exécuter le serveur :

```
docker exec -it webserver bash
```

- Mettre à jour les paquets et installer Apache :

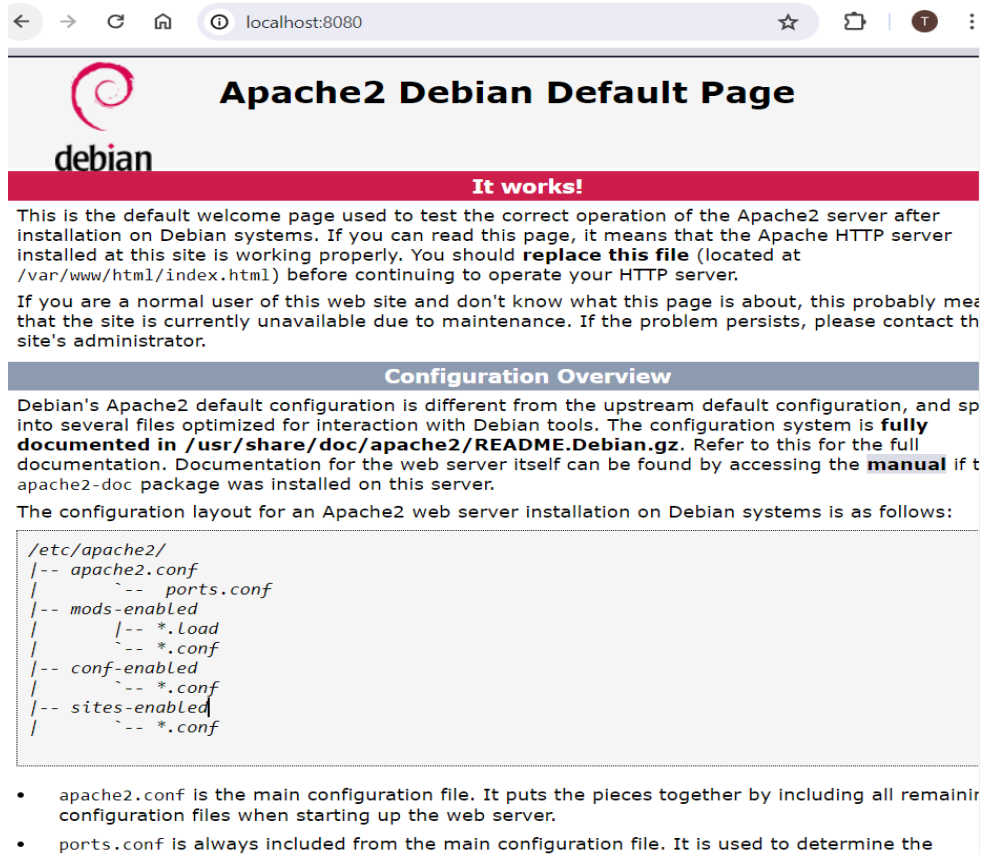
```
apt-get update
```

```
apt-get install apache2 -y
```

- Démarrer Apache

```
service apache2 start
```

Tester le serveur en accédant à localhost:8080 dans le navigateur.



Mise en place SSH

- Dans le même container, installer OpenSSH via la commande suivante :

```
apt-get install openssh-server -y
```

- Démarrer le SSH

```
service ssh start
```

- Configurer le mot de passe root :

```
echo 'root:root123' | chpasswd
```

- Modifier la configuration SSH pour permettre la connexion root :

```
sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/'  
/etc/ssh/sshd_config
```

- Démarrer et vérifier le service SSH:

```
service ssh start
```

```
service ssh status
```

p/s: Pour sortir, on peut utiliser *exit*. Et se connecter en utilisant *ssh root@localhost -p 2222*

```
C:\Users\Admin>ssh root@localhost -p 2222  
The authenticity of host '[localhost]:2222 (:::1):2222' can't be established.  
ECDSA key fingerprint is SHA256:A8E8G29xtLt08XL/QuitkoJ3RXt4HyTiXGa+MU2CtaMk.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '[localhost]:2222' (ECDSA) to the list of known hosts.  
root@localhost's password:  
Linux 81c11be04b64 5.15.146.1-microsoft-standard-WSL2 #1 SMP Thu Jan 11 04:09:03 UTC 2024 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.  
root@81c11be04b64:~# service apache2 status  
apache2 is running.  
root@81c11be04b64:~#
```

Capture de trames avec l'outil Wireshark

- Ouvrez l'outil Wireshark
- Accéder <http://localhost:8080/> (ce qui a été créé par le Docker)

HTTP

Dans l'outil Wireshark, je crée un filtre HTTP. Je trouve une liste des trames concernant le port que j'ai créé via Docker.

The screenshot shows the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. The main display area is divided into three panes. The top pane shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. Two packets are visible, both HTTP GET requests to localhost:8080. The bottom pane shows the details of the selected packet (Frame 472), including the Ethernet II header, Internet Protocol Version 6 header, and the Hypertext Transfer Protocol section. The HTTP section shows a GET request to http://localhost:8080/ with various headers like Host, Connection, Cache-Control, and User-Agent.

No.	Time	Source	Destination	Protocol	Length	Info
472	4.802326	:::1	:::1	HTTP	888	GET / HTTP/1.1
477	4.809723	:::1	:::1	HTTP	3444	HTTP/1.1 200 OK (text/html)

Frame 472: 888 bytes on wire (7104 bits), 888 bytes captured (7104 bits) on interface \Device\NPF_{Loopback}, id 0

Null/Loopback

Internet Protocol Version 6, Src: ::1, Dst: ::1

Transmission Control Protocol, Src Port: 55433, Dst Port: 8080, Seq: 1, Ack: 1, Len: 824

Hypertext Transfer Protocol

> GET / HTTP/1.1\r\n

Host: localhost:8080\r\n

Connection: keep-alive\r\n

Cache-Control: max-age=0\r\n

sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not.A/Brand";v="24"\r\n

sec-ch-ua-mobile: ?0\r\n

sec-ch-ua-platform: "Windows"\r\n

Upgrade-Insecure-Requests: 1\r\n

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n

Sec-Fetch-Site: none\r\n

Sec-Fetch-Mode: navigate\r\n

Sec-Fetch-User: ?1\r\n

Sec-Fetch-Dest: document\r\n

Accept-Encoding: gzip, deflate, br, zstd\r\n

Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5\r\n

If-None-Match: "29cd-61add068daf2-gzip"\r\n

If-Modified-Since: Fri, 14 Jun 2024 17:32:37 GMT\r\n

\r\n

[Full request URI: http://localhost:8080/]

[HTTP request 1/1]

[Response in frame: 477]

Commentaire des trames HTTP

1. **Frame 472:** 888 bytes on wire (7104 bits), 888 bytes captured (7104 bits) on interface \Device\NPF_{Loopback}, id 0
 - o **Frame 472:** Indique le numéro de la trame capturée.
 - o **888 bytes on wire (7104 bits):** La taille de la trame sur le réseau est de 888 octets, ce qui correspond à 7104 bits.
 - o **888 bytes captured (7104 bits):** La taille de la trame capturée par Wireshark est également de 888 octets, ce qui correspond à 7104 bits.

- *on interface \Device\NPF_Loopback, id 0*: La trame a été capturée sur l'interface de bouclage (loopback), avec l'identifiant 0.
- 2. *Null/Loopback*
 - *Null/Loopback*: Indique que la trame est capturée sur une interface de bouclage. Cela signifie que le trafic est généré et consommé localement sur la même machine, sans quitter l'interface réseau locale.
- 3. *Internet Protocol Version 6, Src: ::1, Dst: ::1*
 - *Internet Protocol Version 6*: La trame utilise IPv6 pour la communication réseau.
 - *Src: ::1*: L'adresse source est ::1, qui est l'adresse de bouclage IPv6. Cela signifie que la source est la machine locale.
 - *Dst: ::1*: L'adresse de destination est également ::1, indiquant que la destination est la machine locale.
- 4. *Transmission Control Protocol, Src Port: 55433, Dst Port: 8080, Seq: 1, Ack: 1, Len: 824*
 - *Transmission Control Protocol*: La trame utilise TCP comme protocole de transport.
 - *Src Port: 55433*: Le port source est 55433. Ce port est utilisé par l'application cliente pour envoyer des données.
 - *Dst Port: 8080*: Le port de destination est 8080. Ce port est utilisé par le serveur web pour recevoir des données HTTP.
 - *Seq: 1, Ack: 1*: Le numéro de séquence (Seq) est 1, et le numéro d'acquittement (Ack) est également 1. Cela fait partie de l'établissement de la connexion TCP.
 - *Len: 824*: La longueur des données utiles dans cette trame est de 824 octets.
- 5. *Hypertext Transfer Protocol*
 - *Hypertext Transfer Protocol*: La trame contient des données HTTP.

