Docker - SGBD bis

PROHACKTIVE

Sommaire

1. Description

A des fins de simplification et de polyvalence, la méthode utilisée pour la génération de conteneur a été revue. La création d'un conteneur se fera via un interface web où l'utilisateur pourra renseigner ses paramètres. Elle contiendra également un tableau listant les conteneurs actifs ainsi que la possibilité de les supprimer.

2. Pré-requis

Une VM debian 11 servira d'hôte aux conteneurs Dockers.
Pour cette tâche, on installe au préalable les services requis:

Docker et docker-compose:
Installation des paquet nécessaire pour permettre à apt d'utiliser des dépots par https:

```
# apt install ca-certificates curl gnupg lsb-release
```

On ajoute la clé GPG de docker:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

On ajoute les dépôts:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture)
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

On met à jour la liste des paquets et on install docker:

```
# apt update
# apt install docker-ce docker-ce-cli containerd.io
```

Installation de docker-compose:

```
# curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-compo
se-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
# chmod +x /usr/local/bin/docker-compose
# ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Serveur web nginx:
Afin de lancer la création des différents conteneurs, le service nginx est installé.

```
# apt install nginx
```

l'installation de php est requise afin d'exécuter la page web en php.

```
# apt install php-fpm
```

Configuration:

```
server {
	listen 80 default_server;
	listen [::]:80 default_server;
	root /var/www/html;
	index index.php index.html index.htm index.nginx-debian.html;
	location / {
			try_files $uri $uri/ =404;
	}

	location ~ \.php$ {
			include snippets/fastcgi-php.conf;
			fastcgi_pass unix:/run/php/php7.4-fpm.sock;
	}
```

Configurations réseau:

Ip : **172.17.0.77**

fichier /etc/hosts :

```
172.17.0.107 bdd.phk.lan
```

Routage:
Afin de permettre l'accès aux conteneurs via leurs ip:port, on ajoute une règle iptables:

```
iptables -A FORWARD -i ens192 -o bridge1 -j ACCEPT
```

3. Scripts PHP

index.php

```
<!DOCTYPE html>
<html>

	<head>
	<title>BDD Docker</title>
	<meta charset="UTF-8">
	<link href="style.css" rel="stylesheet" type="text/css">
	</head>

	<header>
	<img src="logo/docker.jpg" />
	</header>

	<body>
	<?php include("fonction.php"); ?>
	<h1>BDD Docker</h1>
	<form method="post">
		<div class="lancement">
		<br>
		<h3>Lancement d'un conteneur Docker</h3>
		<label for="bdd">BDD : </label><input type="text" name="bdd"
placeholder="Ex : mysql"/>
		<br>
		<br>
		<label for="version">Version: </label><input type="text"
name="version" placeholder="Ex : 5.4.6"/>
		<br>
		<br>
		<input type="submit" value="Lancement" name="valider">
		<?php lancement(); ?>
		<br>
		</div>
		<br><br>
		<div class="suppression">
		<br>
		<h3>Suppression d'un conteneur Docker</h3>
		<label for="bdd">ID conteneur à supprimer : </label><input
type="text" name="ID"/>
		<br>
		<br>
		<input type="submit" value="Supprimer" name="Suppression">
		<?php suppression(); ?>
		<br>
		</div>
		<br>
		<table>
```

```
            <caption><h3>Conteneurs Docker</h3></caption>
            <thead>
                  <tr>
                        <th>ID</th>
                        <th>Image</th>
                        <th>IP</th>
                        <th>Port</th>
                        <th>Nom</th>
                  </tr>
            </thead>
            <tbody>
                  <?php tableau(); ?>
            </tbody>
            </table>
      </form>
      </body>
</html>
```

fonction.php

```
<?php
function lancement() //Lancement d'un docker non existant
{
      //Si le bouton Lancement a été pressé
      if(isset($_POST['valider']))
      {
                  //Récupération des informations saisies
            $bdd=strtolower($_POST['bdd']);
            $version=$_POST['version'];
                  //Si aucune BDD n'a été saisi, le signaler
            if (empty($bdd))
            {
                        echo "<h5>Aucune BDD n'a été saisi</h5>";
            }
            else
            {
                        $ajout="$";
                        //Si le conteneur n'existe pas déjà, on lance le
script bash de création de conteneur
                        if (empty(shell_exec("docker ps | awk '{print $2}' |
grep -E $bdd:$version$ajout")))
                        {
                              echo (shell_exec("/script/sgbd.sh $bdd
$version"));
                        }
                        //S'il existe, on le fait remarquer
                        else
                        {
```

```
                                echo "<h5>Ce conteneur existe déjà</h5>";
                        }
                }
        }
}

function tableau() //Stockage des données nécessaires pour le tableau
{
        //Récupération des informations dont on a besoin dans un fichier
        $stocke=shell_exec("docker ps --format 'table
{{.ID}}\t{{.Image}}\t{{.Ports}}\t{{.Names}}' | sed '1d' > stock.txt");
        //Comptage des lignes présente dans le fichier
        $nbLigne=shell_exec("wc -l stock.txt");


