Projet Bulles docker 2



**Rapport de projet**

Zeqiri Amir – CIN2B

ETML, Sébeillon – N510B

40 Périodes

Maître : M. Maître

Table des matières

[1 Description du projet dans son ensemble 3](#_Toc192667882)

[1.1 Titre 3](#_Toc192667883)

[1.2 Sujet 3](#_Toc192667884)

[2 Analyse & Conception 3](#_Toc192667885)

[2.1 Les différents outils utilisés 3](#_Toc192667886)

[3 Réalisation 3](#_Toc192667887)

[3.1 Dockerfile 3](#_Toc192667888)

[3.2 Dockerfile 2 4](#_Toc192667889)

[3.3 Docker Compose 5](#_Toc192667890)

[4 Tests 7](#_Toc192667891)

[4.1 SSH (Connexion à distance sécurisée) 7](#_Toc192667892)

[4.1.1 Serveur 7](#_Toc192667893)

[4.1.2 Client 7](#_Toc192667894)

[4.2 Netcat (Outil réseau polyvalent) 8](#_Toc192667895)

[4.3 Nmap (Scanner de ports) 8](#_Toc192667896)

[4.4 Tshark (Analyse et capture réseau, Wireshark CLI) 9](#_Toc192667897)

[4.5 TCP Dump (Capture et analyse de paquets) 10](#_Toc192667898)

[5 Problèmes rencontrés 10](#_Toc192667899)

[5.1 Création d’une image avec dockerfile 10](#_Toc192667900)

[5.2 Erreur de build avec le docker-compose (Netcat) 11](#_Toc192667901)

[5.3 Erreur avec les permissions (Tshark) 11](#_Toc192667902)

[6 Sources & Aides 11](#_Toc192667903)

[7 Conclusion (pas fini) 11](#_Toc192667904)

# Description du projet dans son ensemble

## Titre

P\_Bulles DockerNetworkToolkit

## Sujet

Bienvenue dans le projet "P\_Bulle DockerNetworkToolkit". Ce projet vise à vous initier aux concepts de la virtualisation par conteneur, en vous permettant de créer, gérer et orchestrer des conteneurs Docker. À la fin de ce projet, vous serez en mesure de comprendre les principes de base de Docker, de créer des conteneurs individuels et de les orchestrer en utilisant Docker Compose.

# Analyse & Conception

Dans ce projet il y’aura avoir besoin de deux container Docker, le premier servira de boîte à outils comportant Tshark, tcpdump, Nmap et SSH comme outils. Le deuxième et dernier container servira tester ces outils-là. Tous les tests seront détaillés dans le [point 4](#_Tests) du rapport.

## Les différents outils utilisés

SSH : Connexion sécurisée à distance via un terminal.

tcpdump : Capture et analyse de paquets réseau en ligne de commande.

tshark : Version CLI de Wireshark pour l'analyse des paquets réseau.

nmap : Scanner de ports et analyseur de réseaux pour la sécurité.

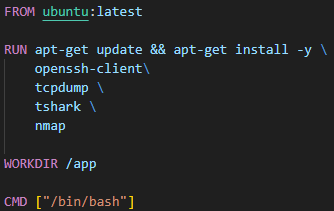
netcat : Outil réseau polyvalent pour le debug et la communication.

iperf3 : Teste la bande passante et la performance réseau.

apache2 : Serveur web open-source pour héberger des sites et applications.

# Réalisation

## Dockerfile

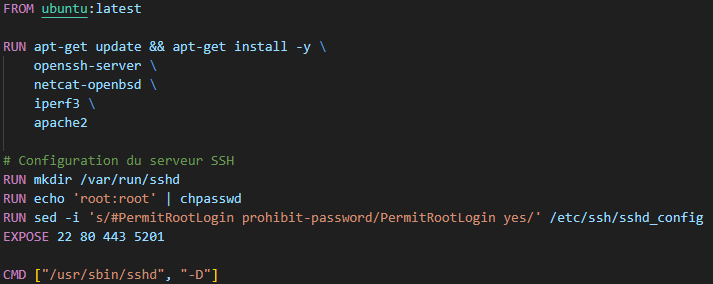


* FROM : Fais appel à la dernière version de l’image officielle Ubuntu, latest singifie que c’est la dernière version que Ubuntu a décidé d’utiliser comme dernière version.
* RUN : Ce point sert à installer les paquets. Premièrement, apt-get update sert à mettre à jour les paquets déjà existants.

apt-get install –y sert à installer les différents paquets listés juste en dessous : SSH Client, TCP dump, Tshark, Nmap.

