

Activité : publier le service Web avec Docker

Pour rendre un service accessible, le conteneur doit être lancé en arrière plan.

Docker utilise la technique classique du mappage de port :

- un port de l'hôte va être redirigé vers un port du container.

Docker dispose pour cela d'une **interface réseau** sur la machine hôte.

Sous Linux

```
ip a
...
docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f6:de:8b:5b brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
...

```

Sous Windows

Une commande possible est la suivante :

```
docker run -d -p <IP:port-hôte:port-conteneur> --name <nom conteneur> <image> COMMANDES
```

Le paramètre permettant le mappage est **-p <IP:port-hôte:port-container>** :

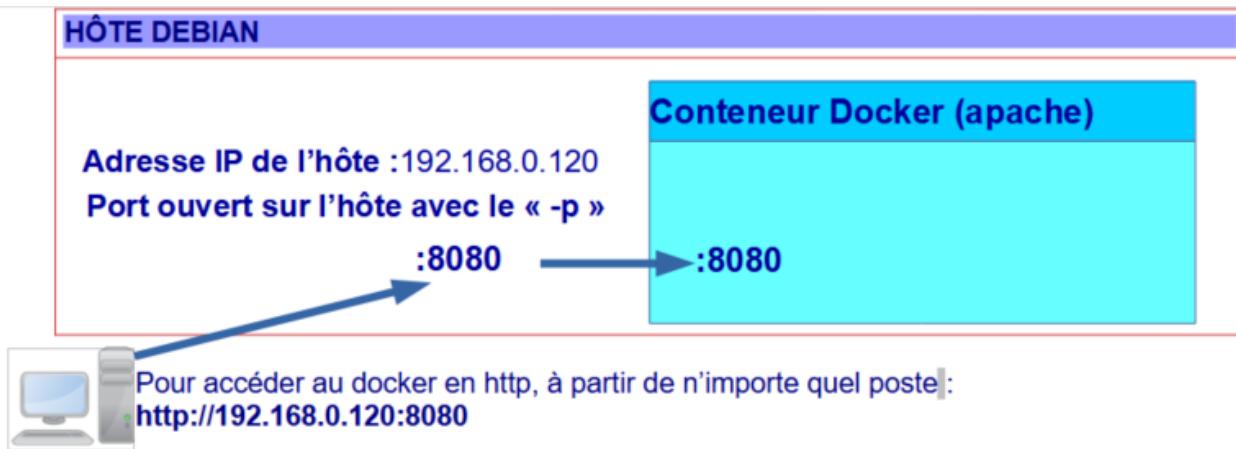
- si l'adresse IP réelle de l'hôte n'est pas indiquée, le système n'écouterai que sur localhost ;
 - si le port de l'hôte à mapper n'est pas indiqué, Docker en choisira un automatiquement.

Le principe est ensuite de lancer le conteneur sera lancé en **arrière plan** (sans laisser de console ouverte) en mode **détaché** : c'est l'option **-d** qui permet cela.

Création d'un conteneur Apache

- **image** : bitnami/apache
 - **ports** :
 - 8080 → http
 - 8443 → https

```
> docker run -d -p 8080:8080 --name servweb bitnami/apache  
02b1a1ec4cd35eaab20930ee9e9dcace4414874ec84811a262aeeee026a73ee5
```



Pour accéder au site par défaut en http à partir de n'importe quel poste :

- <http://192.168.0.120:8080>



Nous pouvons constater via la commande **docker ps** que l'attribut PORT est maintenant rempli :

```
PS C:\Users\Charles> docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS
 NAMES
1f8b8977e68b   bitnami/apache  "/opt/bitnami/script..."  4 minutes ago  Up 4 minutes
0.0.0.0:8080->8080/tcp, 8443/tcp  servweb
...
```

L'option **-p** peut être multiplié autant de fois que nécessaire (si par exemple il y a plusieurs services à exposer dans le conteneur).

Visualisation des **ports ouverts (en écoute) sous Windows** :

- lancez le moniteur de ressources
- Accédez à l'onglet **Réseau** puis sur le bouton **Ports d'écoute**.

Autre solution :

- utiliser la commande netstat

```
netstat -abno | more
```

- utiliser la commande Powershell

```
Get-NetTCPConnection
```

La persistance des données

- Les **images** utilisée par les instances de conteneur sont en **lecture seule**.
- **Toutes les modifications** sur le conteneur sont réalisées **dans une couche supplémentaire**.
- Les **données ajoutées** dans un conteneur **disparaissent avec lui lors de sa destruction**.

Pour **sauvegarder** les modifications dans une autre image, il est nécessaire d'utiliser la commande **commit**.

Cela est pertinent pour construire une nouvelle image personnalisée mais pas pour utiliser le conteneur pour un service en ligne où les données sont modifiées sans cesse ou bien s'il est important de conserver certaines données comme les traces (logs).