        //Déclaration du tableau
        $tableau=array();


        //Ouverture du fichier et on le lit ligne par ligne
        $lignes = file('stock.txt');
        foreach ($lignes as $nb => $contenu)
        {
                //On sépare la ligne traité du reste du fichier
                file_put_contents('ligne.txt', "$contenu");
                $conteneur=shell_exec("cat ligne.txt | awk '{print $1}'");
                        //On entre les données nécessaires dans le tableau
                array_push($tableau, array(shell_exec("cat ligne.txt | awk '{print
$1}'"), shell_exec("cat ligne.txt | awk '{print $2}'"), shell_exec("docker
inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
$conteneur"), shell_exec("cat ligne.txt | awk '{print $3}'"), shell_exec("cat
ligne.txt | awk '{print $4}'")));
        }
        //Après lecture complète du fichier, on supprime le fichier qui traité
les lignes une par une
        shell_exec('rm ligne.txt');


        //On affiche le contenu de notre tableau
        for ($i=0; $i<=$nbLigne-1; $i++) {
                echo "<tr>";
                                echo"<td>" . $tableau[$i][0] . "</td>";
                                echo"<td>" . $tableau[$i][1] . "</td>";
                                echo"<td>" . $tableau[$i][2] . "</td>";
                                echo"<td>" . $tableau[$i][3] . "</td>";
                                echo"<td>" . $tableau[$i][4] . "</td>";
                echo "</tr>";
        }
}
```

```php
function suppression() //Supression d'un conteneur
{
	//Si le bouton Supprimer a été pressé
	if(isset($_POST['Suppression']))
	{
			//Récupération de l'ID saisi
		$ID=$_POST['ID'];
			//S'il est vide, on le fait remarquer
		if (empty($ID))
		{
				echo "<h5>Aucun ID n'a été saisi</h5>";
		}
		else
		{
				//Si le conteneur n'existe pas, on le fait remarquer
				if(empty(shell_exec("docker ps | grep -w $ID")))
				{
					echo "<h5>Ce conteneur n'existe pas</h5>";
				}
				//Sinon on le supprime
				else
				{
					shell_exec("docker kill $ID");
				}
		}
	}
}
?>
```

4. Script Bash

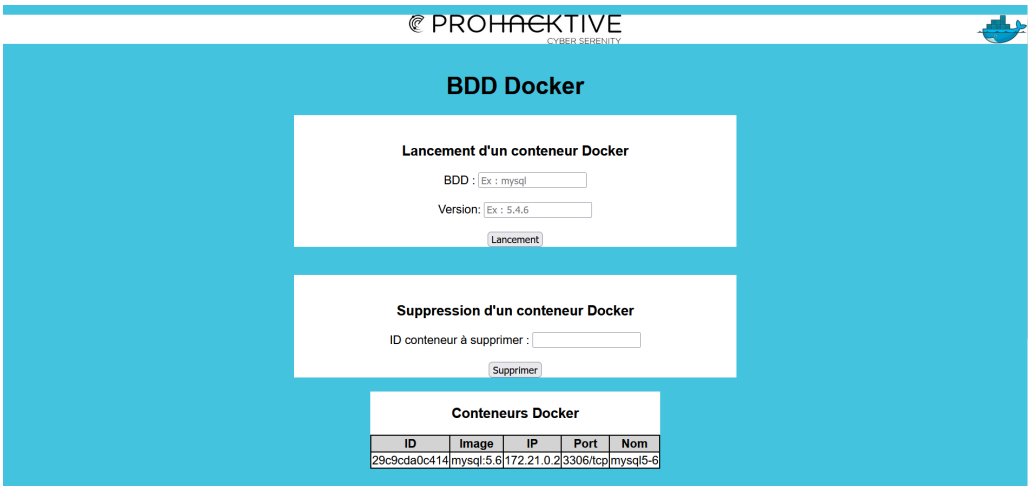
```bash
#!/bin/bash
#
#Syntaxe : ./sgbd.sh <nomImage> <version>
#

#
########Codes erreurs########
# 1 : version ou image introuvable
# 2 : Le numéro de version ne peux pas être définie automatiquement
# 4 : La création du conteneur a échoué (image non reconnue; conteneur existant; ...)
```