Détails supplémentaires de la requête HTTP

1. GET / HTTP/1.1
 - *GET / HTTP/1.1*: Requête HTTP GET pour la ressource racine / en utilisant la version HTTP/1.1.
2. Host: localhost:8080
 - *Host: localhost:8080*: Champ d'en-tête HTTP spécifiant l'adresse de destination et le port (localhost sur le port 8080).
3. Connection: keep-alive
 - *Connection: keep-alive*: Indique que la connexion doit être maintenue ouverte pour des requêtes futures, permettant une meilleure performance en évitant de rouvrir la connexion TCP.
4. Cache-Control: max-age=0
 - *Cache-Control: max-age=0*: Indique que la réponse ne doit pas être mise en cache et doit être revalidée à chaque requête.
5. sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not.A/Brand";v="24"
 - *sec-ch-ua*: En-tête de sécurité utilisé pour spécifier le navigateur et sa version.
6. sec-ch-ua-mobile: ?0
 - *sec-ch-ua-mobile: ?0*: Indique si l'utilisateur utilise un appareil mobile (0 pour non).

7. `sec-ch-ua-platform: "Windows"`
 - `sec-ch-ua-platform: "Windows"` : Indique la plateforme utilisée, ici Windows.
8. `Upgrade-Insecure-Requests: 1`
 - `Upgrade-Insecure-Requests: 1` : Indique que le client souhaite que le serveur serve le contenu via une connexion sécurisée si possible.
9. `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36`
 - `User-Agent` : Spécifie le navigateur client et ses détails (ici Google Chrome sur Windows 10).
10. `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,/;q=0.8,application/signed-exchange;v=b3;q=0.7`
 - `Accept` : Spécifie les types de contenu que le client peut traiter, avec leurs priorités.
11. `Sec-Fetch-Site: none`
 - `Sec-Fetch-Site: none` : Indique l'origine de la requête (none signifie que la requête n'a pas de site référent).
12. `Sec-Fetch-Mode: navigate`
 - `Sec-Fetch-Mode: navigate` : Indique que la requête est utilisée pour la navigation.
13. `Sec-Fetch-User: ?1`
 - `Sec-Fetch-User: ?1` : Indique que l'utilisateur a déclenché la navigation (1 pour oui).
14. `Sec-Fetch-Dest: document`
 - `Sec-Fetch-Dest: document` : Indique que la requête est destinée à obtenir un document.
15. `Accept-Encoding: gzip, deflate, br, zstd`
 - `Accept-Encoding` : Spécifie les méthodes de compression que le client peut traiter.
16. `Accept-Language: vi-VN;vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5`
 - `Accept-Language` : Spécifie les langues préférées du client pour la réponse.
17. `If-Modified-Since: Fri, 14 Jun 2024 17:32:37 GMT`
 - `If-Modified-Since` : Indique que le client souhaite recevoir la ressource uniquement si elle a été modifiée depuis la date spécifiée.

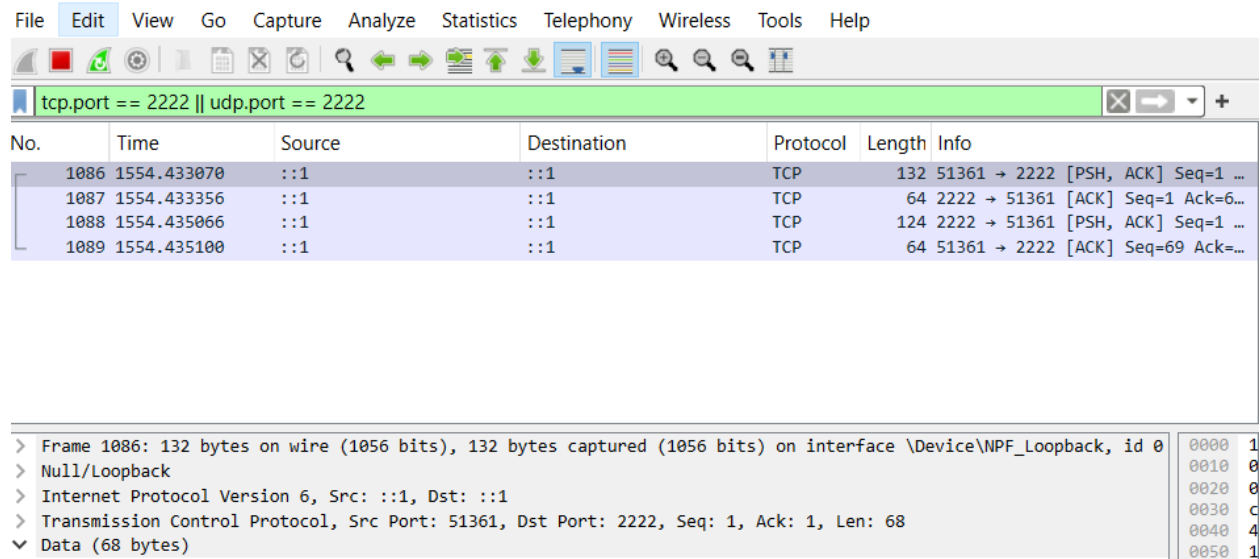
En résumé

- **If-Modified-Since** : Champ d'en-tête HTTP fournissant des informations sur la requête et les capacités du client.
- **Frame 472** : Numéro de trame et taille des données capturées.
- **Null/Loopback** : Indique que la trame est capturée sur l'interface de boucle locale.
- **Internet Protocol Version 6** : Utilisation d'IPv6 avec l'adresse de bouclage `::1` pour source et destination.
- **Transmission Control Protocol** : Communication via TCP, avec détails sur les ports, numéros de séquence et d'acquittement.
- **Hypertext Transfer Protocol** : Indique que la trame contient des données HTTP.
- **GET / HTTP/1.1** : Requête HTTP GET pour la ressource racine `/`.
- **Host** : Spécifie l'adresse de destination.
- **Connection, Cache-Control, sec-ch-ua, sec-ch-ua-mobile, sec-ch-ua-platform, Upgrade-Insecure-Requests, User-Agent, Accept, Sec-Fetch-Site, Sec-Fetch-Mode, Sec-Fetch-User, Sec-Fetch-Dest, Accept-Encoding, Accept-Language,**

SSH

- Chercher ssh via la commande : `tcp.port == 2222 || udp.port == 2222`

p/s: SSH devrait être en cours d'exécution pour avoir une liste des trames sur Wireshark