Pour un **site Web dynamique** il est nécessaire de disposer :

- d'un **espace de stockage** des **fichiers Web du site** ;
- d'un **espace de stockage** pour la **base de données** ;
- d'un **espace de stockage** pour les logs.

Le mécanisme le plus couramment utilisé avec Docker consiste à utiliser un **stockage externe au conteneur** et donc sur l'hôte ou sur n'importe quel autre support (baies SAN ou NAS) appelé **volume de données**.

Volume de données pour les pages Web

Le paramètre **-v (volume)** de la commande Docker permet :

- de définir un **mappage** entre un **dossier local** du PC et un **dossier du conteneur** ;
- et d'avoir un **stockage qui persistera** même lorsque le conteneur sera supprimé.

De cette manière, une arborescence de fichiers du PC sera montée à l'intérieur du container permettant la publication du site Web utilisant le **serveur Web Apache** et le **moteur de script php**.

- **image utilisée** : php:8.1-apache
- **dossier du serveur Apache** : /var/www/html
 - **dossier local sur le PC** : par exemple C:\www\html
 - Commande sous **Windows**. La commande à utiliser sera la suivante si les fichiers du site Web sont situés dans le dossier local **c:\www\html\siteweb** et les logs du serveur Apache externalisés dans le dossier **c:\www\log**** : `<code> docker run -name servweb -d -v c:\www\html\siteweb:/var/www/html -v c:\www\log:/var/log/apache2 -p 8001:80`
 - **Commande sous Linux** La commande à utiliser sera la suivante si les fichiers du site Web sont situés dans le dossier local **/home/user/www/html/siteweb** et les logs du serveur Apache externalisés dans le dossier **/home/user/www/log** : `<code> docker run -name servweb -d -v /home/user/www/html/siteweb:/var/www/html -v /home/user/www/log:/var/log/apache2 -p 8001:80 php:8.1-apache </code>` Le site Web est accessible à l'URL : **http://localhost:8001** Toute page créée ou modifiée dans le dossier **C:\www\html\siteweb** est publiée dans le site Web.

Pour en savoir plus sur la gestion des volumes Docker :

- <https://www.ionos.fr/digitalguide/serveur/know-how/volumes-de-conteneurs-docker/>

===== Activité à faire =====

Mission 1 :

- Créez un conteneur Docker à partir de l'image **php:8.1-apache** permettant d'accéder à la page Web du site **Geststage**.
- Testez l'accès à la page d'accueil du site Geststage.

Mission 2 : Vous devez **externaliser les logs du serveur Apache** afin de pouvoir les consulter en cas de besoin et d'erreur de fonctionnement du site Web **Geststage** et du serveur.

- Créez un conteneur Docker à partir de l'image **bitnami/apache** affichant la page Web par défaut et en configurant la persistance des logs en local en créant un volume local pour le dossier **/var/log/apache2** du conteneur.
- **Vérifiez la présence** des fichiers de logs dans le **dossier local** (access.log ; error.log ; othervhhostsaccess.log)

Mission 3 :

- Lancez 2 conteneurs Docker (servweb1 et servweb2) à partir de l'image **bitnami/apache**

permettant d'accéder à la page Web du site **Geststage**.

- Testez l'accès à chacun des services Web.

===== Créez un conteneur personnalisé===== L'image **php:8.1-apache** contient une **configuration de php** qui ne possède de pilote que pour gérer les accès à la base de données **sqlite**. Pour pouvoir disposer d'une image contenant le **serveur Web Apache** mais aussi **php** configuré pour communiquer avec la base de données **MariaDB**, vous allez créer une **nouvelle image** à partir de l'image **php:8.1-apache** qui contiendra les pilotes nécessaires. * Créez un fichier appelé **Dockerfile** (sans extension) ; * Editez ce fichier pour y mettre les instructions suivantes : <code file Dockerfile> FROM php:8.1-apache RUN apt-get update && apt-get upgrade -y RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli RUN docker-php-ext-install pdo pdo_mysql </code> * Depuis l'invite de commande **Powershell**, dans le dossier contenant le **fichier Dockerfile**, lancez la commande suivante pour créer : * une image **appelée php et préfixée** par votre prénom : <code powershell> docker build -t charles/php . </code> ===== Afficher les logs du conteneur ===== Si la **persistante des logs en local n'est pas configurée**, cette commande est utile notamment si la création du conteneur se passe mal et si l'accès au service Web n'est pas possible (cette commande ne renvoie rien dans ce cas si tout se passe bien). <code shell> docker logs servweb </code> ===== Mémento Docker =====

Mémento Docker :

<https://www.digitalocean.com/community/tutorials/how-to-remove-docker-images-containers-and-volumes-fr>

===== Retour Accueil Docker ===== * [Docker](#)

From:

/ - **Les cours du BTS SIO**

Permanent link:

</doku.php/reseau/docker/webdocker?rev=1734284728>

Last update: **2024/12/15 18:45**