* WORKDIR : Spécifie le répertoire de travail sur /app.
* CMD : Avec cette commande /bin/bash, le container ouvrira un terminal en bash.

## Dockerfile 2



* FROM : Fais appel à la dernière version de l’image officielle Ubuntu, latest singifie que c’est la dernière version que Ubuntu a décidé d’utiliser comme dernière version.
* RUN : Ce point sert à installer les paquets. Premièrement, apt-get update sert à mettre à jour les paquets déjà existants.

apt-get install –y sert à installer les différents paquets listés juste en dessous : SSH Server cette fois-ci, Netcat bsd, Iperf3 et Apache2.

* RUN mkdir : Ce point crée le répertoire pour le serveur SSH.
* RUN echo : Assigne le mot de passe de l’utilisateur root en root.
* RUN sed : Cette commande modifie la configuration de SSH dans sshd\_configs et fait en sorte d’autoriser la connexion en root SSH.
* EXPOSE : Mappe les ports pour les différents services. 22 pour SSH, 80 pour Apache2 (http), 443 HTTPS, 5201 pour Iperf3.
* CMD : La commande "/usr/sbin/sshd", "-D" lance le serveur SSH en mode Deamon afin qu’il reste actif constamment.

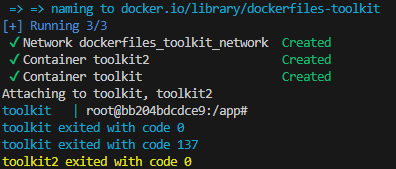
## Docker Compose

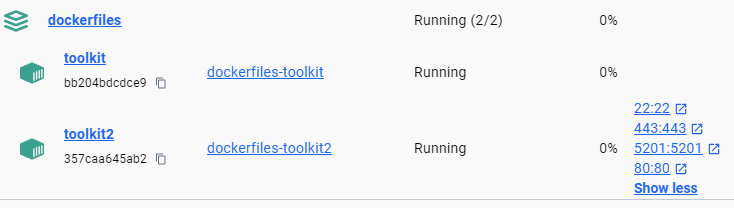
Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

* SERVICES : Définit les différents services qui seront déployés avec Docker.
* TOOLKIT : Indique sur quel container effectuer le build et assigner les différents paramètres.
* BUILD : Dans le build se trouve le context et le dockerfile. Context sert à indiquer l’emplacement du fichier dockerfile, et Dockerfile sert à indiquer le nom du fichier dockerfile.
* CONTAINER\_NAME : Indique le nom que va porter le container, dans ce cas c’est "toolkit".
* NETWORKS (Services) : Indique sur quel réseau va se connecter le container, dans ce cas c’est le réseau "toolkit\_network".
* TTY : Ce point sert à avoir un terminal interactif en bash dans ce cas.
* DEPENDS\_ON : Cela indique le premier container "toolkit" dépend du deuxième container "toolkit2". Le deuxième container se lancera donc en premier.
* PORTS : Mappe les différents services à des ports d’hôte(Windows) spécifiques. Pour SSH c’est le port 22 qui sera assigné dans le container mais également sur Windows.
* NETWORKS (Hors services) : Définit les différents réseaux qui seront déployés avec Docker.
* TOOLKIT\_NETWORK : C’est le nom du réseau qui va être créer lors de l’orchestration, celui-ci est utilisé dans les deux containers.
* DRIVER : Dans ce point, est signifié le type de réseau qu’il doit employer. Dans ce projet, le type idéal pour faire communiquer des containers entre eux est le type "bridge" qui signifier "pont".

Voici le résultat de l’orchestration de tous les containers avec Docker compose. Le build s’est fait avec cette commande : docker-compose up –build.



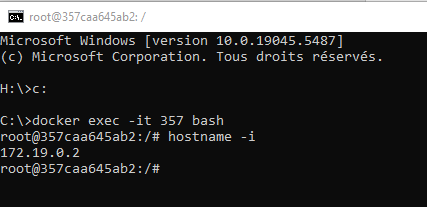


# Tests

## SSH (Connexion à distance sécurisée)

Pour réaliser le test SSH entre deux containers, entre le client et le serveur. Voici les différentes étapes à faire sur le serveur en premier et le client pour finir.