Commentaire des trames SSH

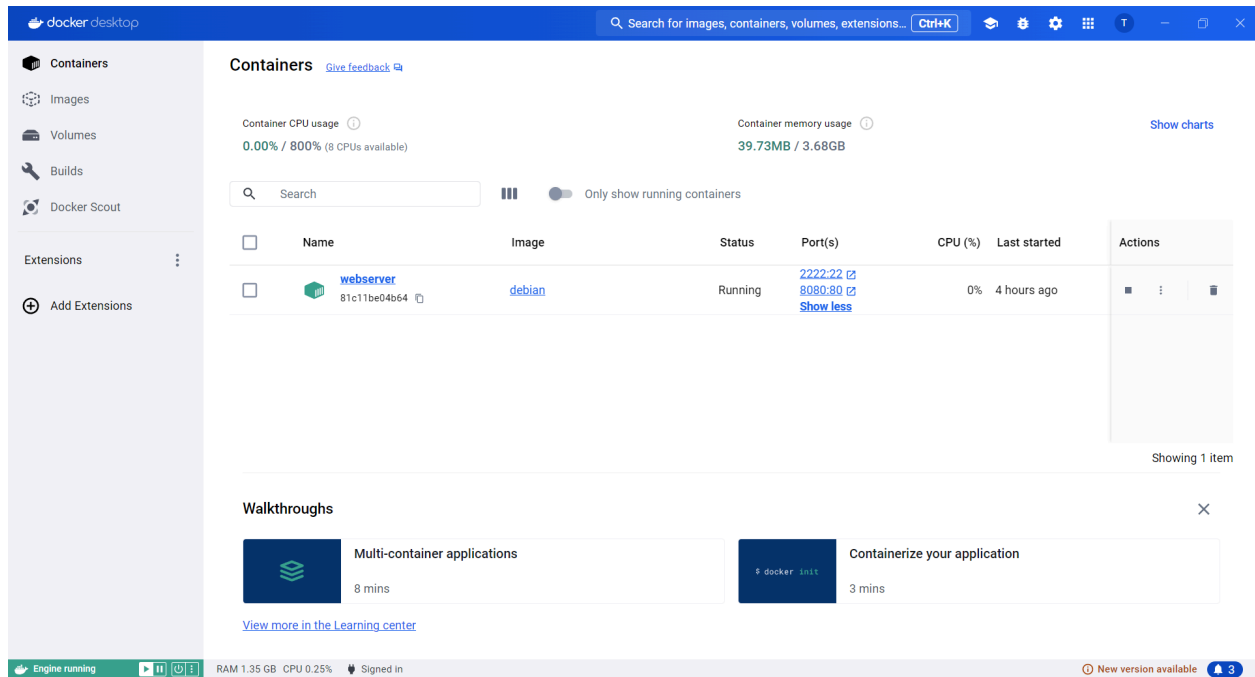
1. **Frame 1086:** 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface \Device\NPF_Loopback, id 0
 - o **Frame 1086 :** Il s'agit de la 1086e trame capturée dans cette session Wireshark.
 - o **132 bytes on wire (1056 bits) :** La taille de la trame sur le réseau est de 132 octets, ce qui correspond à 1056 bits.
 - o **132 bytes captured (1056 bits) :** La taille de la trame capturée par Wireshark est également de 132 octets, ce qui correspond à 1056 bits.
 - o **on interface \Device\NPF_Loopback, id 0 :** La trame a été capturée sur l'interface de bouclage (loopback), avec l'identifiant 0.
2. **Null/Loopback**
 - o Indique que la trame est capturée sur une interface de bouclage. Cela signifie que le trafic est généré et consommé localement sur la même machine, sans quitter l'interface réseau locale.
3. **Internet Protocol Version 6, Src: ::1, Dst: ::1**
 - o **Internet Protocol Version 6 :** La trame utilise IPv6 pour la communication réseau.
 - o **Src: ::1 :** L'adresse source est ::1, qui est l'adresse de bouclage IPv6. Cela signifie que la source est la machine locale.
 - o **Dst: ::1 :** L'adresse de destination est également ::1, indiquant que la destination est la machine locale.
4. **Transmission Control Protocol, Src Port: 51361, Dst Port: 2222, Seq: 1, Ack: 1, Len: 68**
 - o **Transmission Control Protocol :** La trame utilise TCP comme protocole de transport.

- *Src Port: 51361* : Le port source est 51361. Ce port est utilisé par l'application cliente pour envoyer des données.
 - *Dst Port: 2222* : Le port de destination est 2222. Ce port est utilisé par le serveur SSH pour recevoir des données.
 - *Seq: 1, Ack: 1* : Le numéro de séquence (Seq) est 1, et le numéro d'acquittement (Ack) est également 1. Cela fait partie de l'établissement de la connexion TCP.
 - *Len: 68* : La longueur des données utiles dans cette trame est de 68 octets.
5. *Data (68 bytes)*
- *Data (68 bytes)* : Cette section montre les données utiles envoyées dans cette trame. Les données sont chiffrées car il s'agit d'une connexion SSH sécurisée.

En résumé

- **Frame 1086** : Numéro de trame et taille des données capturées.
- **Null/Loopback** : Indique que la trame est capturée sur l'interface de bouclage locale.
- **Internet Protocol Version 6** : Utilisation d'IPv6 avec l'adresse de bouclage `::1` pour source et destination.
- **Transmission Control Protocol** : Communication via TCP, avec détails sur les ports, numéros de séquence et d'acquittement.
- **Data (68 bytes)** : Données chiffrées envoyées via la connexion SSH.
- **Stream index, TCP Segment Length, Sequence Number, Acknowledgment Number, Flags, Window, Checksum, Urgent Pointer** : Détails supplémentaires sur le segment TCP.

Modifications dans les trames HTTP



État du conteneur avant de la modification

Changement du port de conteneur 80 au 8080

- Accéder à votre container via la command :

`docker exec -it webserver bash`

- Modifier les fichiers de configuration d'Apache dans votre container avec ces commandes :

`sed -i 's/Listen 80/Listen 8080/' /etc/apache2/ports.conf`

*`sed -i 's/<VirtualHost *:80>/<VirtualHost *:8080>/' /etc/apache2/sites-available/000-default.conf`*

- Vérifier la configuration

cat /etc/apache2/ports.conf

Dans cet exemple, Apache est configuré pour écouter sur le port 8080 pour les connexions HTTP

```
</IfModule>
root@81c11be04b64:/# cat /etc/apache2/ports.conf
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8080

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
root@81c11be04b64:/# cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:8080>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

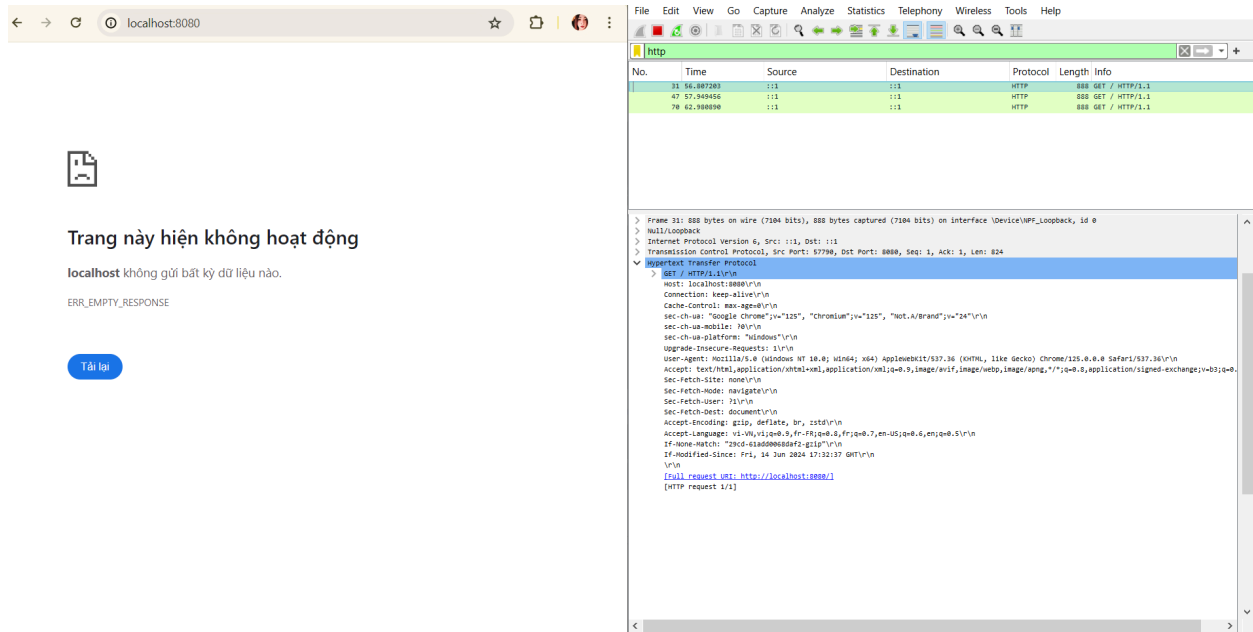
    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Ensuite, on redémarre le serveur avec command

service apache2 restart

Sur le navigateur, on va sur <http://localhost:8080>. Après la modification du port, je recharge le localhost:8080 avec le conteneur 8080:8080.

Le site ne fonctionne plus.



The screenshot shows a web browser window with the address bar set to `localhost:8080`. The page content displays a 404 error: "Trang này hiện không hoạt động" (This page is not working) and "localhost không gửi bất kỳ dữ liệu nào." (localhost did not send any data). Below the error message is a blue button labeled "Tải lại" (Reload).

To the right of the browser window is a Wireshark packet capture window. The top pane shows a list of captured packets, with the selected packet being an HTTP GET request to `http://localhost:8080/`. The bottom pane shows the details of this packet, including the HTTP request line `GET / HTTP/1.1` and various headers such as `Host: localhost:8080`, `Connection: keep-alive`, `Cache-Control: max-age=0`, `sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not.A.Brand";v="24"`, `sec-ch-ua-mobile: ?0`, `sec-ch-ua-platform: "Windows"`, `Upgrade-Insecure-Requests: 1`, `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.6.0 Safari/537.36`, `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.0`, `Sec-fetch-site: none`, `Sec-fetch-mode: navigate`, `Sec-fetch-user: ?1`, `Sec-fetch-dest: document`, `Accept-Encoding: gzip, deflate, br, zstd`, `Accept-Language: vi-VN,vi;q=0.8,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5`, `If-none-match: "3nc6-clad0m0d0f-g1p"`, `If-Modified-Since: Fri, 14 Jun 2024 17:32:37 GMT`. The bottom of the details pane shows the raw request data: `[Full request url: http://localhost:8080/]` and `[HTTP request 1/1]`.

Partie 2: Configuration théorique du réseau

Configuration d'un réseau sur le Docker

- Créer deux réseaux avec les IP suivants

docker network create --subnet=172.20.0.0/16 web-network

docker network create --subnet=172.28.0.0/16 client-network

p/s: L'IP que nous avons choisie ne doit pas exister auparavant.

