



L'utilisation d'un container Docker pour créer une application n'est plus à défendre, tant ceux-ci ont facilité la mise en place et la distribution des plateformes web. Et le Cloud Azure n'est pas en reste, puisqu'il permet de déployer des containers directement dans son **"App Services"**. Mais quant est-il de la connexion à ceux-ci ? Peut-on au moins se connecter sur un container et/ou sur l'application avec le protocole SSH. Et bien la réponse est oui, mais avec quelques limitations.



# Le challenge

Comment créer une application web avec Docker, tout en gardant la possibilité d'accéder au serveur avec le protocole SSH.

Et dans un deuxième temps, déployer cette même application sur le cloud Azure tout en gardant les mêmes options.

# Création du container web

La première tâche à réaliser est de créer le serveur web qui contient l'application. Celle-ci reste très simple pour la démonstration.

- Créez un dossier "web\_server".
- Créez un dossier source à l'intérieur de celui-ci.
- Créer un fichier **"dockerfile"**.
- Dans le dossier source, créez un fichier "index.php".

Le fichier **"index.php"** ne contient que quelques lignes, juste pour indiquer que l'application web fonctionne.

Nom	Modifié le 27.06.2022 09:05	php<br echo "Serveur WEB";?>
b dockerfile	27.06.2022 13:19	

Le fichier "dockerfile" n'est guère plus compliqué. Il va effectuer les opérations suivantes :

- Installer un serveur "Apache" en version 7.4.
- Copier le contenu du dossier source, à savoir le fichier **"index.php"** dans la structure web du serveur (/var/www/html).



• Ouvrir le port 80 pour permettre le trafic web.



Créez l'image du container avec la commande ci-dessous.

docker build -t server\_web .

Voilà, le container web est terminé. On peut le tester en local avec la commande ci-dessous :

docker run -d -p 80:80 --name server\_web server\_web

puis ouvrir un navigateur et entrez l'adresse URL **"localhost"**. L'expression **"Serveur WEB"** devrait s'afficher.

## Création du container SSH

Deuxième étape, la création du container SSH. En effet, avec les containers, il est recommandé de séparer les services. Donc, un container pour l'application web et un autre pour le service SSH.

- Créez un dossier "ssh\_server" à côté du dossier "web\_server".
- Créez un fichier "dockerfile" à l'intérieur de celui-ci.
- Copiez le contenu du "dockerfile" ci-dessous et sauvegardez.

Voici les tâches effectuées :

- Installation d'un serveur Debian et mise à jour.
- Installation d'un serveur SSH.
- Installation de différents outils (ping, nslookup, telnet, vim, azure-cli).
- Paramètres du serveur SSH (Port, Authorisation de connexion, méthode d'encryption, ...).



- Modification du mot de passe du compte "root".
- Génération des clés du serveur
- Ouverture du port 2222 sur le serveur.
- Démarrage du service SSH.

```
FROM debian:latest
# Install Debian server with some useful tools
RUN apt-get update -y
RUN apt-get install openssh-server -y
RUN apt-get install iputils-ping net-tools -y
RUN apt-get install telnet -y
RUN apt-get install dnsutils -y
RUN apt-get install vim -y
RUN apt-get install git -y
RUN apt-get install zip -y
RUN apt-get install unzip -y
RUN apt-get install azure-cli -y
# Create OpenSSH settings file
RUN echo "PasswordAuthentication yes" > /etc/ssh/sshd_config
RUN echo "PermitEmptyPasswords
                                          no" >> /etc/ssh/sshd_config
RUN echo "PermitRootLogin
                                          yes" >> /etc/ssh/sshd_config
                                         2222" >> /etc/ssh/sshd_config
0.0.0.0" >> /etc/ssh/sshd_config
RUN echo "Port
RUN echo "ListenAddress
RUN echo "LoginGraceTime
                                         180" >> /etc/ssh/sshd_config
                                         yes" >> /etc/ssh/sshd_config
RUN echo "X11Forwarding
RUN echo "Ciphers
                                          aes128-cbc, 3des-cbc, aes256-cbc, aes128-ctr, aes192-ctr, aes256
RUN echo "MACs
RUN echo "StrictModes
                                         hmac-shal,hmac-shal-96" >> /etc/ssh/sshd_config
                                          yes" >> /etc/ssh/sshd_config
                                        DAEMON" >> /etc/ssh/sshd_config
RUN echo "SyslogFacility
RUN echo "Subsystem
                                          sftp internal-sftp" >> /etc/ssh/sshd_config
# Set root password
RUN echo "root:Docker!" | chpasswd
# Generate some keys
RUN ssh-keygen -A
# Open port (metadata only)
EXPOSE 2222/tcp
# Start services
RUN service ssh startCMD ["/usr/sbin/sshd","-D"]
```