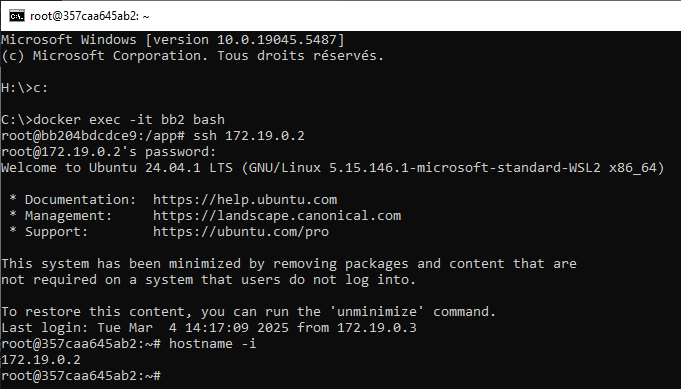
### Serveur



Pour commencer il faut se positionner dans le container en bash avec cette commande docker exec –it 357 bash. 357 est égal aux trois premiers chiffres de l’ID du container(le terminal va savoir de quel container il s’agit).

Il est primordiale de connaître l’adresse IP du serveur SSH, et pour cela il suffit d’entrer cette commande hostname -i. L’adresse IP du serveur SSH est 172.19.0.2

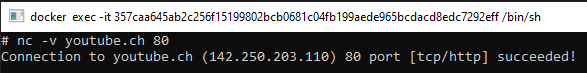
### Client



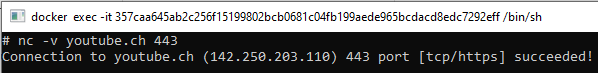
Après s’être placé dans le client SSH, je lance la commande shh 172.19.0.2 pour pouvoir accéder à au serveur SSH. Le mot de passe définit dans le dockerfile est demandé, dans ce cas c’est "root".

Une fois ces deux étapes finies nous accédons au serveur SSH depuis le client. Pour en faire la vérification, il suffit de retaper la commande hostname –i et nous voyons que l’adresse IP correspond au serveur. L’adresse IP du client est 172.19.0.3, comme indiqué dans le "Last login".

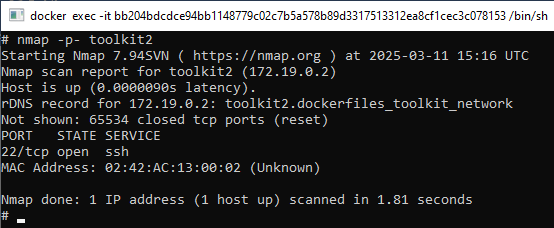
## Netcat (Outil réseau polyvalent)

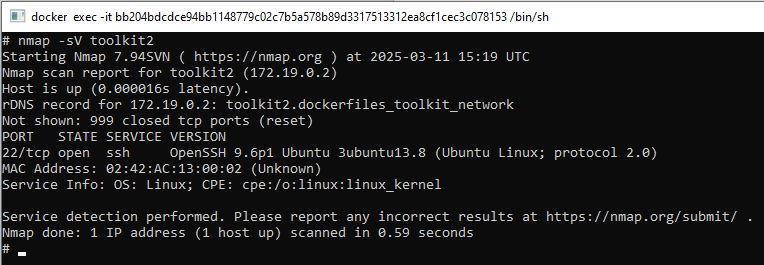


Pour tester le bon fonctionnement de Netcat, il suffit d’entrer cette commande : nc –v youtube.ch 80. Cette commande établit une connexion TCP vers youtube.ch sur le port 80 en HTTP. Il est possible de le faire en HTTPS sur le port 443.



## Nmap (Scanner de ports)

  
  
Pour tester le bon fonctionnement de Nmap, il suffit d’entrer cette commande dans le premier container : nmap -p- toolkit2. Cette commande va identifier tous les ports du deuxième container et d’afficher ceux qui sont ouverts.