```
C:\Users\Admin>docker network create --subnet=172.20.0.0/16 web-network
Error response from daemon: Pool overlaps with other one on this address space

C:\Users\Admin>docker network create --subnet=172.26.0.0/16 web-network
34f8d42ec36e94c4fef2faf036bc546bd237768a98ab20220b5f10a0cce38bf9

C:\Users\Admin>docker network create --subnet=172.21.0.0/16 client-network
Error response from daemon: Pool overlaps with other one on this address space

C:\Users\Admin>docker network create --subnet=172.28.0.0/16 client-network
bbdb71840815b87abb54db5587f18f9356dea679b841d58bccb162da03074d73
```

Ensuite, on lance des conteneurs pour chaque réseau:

- Un pour le web-network avec nginx
- Un pour le client-network avec ubuntu

```
C:\Users\Admin>docker run -d --name web-server --network web-network nginx
2c5b625af662516d3dc27f1a04d559dc5a50fac5a1ab61fb6168d838e515eeac

C:\Users\Admin>docker run -it --name client --network client-network ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
00d679a470c4: Pull complete
Digest: sha256:e3f92abc0967a6c19d0dfa2d55838833e947b9d74edbc0113e48535ad4be12a
Status: Downloaded newer image for ubuntu:latest
```


- Vérifier l'ip

```
root@1c6a224e00ee:/# ip a
bash: ip: command not found
root@1c6a224e00ee:/# apt update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [194 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [84.8 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [52.1 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Ign:10 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [3097 B]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [209 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [84.8 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [77.6 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [7519 B]
Get:10 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Fetched 3908 kB in 11s (353 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@1c6a224e00ee:/# apt install iproute2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  krb5-locales libatm1t64 libbbpf1 libcap2-bin libdb5.3t64 libelf1t64 libgssapi-krb5-2 libk5crypto3
  libkeyutils1 libkrb5-3 libkrb5support0 libmn10 libpam-cap libtirpc-common libtirpc3t64 libxtables12
Suggested packages:
  iproute2-doc python3:any krb5-doc krb5-user
The following NEW packages will be installed:
  iproute2 krb5-locales libatm1t64 libbbpf1 libcap2-bin libdb5.3t64 libelf1t64 libgssapi-krb5-2
  libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 libmn10 libpam-cap libtirpc-common libtirpc3t64
  libxtables12
0 upgraded, 17 newly installed, 0 to remove and 6 not upgraded.
Need to get 2914 kB of archives.
After this operation, 8264 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Après le téléchargement, vérifiez `ip a` pour obtenir l'IP de l'hôte(127.0.0. /8) et l'IP initialement créée pour le réseau 172.28.0.2 /16 (du client dans network client) et depuis client on ping server

```

root@1c6a224e00ee:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
14: eth0@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:1c:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.28.0.2/16 brd 172.28.255.255 scope global eth0
        valid_lft forever preferred_lft forever
root@1c6a224e00ee:/# ping web-server
bash: ping: command not found
root@1c6a224e00ee:/# iputils-ping
bash: iputils-ping: command not found
root@1c6a224e00ee:/# iputils-ping
bash: iputils-ping: command not found
root@1c6a224e00ee:/# apt install iputils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  iputils-ping

```

Ensuite, créez un réseau qui sert de pont pour relier les deux réseaux en utilisant 'bridge'

(Bridge est le réseau par défaut, si vous ne modifiez rien, il sera utilisé automatiquement. Plus précisément, vous pouvez inspecter le réseau bridge).

```

C:\Users\Admin>docker network create --driver bridge shared-network
e143a8a6062971114e0c5103c9fc76a2a67a323942afd96b132f989d2bd30ee8

C:\Users\Admin>docker network connect shared-network web-server

C:\Users\Admin>docker network connect shared-network client

C:\Users\Admin>

```

Après avoir terminé, connectez les deux réseaux au réseau commun pour établir la connexion. Une fois que les deux réseaux sont connectés à un pont réseau (bridge), depuis le client dans network client, essayez de faire un ping server web

Car si cela n'est pas fait, même si vous avez déjà installé iputilsping, le message d'erreur suivant s'affichera :

```

root@1c6a224e00ee:/# ping web-server
ping: web-server: Name or service not known

```

Après connecter deux réseaux:
Depuis le container client network-client

```
root@1c6a224e00ee:/# ping web-server
PING web-server (172.23.0.2) 56(84) bytes of data.
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=1 ttl=64 time=68.9 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=2 ttl=64 time=0.061 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=3 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=4 ttl=64 time=0.062 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=5 ttl=64 time=0.043 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=6 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=7 ttl=64 time=0.054 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=8 ttl=64 time=0.058 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=9 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=10 ttl=64 time=0.057 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=11 ttl=64 time=0.058 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=12 ttl=64 time=0.058 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=13 ttl=64 time=0.057 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=14 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=15 ttl=64 time=0.061 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=16 ttl=64 time=0.053 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=17 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=18 ttl=64 time=0.057 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=19 ttl=64 time=0.064 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=20 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=21 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=22 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=23 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=24 ttl=64 time=0.061 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=25 ttl=64 time=0.062 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=26 ttl=64 time=0.055 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=27 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=28 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=29 ttl=64 time=0.103 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=30 ttl=64 time=0.061 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=31 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=32 ttl=64 time=0.057 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=33 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=34 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=35 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=36 ttl=64 time=0.054 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=37 ttl=64 time=0.059 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=38 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=39 ttl=64 time=0.060 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=40 ttl=64 time=0.061 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=41 ttl=64 time=0.058 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=42 ttl=64 time=0.062 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=43 ttl=64 time=0.099 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=44 ttl=64 time=0.077 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=45 ttl=64 time=0.055 ms
64 bytes from web-server.shared-network (172.23.0.2): icmp_seq=46 ttl=64 time=0.059 ms
```

Cela fonctionne pour l'interaction des deux réseaux, car depuis le réseau client-network. Je peux également faire pour le réseau web-network.