Créez l'image du container avec la commande ci-dessous.

docker build -t server\_ssh .

Voilà, le container web est terminé. On peut le tester en local avec la commande ci-dessous :

docker run -d -p 22:2222 --name server\_ssh server\_ssh

puis ouvrez une fenêtre de commande et entrez la commande **"ssh root@localhost"**. Acceptez la clé proposé puis entrez le mot de passe qui est **"Docker!"**. La connexion SSH devrait s'établir.



### Utilisation de l'orchestrateur

Pour créer l'illusion que l'on va regrouper nos deux containers en un seul, on va utiliser l'orchestrateur de Docker, le **"docker-compose"**.

- Créez un fichier "docker-compose.yml" au même niveau que les deux dossiers "web\_server" et "ssh\_server".
- Copiez le contenu du fichier ci-dessous.

Voici les tâches effectuées par l'orchestrateur :

- Création d'un serveur Web
- Le nom du container est "server\_web".
- Le nom complet (FQDN) du serveur est "server\_web.fuyens.ch".
- L'image docker utilisé est "server\_web".
- Le port local est le 80. Le port du serveur est le 80.

Idem pour le serveur SSH, à l'exception du port qui devient le 22 en local et le 2222 sur le serveur.



version: '3'
services:
web_server:
container_name: server_web
hostname: server_web.fuyens.ch
image: server_web
ports:
- 80:80
ssh_server:
container_name: server_ssh
hostname: server_ssh.fuyens.ch
image: server_ssh
ports: - 22:2222

Créez l'image des deux containers avec la commande ci-dessous :

docker-compose up

Le log de la commande "docker-compose" qui affiche bien que les deux serveurs sont créées.

```
PS C:\Users\olivi\Desktop\projet_ssh> docker-compose up
[+] Running 2/2
Container server_ssh Created
Container server_web Created
Attaching to server_ssh, server_web
server_web | [Tue Jun 28 09:47:35.579871 2022] [mpm_prefork:notice] [pid 1] AH00163: Apache/2
server_web | [Tue Jun 28 09:47:35.580013 2022] [core:notice] [pid 1] AH00094: Command line:
```

Voilà, l'orchestrateur a créé les deux containers web et ssh séparément. Comme dans les exemples cidessus, on peut se connecter sur le serveur web et sur le serveur ssh.

## Création d'une application sur Azure

Il ne reste plus qu'à essayer de pousser les containers vers Azure et de retenter l'expérience.

La surprise est de taille, mais Microsoft Azure, qui accepte pourtant de créer des applications avec de multiples containers, ne permet pas d'utiliser ceux-ci pour séparer le serveur SSH du serveur Web. Il va falloir **"Assembler les deux containers en un seul"** pour y remédier. Aux dernières nouvelles, Microsoft devrait corriger ceci, pour que l'utilisation d'un seul container soit possible.



### Création d'un container offrant les deux services

- Créez un nouveau dossier "all\_server".
- Copiez le dossier source contenant le fichier "index.php".
- Créez un nouveau fichier "dockerfile" et copiez le code ci-dessous.
- Créez un fichier "start.sh" qui va permettre de démarrer les deux services du container.