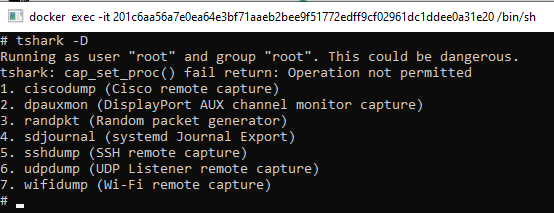


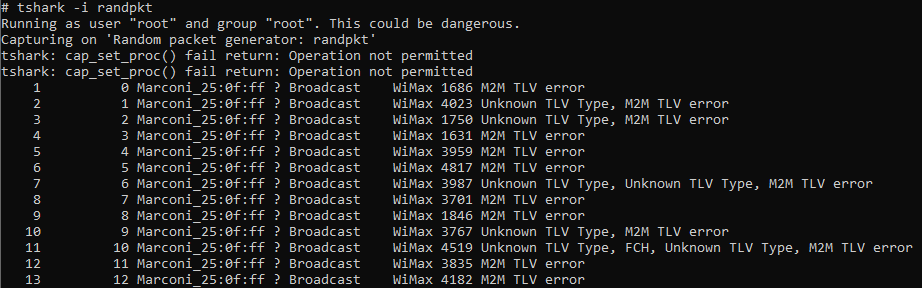
Avec la commande nmap -sV toolkit2 on va afficher les services qui tournent et sur quels ports ils sont actifs. On peut voir que SSH est actif sur son port assigné 22, la version est également affichée.

Pour scanner seulement certains ports, il faut entrer cette commande : nmap -p 22,80,443 toolkit2

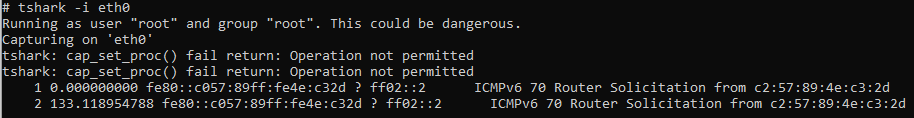
## Tshark (Analyse et capture réseau, Wireshark CLI)

Pour afficher les interfaces réseau, il suffit d’entrer la commande tshark –D, et voici les interfaces accessibles :





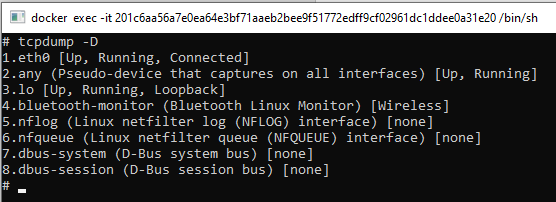
Pour capturer le trafic d’une interface précise, il suffit d’entrer cette commande : tshark -i randpkt. Randpkt est un générateur de paquet random, ce trafic et donc faux.

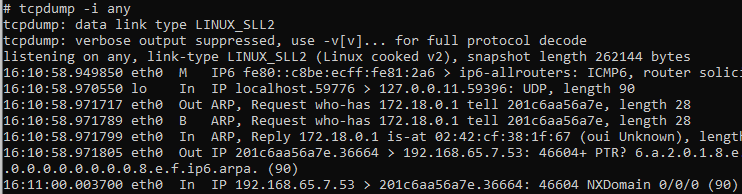


Ceci est la capture de trafic de l’interface réseau "eth0" qui est celle par défaut. Ce trafic est cette fois-ci réel.

## TCP Dump (Capture et analyse de paquets)

Pour afficher les interfaces réseau, il suffit d’entrer la commande tcpdump –D, et voici les interfaces accessibles :

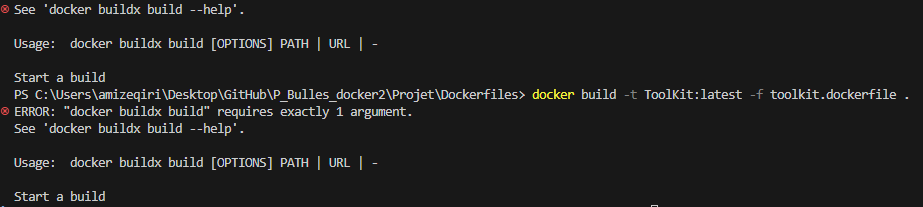




Avec la commande tcpdump –i any je vois le trafic de toutes les interfaces.