Option

On peut essayer avec autre machine (par exemple client dans web-server)
Créer et lancer le conteneur avec network web-network.

```
C:\Users\Admin>docker run -it --name clientwn --network web-network ubuntu /bin/bash
```

Il faut connecter les pour les réseaux point

```
C:\Users\Admin>docker network connect shared-network clientwn
```

Et faire la même chose comme le web-server ça va nous aider à trouver le ping et l'ip depuis *clientwn*.

Si tu lance correctement tu vas trouver ip a depuis *clientwn* comme img:

```
root@725bf60e6be6:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
25: eth1@if26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:1a:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.26.0.3/16 brd 172.26.255.255 scope global eth1
        valid_lft forever preferred_lft forever
27: eth0@if28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:17:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.23.0.3/16 brd 172.23.255.255 scope global eth0
        valid_lft forever preferred_lft forever
```

Et pour le ping depuis *clientwn*. Il faut pinger le client dans client-network

```
root@725bf60e6be6:/# ping client
PING client (172.23.0.4) 56(84) bytes of data.
64 bytes from client.shared-network (172.23.0.4): icmp_seq=1 ttl=64 time=35.5 ms
64 bytes from client.shared-network (172.23.0.4): icmp_seq=2 ttl=64 time=0.099 ms
64 bytes from client.shared-network (172.23.0.4): icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from client.shared-network (172.23.0.4): icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from client.shared-network (172.23.0.4): icmp_seq=5 ttl=64 time=0.059 ms
64 bytes from client.shared-network (172.23.0.4): icmp_seq=6 ttl=64 time=0.062 ms
^C
--- client ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5135ms
```

Récapitulatif des commandes

Voici le récapitulatif des commandes pour la partie 2, mais vous pouvez suivre les étapes que j'ai écrites auparavant.

```
#!/bin/bash

# Création de réseaux Docker
docker network create --subnet=172.20.0.0/16 web-network
docker network create --subnet=172.21.0.0/16 client-network

# Création d'un réseau bridge partagé
docker network create --driver bridge shared-network

# Lancement du serveur web sur le réseau web-network
docker run -d --name web-server --network web-network nginx

# Lancement d'un client sur le réseau web-network
docker run -it --name clientwb --network web-network ubuntu /bin/bash

# Lancement d'un client sur le réseau client-network
docker run -it --name client --network client-network ubuntu /bin/bash

# Connexion du serveur web et du client au réseau bridge partagé
docker network connect shared-network web-server
docker network connect shared-network client

# Vérification de la connexion depuis le client web-client vers le serveur web
docker exec -it clientwb ping web-server

# Vérification de la connexion depuis le client vers le clientwb
docker exec -it client ping clientwb

# Affichage des adresses IP des conteneurs
docker exec -it web-server ip a
docker exec -it clientwb ip a
docker exec -it client ip a
```



Partie 3: Configuration pratique du réseaux et déploiement du serveur web sur le réseau

Afficher le site web sur les 2 machines clientes

```
C:\Users\Admin>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
725bf60e6be6	ubuntu	"/bin/bash"	41 minutes ago	Exited (127) 28 minutes ago		clientwn
1c6a224e00ee	ubuntu	"/bin/bash"	19 hours ago	Exited (127) 2 hours ago		client
2c5b625af662	nginx	"/docker-entrypoint..."	19 hours ago	Up 19 hours	80/tcp	web-server
81c11be04b64	debian	"/bin/bash"	3 days ago	Exited (137) 27 minutes ago		webserver

Le container clientwn et web-server est dans le même web-network

Container client est dans network client

Dans la partie précédente, nous avons configuré le réseau.

Maintenant, essayez d'accéder au serveur web en utilisant le domaine (web-server) et l'adresse IP pour voir si l'accès est possible. Si l'accès fonctionne, passez à l'étape suivante.

Sur les machines clientes, installez le paquet curl avec la commande:

```
apt update && apt install curl -y.
```

Les informations concernant l'IP du serveur web:

```
C:\Users\Admin>docker exec -it web-server bash
root@2c5b625af662:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:1a:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.26.0.2/16 brd 172.26.255.255 scope global eth0
        valid_lft forever preferred_lft forever
17: eth1@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:17:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.23.0.2/16 brd 172.23.255.255 scope global eth1
        valid_lft forever preferred_lft forever
root@2c5b625af662:/#
```

Après, on a bien vérifié comme ce que je fais pour la partie 2. On va essayer de faire la commande depuis le client (client-network)

```
docker exec -it client bash
```

```
curl web-server
```

```
C:\Users\Admin>docker exec -it client bash
root@1c6a224e00ee:/# curl web-server
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```

```
curl 172.23.0.2
```

(ip de webserver)

```
root@1c6a224e00ee:/# curl 172.23.0.2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```


Ensuite, essayez depuis la deuxième machine cliente de faire un curl vers le web-server. Accédez au conteneur *clientwn*.

```
C:\Users\Admin>docker exec -it clientwn bash
root@725bf60e6be6:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
25: eth1@if26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:1a:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.26.0.3/16 brd 172.26.255.255 scope global eth1
        valid_lft forever preferred_lft forever
27: eth0@if28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:17:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.23.0.3/16 brd 172.23.255.255 scope global eth0
        valid_lft forever preferred_lft forever
```

```
root@725bf60e6be6:/# curl web-server
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@725bf60e6be6:/# curl 172.23.0.2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

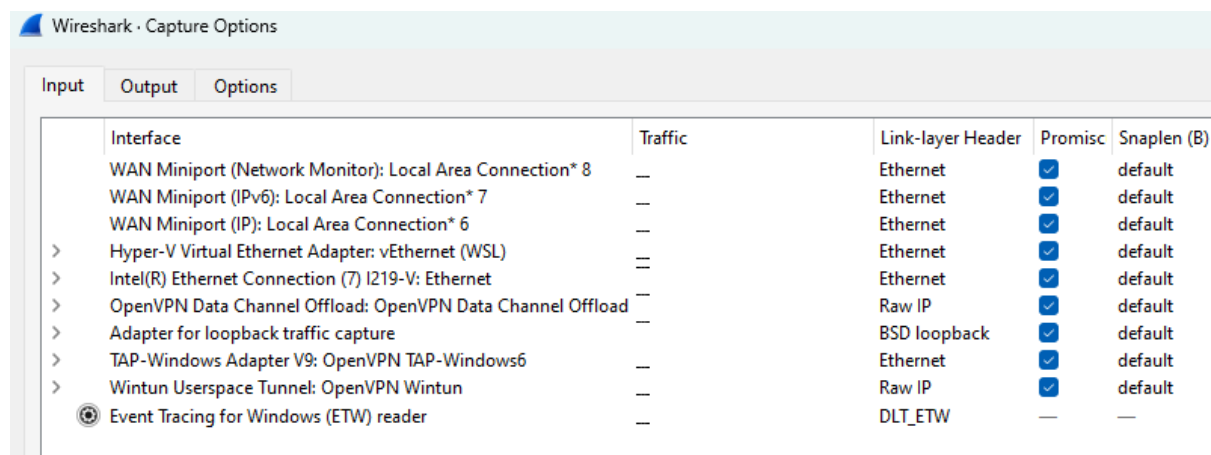
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```


Donc, les deux clients peuvent accéder au serveur web.

Passez à l'étape suivante. L'exigence à ce stade est : "Vous pouvez accéder à Docker via SSH depuis les machines clientes. Vous commenterez les trames de données HTTP circulant."

Pour simplifier, je n'utiliserai pas le serveur SSH mais plutôt la commande 'exec' dans le conteneur pour créer des requêtes et capturer directement les paquets.

Remarque : En temps normal, Wireshark ne peut pas capturer les paquets échangés en interne entre les conteneurs car il n'y a pas d'interface pour écouter ces connexions.



Comme illustré ci-dessus, il n'y a pas d'interface liée à Docker. Donc, si vous utilisez simplement Wireshark, vous ne pourrez pas capturer les paquets.

J'ai trouvé deux solutions pour capturer les trames dans le conteneur :

1. Utiliser tcpdump pour capturer les paquets, les enregistrer dans un fichier, puis les ouvrir avec Wireshark pour les analyser. (Il est possible d'analyser directement avec tcpdump, mais ce ne sera pas aussi visuel.)
2. Utilisez edgeshark + plugin

Pour simplifier et éviter d'installer des outils supplémentaires sur l'hôte, je vais procéder avec la méthode 1. Utiliser tcpdump pour capturer les paquets et les analyser.