Bureau > projet_ssh > all_server	
Nom source dockerfile	#!/bin/sh service ssh start /usr/sbin/sshd /usr/sbin/apache2ctl -D FOREGROUND
start.sh	

La structure du nouveau container et le script de démarrage des services





# Open port (metadata only) EXPOSE 2222/tcp EXPOSE 80/tcp # Start services COPY start.sh /start.sh RUN chmod 777 /start.shCMD /start.sh

Créez l'image du container avec la commande ci-dessous.

#### docker build -t server\_all .

Voilà, le nouveau container offrant les deux services est disponible. Il ne reste plus qu'à le pousser vers le Cloud Azure et créer une application.

### Le registre de containers du Cloud Azure

- Ouvrez la console Azure et connectez-vous.
- Créez un "Azure Container Registry" et nommez-le. Dans l'exemple ci-dessous, il se nomme "acrwesteu001".
- Depuis VSCode, connectez-vous au registre.

az acr login --name acrwesteu001

Créez le tag du container **"server\_all"**, puis poussez-le vers le registre du Cloud Azure.

docker tag server\_web acrwesteu001.azurecr.io/server\_all:1.0
docker push acrwesteu001.azurecr.io/server\_all:1.0

Vérifiez que le container se trouve bien dans le registre.

acrwesteu001   Re     Container registry	epositories 🛧 …
♀ Search (Ctrl+/)	« 🕐 Refresh
Overview	Search to filter repositories
Activity log	Repositories ↑↓
R Access control (IAM)	server_all
Tags	
Quick start	

### Le service applications du Cloud Azure

- Créez une application avec "App Services".
- Dans la section "Docker", sélectionnez l'option "Single Container".
- Dans le fichier de configuration, sélectionnez tour à tour le fichier image "server\_all".
- Dans les deux cas, sélectionnez la version "1.0".
- Sauvegardez.



Basics <b>Docker</b> Networkin	ng (preview) Monitoring Tags Review + create	
Pull container images from Azure he containerized app with your p	Container Registry, Docker Hub or a private Docker repository. App referred dependencies to production in seconds.	Service will deploy
Options	Single Container	$\sim$
mage Source	Azure Container Registry	$\sim$
Azure container registry optio	ns	
Registry *	acrwesteu001	$\sim$
mage *	server_all	$\sim$
āg *	1.0	~
itartup Command ①	La création d'une application à l'aide d'un container	
Startup Command ① nectez-vous au serveur we tps://[nom de l'applicati	La création d'une application à l'aide d'un container b avec l'URL: .on].azurewebsites.net	
Startup Command ①	La création d'une application à l'aide d'un container b avec l'URL : Lon].azurewebsites.net rend du temps. Pas de panique !	
Startup Command ①	La création d'une application à l'aide d'un container b avec l'URL : Lon].azurewebsites.net rend du temps. Pas de panique !	
Startup Command ①  Inectez-vous au serveur we  tps://[nom de l'applicati La première connection pr but fonctionne bien, le mess service SSH	La création d'une application à l'aide d'un container b avec l'URL : con].azurewebsites.net rend du temps. Pas de panique ! sage <b>"Serveur WEB"</b> devrait s'afficher.	



#### Retournez dans VSCode et entrez la commande Azure ci-dessous

az webapp create-remote-connection -- subscription [id] -- resource-group [name of the resource-group] -- name [name of the application]

Le terminal devrait retourner un port de connexion

```
Verifying if app is running....
App is running. Trying to establish tunnel connection...
Opening tunnel on port: 50818
SSH is available { username: root, password: Docker! } Ctrl + C to close
```

Il est maintenant possible de se connecter en utilisant la commande ci-dessous :

ssh root@localhost -p 50818

```
C:\Users\olivi>ssh root@localhost -p 50818
root@localhost's password:
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@cla48fdd6ef8:~#
```

### Conclusion

Voilà un petit exercice auquel vous n'échapperez pas si vous rencontrez des problèmes avec votre application. Alors, plutôt que de tatonner à la recherche de l'erreur, de repousser 50x le container vers le cloud Azure afin de trouver le problème, une petite connexion SSH est bien pratique.

Il est a noté également qu'il est possible de se connecter en SSH directement depuis la console Azure en sélectionnant **"l'onglet SSH"** depuis l'application.