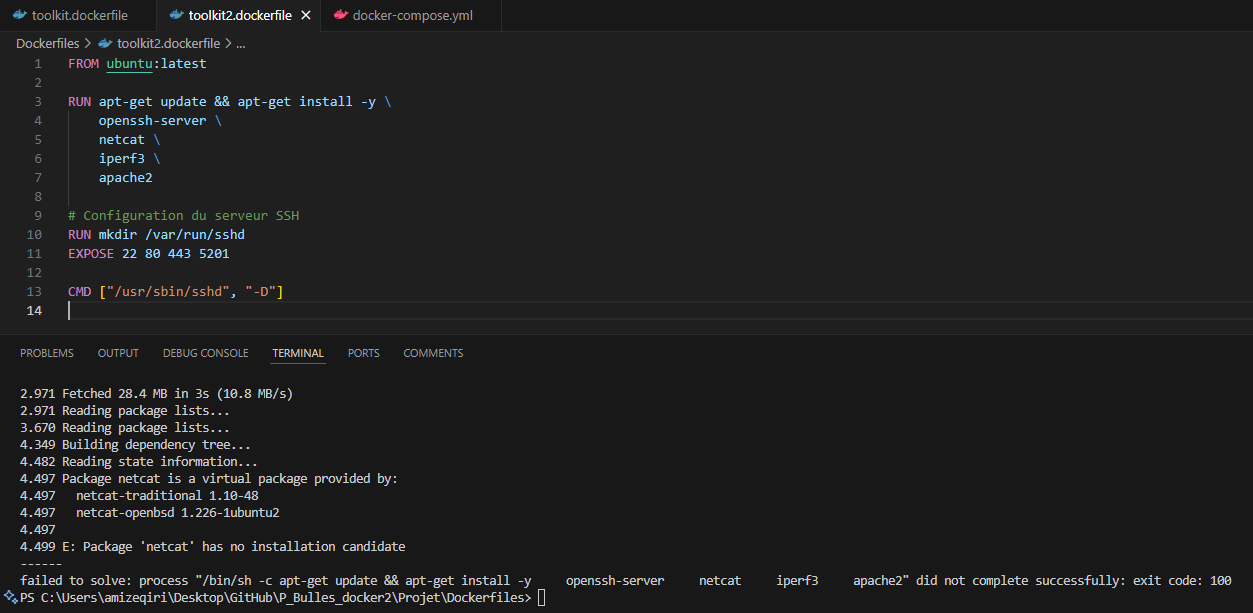
# Problèmes rencontrés

## Création d’une image avec dockerfile



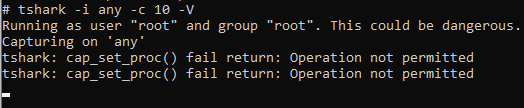
Ce problème est arrivé au début du projet. Il était impossible de créer une image à partir du dockerfile. Après multiples essais, toujours impossible de résoudre ce soucis. Le prof est intervenu est à indiquer qu’il fallait simplement relancer Docker et que ce n’était pas un réel problème.

## Erreur de build avec le docker-compose (Netcat)

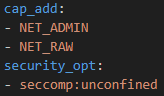


Il manquait la version de Netcat dans le dockerfile. Je ajouté netcat-openbsd. Le problème était résolu.

## Erreur avec les permissions (Tshark)



L’accès à Tshark et aux requêtes était impossible à cause des différentes permissions…



Voici les paramètres ajoutés au docker compose…

# Sources & Aides

[Docker hub](https://hub.docker.com/) : Ce site m’a servi pour pouvoir récupérer l’image Ubuntu.

# Conclusion

Avant de commencer ce projet, je savais à peine utiliser Docker desktop. Maintenant, je sais non seulement utiliser Docker desktop mais surtout Docker en ligne de commande. La notion de dockerfile était très moindre, aujourd’hui je sais ce que je fais. Egalement concernant le docker compose, je n’avais aucune idée de comment était composé le docker compose. Aujourd’hui je comprends tout à fait, à quoi sert-il, que signifie les différents paramètres, etc. Sur les Networks de Docker, j’en ai beaucoup appris, je ne connaissais pas leurs existences, aujourd’hui, je sais reconnaître les différents types de réseau(driver). Concernant les outils, j’ai pu apprendre à quoi ils servent, mais surtout, comment les utiliser.