Depuis la machine cliente, exécutez la commande `apt update && apt install tcpdump -y`. Ensuite, exécutez la commande `tcpdump --interface any -w capture.pcap` pour capturer les paquets depuis le réseau.

```

C:\Users\Admin>docker exec -it client bash
root@1c6a224e00ee:/# apt update && apt install tcpdump -y.
apt update && apt install tcpdump -y.
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Fetched 126 kB in 3s (38.4 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
E: Command line option '.' [from -y.] is not understood in combination with the other options.
root@1c6a224e00ee:/# apt update && apt install tcpdump -y.
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
E: Command line option '.' [from -y.] is not understood in combination with the other options.
root@1c6a224e00ee:/# tcpdump --interface any -w capture.pcap
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes

```

Une fois qu'il est prêt à écouter, nous allons ouvrir un autre terminal pour accéder au même conteneur.

```

C:\Users\Admin>docker exec -it client bash
root@1c6a224e00ee:/# curl web-server
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

```

Après, nous arrêtons l'écoute pour récupérer la capture.

```
C:\Users\Admin>docker exec -it client bash
root@1c6a224e00ee:/# curl web-server
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@1c6a224e00ee:/# ^C
root@1c6a224e00ee:/#

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
E: Command line option '.' [from -y.] is not understood in combination with the other options.
root@1c6a224e00ee:/# tcpdump --interface any -w capture.pcap
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
^C21 packets captured
25 packets received by filter
0 packets dropped by kernel
root@1c6a224e00ee:/# pwd
/
root@1c6a224e00ee:/# ls -la
total 80
drwxr-xr-x 1 root root 4096 Jun 18 15:31 .
drwxr-xr-x 1 root root 4096 Jun 18 15:31 ..
-rwxr-xr-x 1 root root  0 Jun 17 15:18 .dockerenv
lrwxrwxrwx 1 root root   7 Apr 22 13:08 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Mar 31 09:00 bin.usr-is-merged
drwxr-xr-x 2 root root 4096 Apr 22 13:08 boot
-rw-r--r-- 1 tcpdump tcpdump 2768 Jun 18 17:35 capture.pcap
drwxr-xr-x 5 root root 360 Jun 18 14:03 dev
drwxr-xr-x 1 root root 4096 Jun 18 15:25 etc
drwxr-xr-x 3 root root 4096 May 30 02:07 home
lrwxrwxrwx 1 root root   7 Apr 22 13:08 lib -> usr/lib
lrwxrwxrwx 1 root root   9 Apr 22 13:08 lib64 -> usr/lib64
drwxr-xr-x 2 root root 4096 May 30 02:03 media
drwxr-xr-x 2 root root 4096 May 30 02:03 mnt
drwxr-xr-x 2 root root 4096 May 30 02:03 opt
dr-xr-xr-x 246 root root   0 Jun 18 14:03 proc
drwxr-xr-x 1 root root 4096 Jun 18 08:27 root
drwxr-xr-x 1 root root 4096 Jun 18 15:25 run
lrwxrwxrwx 1 root root   8 Apr 22 13:08/sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Mar 31 09:00/sbin.usr-is-merged
drwxr-xr-x 2 root root 4096 May 30 02:03 srv
dr-xr-xr-x 11 root root   0 Jun 18 15:29 sys
drwxrwxrwt 1 root root 4096 Jun 18 17:34 tmp
drwxr-xr-x 1 root root 4096 May 30 02:03 usr
drwxr-xr-x 1 root root 4096 May 30 02:07 var
root@1c6a224e00ee:/#
```

La commande `pwd` permet de savoir où nous sommes, donc nous pouvons connaître le chemin pour nous aider à télécharger le fichier que nous venons de capturer. Utilisez `ls -la` pour voir le nom du fichier.

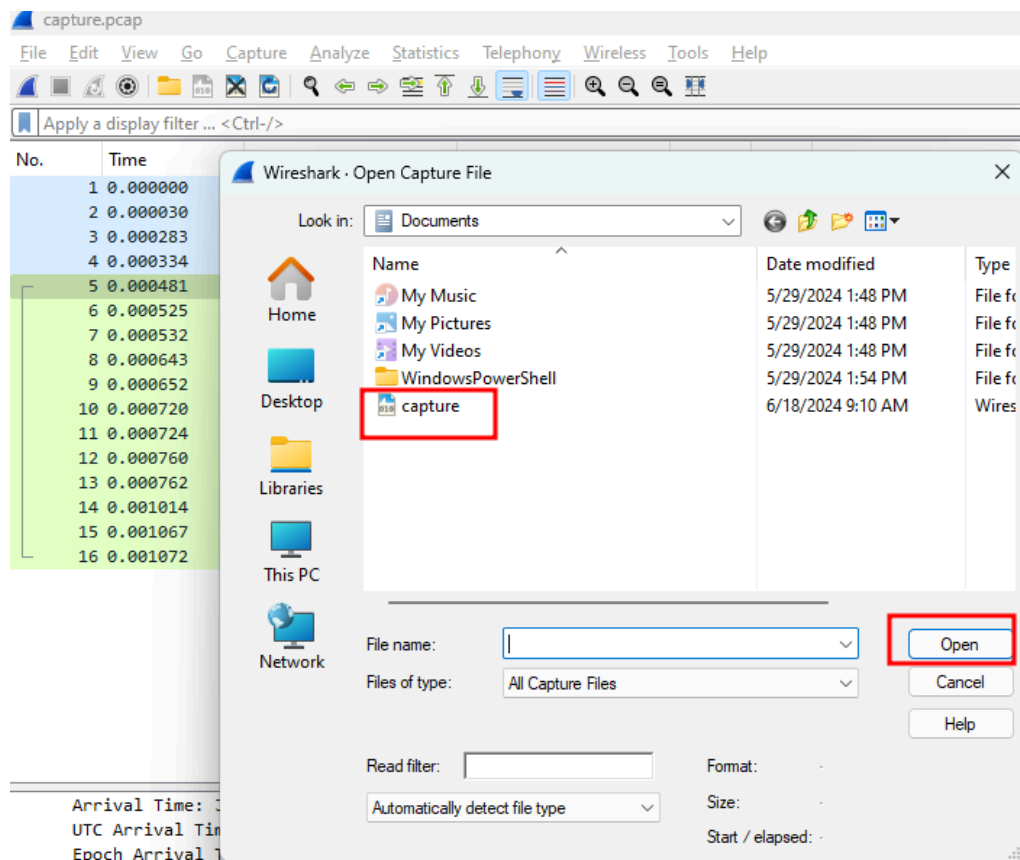
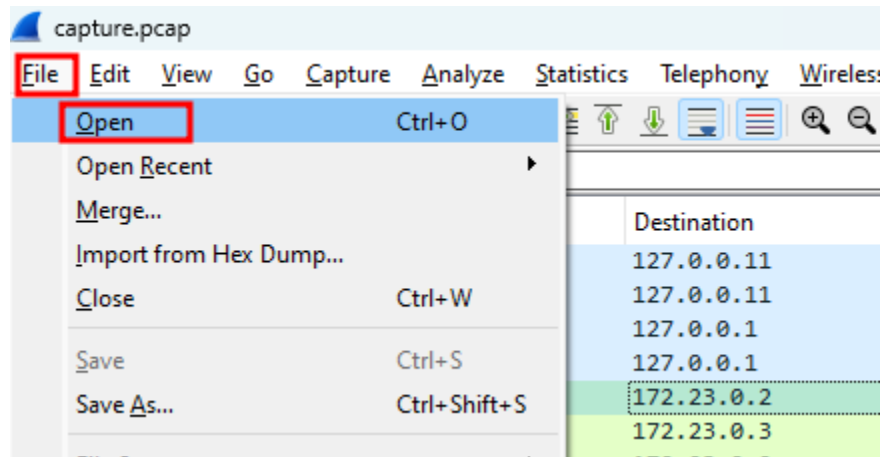
Ensuite, nous lançons un autre terminal avec la commande suivante :

`docker cp client:/capture.pcap ./`

```
C:\Users\Admin>docker cp client:/capture.pcap ./
Successfully copied 4.1kB to C:\Users\Admin\.
```

Cette commande permet de copier le fichier `capture.pcap` depuis le conteneur `client` vers le répertoire courant.

Ouvrez ensuite le fichier avec Wireshark.



Dans Wireshark, cherchez dans le menu "Fichier" et choisissez "Ouvrir" pour ouvrir le fichier de capture. Ensuite, utilisez le filtre `http` pour afficher uniquement les trames HTTP.

Comme mentionné précédemment, nous avons capturé 21 paquets.

^C21 packets captured
25 packets received by filter

TRAMES HTTP CAPTURÉES - client du client-network

Depuis client dans le réseau client-network:

http

No.	Time	Source	Destination	Protocol	Length	Info
8	0.006464	172.23.0.3	172.23.0.2	HTTP	145	GET / HTTP/1.1
12	0.104497	172.23.0.2	172.23.0.3	HTTP	687	HTTP/1.1 200 OK (text/html)

> Frame 8: 145 bytes on wire (1160 bits), 145 bytes captured (1160 bits)
> Linux cooked capture v2
> Internet Protocol Version 4, Src: 172.23.0.3, Dst: 172.23.0.2
✓ Transmission Control Protocol, Src Port: 34930, Dst Port: 80, Seq: 1, Ack: 1, Len: 73
Source Port: 34930
Destination Port: 80
[Stream index: 0]
> [Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 73]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 811869927
[Next Sequence Number: 74 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 13916856
1000 = Header Length: 32 bytes (8)
> Flags: 0x018 (PSH, ACK)
Window: 502
[Calculated window size: 64256]
[Window size scaling factor: 128]
Checksum: 0x58a3 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
> [Timestamps]
> [SEQ/ACK analysis]
TCP payload (73 bytes)
✓ Hypertext Transfer Protocol
> GET / HTTP/1.1\r\nHost: web-server\r\nUser-Agent: curl/8.5.0\r\nAccept: */*\r\n\r\n[Full request URI: http://web-server/]
[HTTP request 1/1]
[Response in frame: 12]

Commentaires de trames

Trame 8 : Requête HTTP GET

1. **Requête GET envoyée par la machine cliente (172.23.0.3) au serveur web (172.23.0.2) :**
 - **Source IP :** 172.23.0.3
 - **Destination IP :** 172.23.0.2

- **Source Port** : 54903
- **Destination Port** : 80
- **Méthode HTTP** : GET
- **Ressource demandée** : /
- **En-tête Host** : web-server
- **User-Agent** : curl/8.5.0
- **Accept** : /

Trame 12 : Réponse HTTP 200 OK

2. **Réponse 200 OK envoyée par le serveur web (172.23.0.2) à la machine cliente (172.23.0.3) :**
 - **Source IP** : 172.23.0.2
 - **Destination IP** : 172.23.0.3
 - **Source Port** : 80
 - **Destination Port** : 54903
 - **Code de réponse** : 200 OK
 - **Contenu** : text/html

Conclusion pour trames capture client

- **Trame 8** montre une requête HTTP GET de la machine cliente (172.23.0.3) vers le serveur web (172.23.0.2), demandant la ressource / sur le port 80.
- **Trame 12** montre la réponse du serveur web (172.23.0.2) à la machine cliente (172.23.0.3) avec un code de réponse 200 OK, indiquant que la requête a été traitée avec succès et le contenu est de type text/html.

TRAMES HTTP CAPTURÉES - clientwn du réseau server-network

depuis *clientwn* dans le réseau server-network :

```
C:\Users\Admin>docker exec -it clientwn bash
root@725bf60e6be6:/# apt update && apt install tcpdump
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tcpdump is already the newest version (4.99.4-3ubuntu4).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
root@725bf60e6be6:/# tcpdump --interface any -w capture.pcap
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
^C13 packets captured
13 packets received by filter
0 packets dropped by kernel
root@725bf60e6be6:/#
```

```
<style>
html { color:scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@725bf60e6be6:/# curl 172.26.0.2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color:scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@725bf60e6be6:/#
```

La même ce qu'on a fait precedent

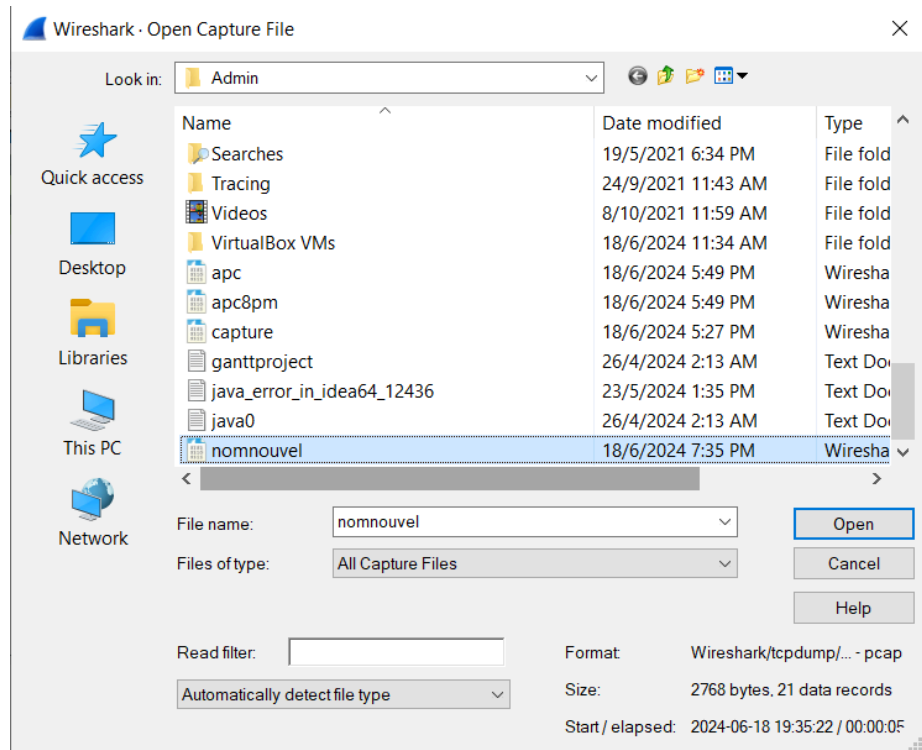
Attention: changer votre nom fichier de capture avec command (si il y a erreur comme imag)

```
docker cp client:/capture.pcap ./nomnouvel.pcap
```

```
C:\Users\Admin>docker cp client:/capture.pcap ./
Successfully copied 4.61kB to C:\Users\Admin\.\
remove C:\Users\Admin\capture.pcap: The process cannot access the file because it is being used by another process.

C:\Users\Admin>docker cp client:/capture.pcap ./nomnouvel.pcap
Successfully copied 4.61kB to C:\Users\Admin\nomnouvel.pcap

C:\Users\Admin>
```



Commentaires de trames:

Trame 9 : Requête HTTP GET

1. **Frame 9 :**
 - **Bytes on wire :** 145 bytes
 - **Captured bytes :** 145 bytes
 - **Interface :** Ethernet
2. **Internet Protocol Version 4 (IPv4) :**
 - **Source IP Address :** 172.23.0.4 (Machine cliente)
 - **Destination IP Address :** 172.23.0.2 (Serveur web)
3. **Transmission Control Protocol (TCP) :**
 - **Source Port :** 42402
 - **Destination Port :** 80 (HTTP)
 - **Flags :** 0x018 (PSH, ACK)
 - **Sequence Number :** 1
 - **Acknowledgment Number :** 1
 - **Header Length :** 32 bytes
 - **Window :** 64260
4. **Hypertext Transfer Protocol (HTTP) :**
 - **Request Method :** GET
 - **Request URI :** /
 - **HTTP Version :** HTTP/1.1

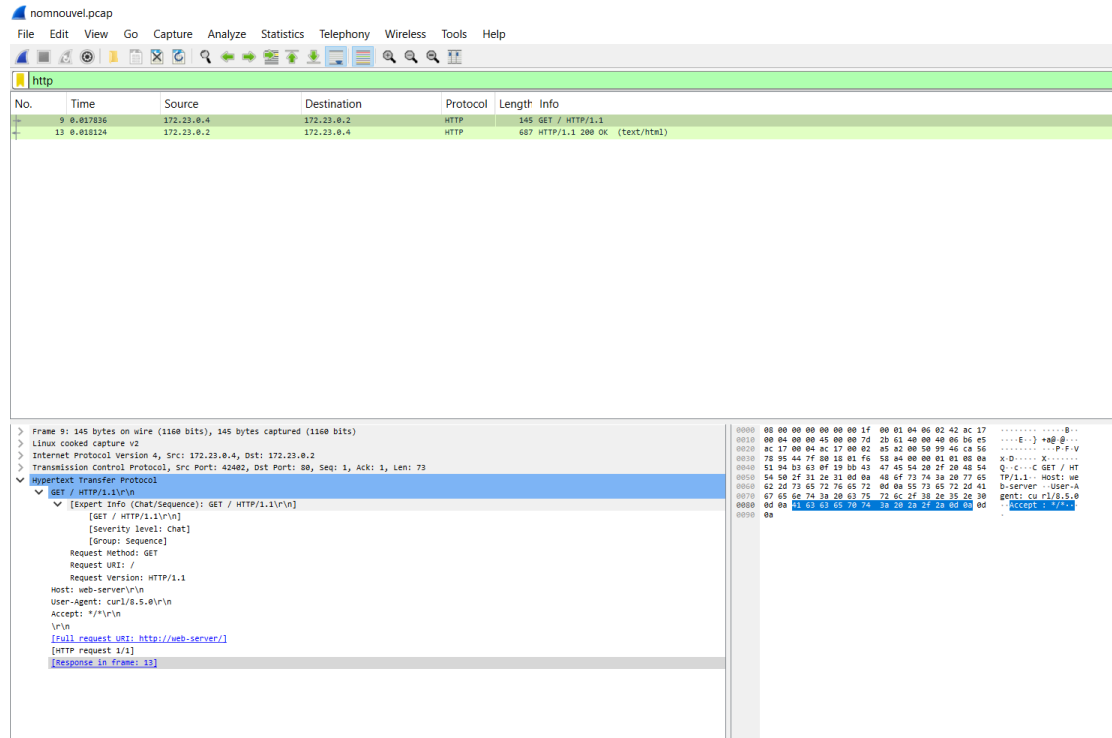
- **Host** : web-server
- **User-Agent** : curl/8.5.0
- **Accept** : /

Trame 13 : Réponse HTTP 200 OK

1. **Frame 13** :
 - **Bytes on wire** : 687 bytes
 - **Captured bytes** : 687 bytes
 - **Interface** : Ethernet
2. **Internet Protocol Version 4 (IPv4)** :
 - **Source IP Address** : 172.23.0.2 (Serveur web)
 - **Destination IP Address** : 172.23.0.4 (Machine cliente)
3. **Transmission Control Protocol (TCP)** :
 - **Source Port** : 80 (HTTP)
 - **Destination Port** : 42402
 - **Flags** : 0x018 (PSH, ACK)
 - **Sequence Number** : 1
 - **Acknowledgment Number** : 74
 - **Header Length** : 32 bytes
 - **Window** : 64260
4. **Hypertext Transfer Protocol (HTTP)** :
 - **Response Code** : 200 OK
 - **HTTP Version** : HTTP/1.1
 - **Content-Type** : text/html

Différences principales

1. **Adresses IP Source et Destination** :
 - Dans la première image, la source est 172.23.0.3 et la destination est 172.23.0.2.
 - Dans la deuxième image, la source est 172.23.0.4 et la destination est 172.23.0.2.
2. **Ports Source** :
 - Dans la première image, le port source pour la requête est 54903.
 - Dans la deuxième image, le port source pour la requête est 42402.
3. **Trames spécifiques** :
 - Les trames spécifiques diffèrent par les numéros de trame : 8 et 12 dans la première image, 9 et 13 dans la deuxième image.



3 Serveurs web sur 3 Containers docker

Ensuite, pour créer 3 conteneurs de serveurs web et y accéder depuis le réseau LAN sur un autre ordinateur. Dans cette partie, vous pouvez choisir 3 serveurs web différents ou le même type. Il suffit de lancer les 3 commandes suivantes :

```
docker run -d -p 80:80 nginx
docker run -d -p 8085:80 httpd
docker run -d -p 8088:80 caddy
```

<http://votreIP:80>

<http://votreIP:8085>

<http://votreIP:8088>

Pour trouver ip, il faut lancer cette commande:

```
C:\Users\Admin>ipconfig
```

Et cherchez cette ligne pour votre ip

```
Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : 
IPv6 Address. . . . . : 2a02:8440:9118:f9e:4792:d378:cf23:1255
Temporary IPv6 Address. . . . . : 2a02:8440:9118:f9e:5107:7f18:3bfd:3001
Link-local IPv6 Address . . . . . : fe80::9dac:9a62:79b6:354b%11
IPv4 Address. . . . . : 172.20.10.2
Subnet Mask . . . . . : 255.255.255.240
Default Gateway . . . . . : fe80::f8e9:4eff:fe20:2564%11
                             172.20.10.1
```

Par exemple :

```
C:\Users\Admin>docker run -d -p 8081:80 caddy
Unable to find image 'caddy:latest' locally
latest: Pulling from library/caddy
d25f557d7f31: Already exists
2c0c17092266: Pull complete
ed0d427b3a04: Pull complete
1ea7e62409d2: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:60199fbf2046892e0aa4b19c7d3adf71f530c36abc65728627422148a75b3475
Status: Downloaded newer image for caddy:latest
e4183bb9ad7921bbfca6c8d5473950f8e07b88da37aebd7ae2ee561b787951

C:\Users\Admin>docker run -d -p 8080:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
2c0c17092266: Pull complete
ed0d427b3a04: Pull complete
4f4fb700ef54: Pull complete
0d1465828338: Pull complete
4a16a983b278: Pull complete
9129890c4c50: Pull complete
Digest: sha256:10182d88d7fb5161ae0f6f758cba7adc56d4aac2dc950e51d72c0cf68967cea
Status: Downloaded newer image for httpd:latest
3bc50c3fe8f086468a10d15a31a7ba106a756fe2d7f7357b055a3a10b908e98e
docker: Error response from daemon: driver failed programming external connectivity on endpoint optimistic_ishizaka (dfcf6f0fcbddb45abe83aa8d1b3cb4a2ef140d0a9b1a3bfc9e234e55b90dac03): Bind for 0.0.0.0:8080 failed: port is already allocated.

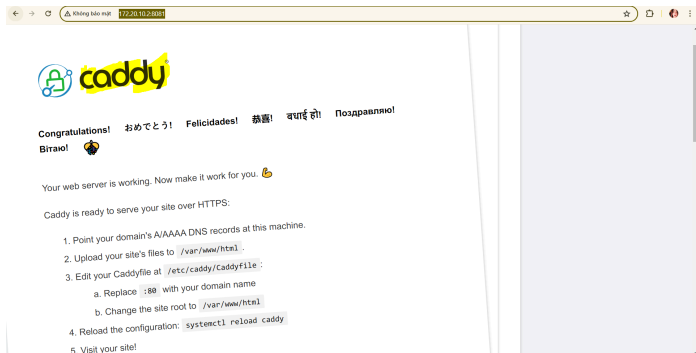
C:\Users\Admin>docker run -d -p 8085:80 httpd
4e7a5c385c59780e5a901bd67a9b259fcebcd04fabe30fce22de9f848858ebf

C:\Users\Admin>docker run -d -p 8088:80 nginx
09aeb66a6cb51bf391dacd0dc05c550d9cbf9d7105bf4f38a7f9616acdbb6d89
```

Démonstration avec mon réseau

Avec mon ip, je lance dans navigateur :

<http://172.20.10.2:8081/>



<http://172.20.10.2:8085/>



It works!

http://172.20.10.2:8088/



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Tester avec mon téléphone (même réseau)