```
#Initialisation des variables
image=$1
versionTag=$2
version=""
let out=0

#Vérification des entrées
[[ $image == "" ]] && echo "Requires at least 1 argument." && exit 1 ||
image=$image
[[ $versionTag == "" ]] && versionTag="latest" || versionTag=$versionTag

if [[ $versionTag == "latest" ]]; then
    if [[ `docker pull $image:$versionTag 2> /dev/null` ]]; then
       if [[ $image == "mariadb" ]]; then
             version=$(docker image inspect $image:$versionTag | grep
${image^^}_VERSION | awk 'END {print}' | awk -F'[:]+'
'/MARIADB_VERSION=1:/{print $2}' | awk -F'+' '{print$1}')
       else
             version=$(docker image inspect $image:$versionTag | grep
${image^^}_VERSION | awk 'END {print}' | awk -F'=' '{print$2}' | awk
-F'"' '{print$1}')
       fi

       if [[ $version == "" ]];then
             #Le numéro de version ne peux pas être définit
automatiquement
             let out=2
             echo $out
             exit 1
       fi
    else
       #version ou image introuvable
       let out=1
       echo $out
       exit 1
    fi
else
    version=$versionTag
fi

#Création du nom du conteneur
name=$image${version//./-}


#Configuration en fonction de l'image
```

```
case $image in
    mariadb)
       option="-e MARIADB_ROOT_PASSWORD=root -e MARIADB_USER=user -e
MARIADB_PASSWORD=root"
    ;;

    mongo)
       option="-e MONGO_INITDB_ROOT_USERNAME=root -e
MONGO_INITDB_ROOT_PASSWORD=root"
    ;;

    postgres)
       option="-e POSTGRES_USER=root -e POSTGRES_PASSWORD=root"
    ;;

    *)
       option="-e ${image^^}_ROOT_PASSWORD=root"
    ;;
esac

#Création du conteneur
if [[ `docker run --rm -dti --net bridge1 --name $name $option
$image:$version 2> /dev/null` ]]; then

    #Waiting for the container to up ...
    sleep 10

    #Récupération de l'ip du conteneurs
    #IP=`docker container inspect $name | grep '"IPAddress": "' | sed
'1d' | awk -F'[:]+' '/IPAddress/{print $2}' | awk -F'["]+' '{print $2}'`


else
    #La création du conteneur a échoué (image non reconnue; conteneur
existant; ...)
      let out=$out+4
    echo $out
    exit 1
fi

echo $out
```

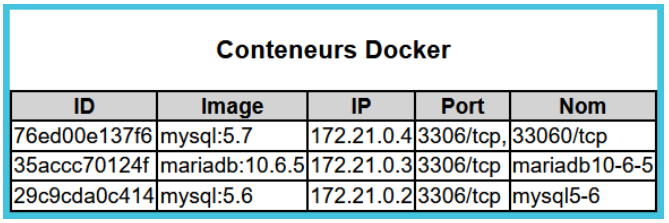
5. Recette

Rendu du php :



Lancement d'un conteneur mysql:5.7 :

Il a bien été lancé et ajouter au tableau :



Conteneurs Docker

ID	Image	IP	Port	Nom
76ed00e137f6	mysql:5.7	172.21.0.4	3306/tcp,	33060/tcp
35accc70124f	mariadb:10.6.5	172.21.0.3	3306/tcp	mariadb10-6-5
29c9cda0c414	mysql:5.6	172.21.0.2	3306/tcp	mysql5-6

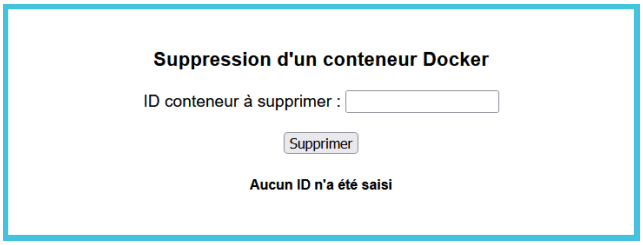
Le conteneur a bien été créé :

```
root@bdd:/var/www/html# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
76ed00e137f6   mysql:5.7      "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes    3306/tcp, 33060/tcp    mysql5-7
35accc70124f   mariadb:10.6.5 "docker-entrypoint.s…"   9 minutes ago    Up 9 minutes    3306/tcp               mariadb10-6-5
29c9cda0c414   mysql:5.6      "docker-entrypoint.s…"   25 minutes ago   Up 25 minutes   3306/tcp               mysql5-6
```

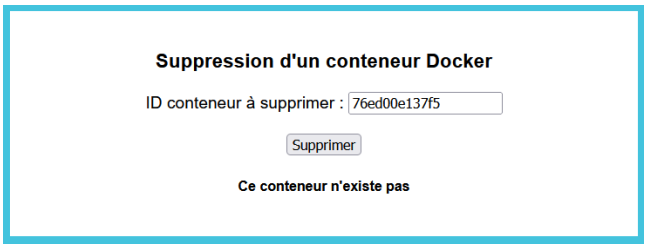
Lancement d'un conteneur déjà existant :

Lancement sans avoir renseigné de service (si on ne renseigne pas de version, la plus récente sera choisi) :

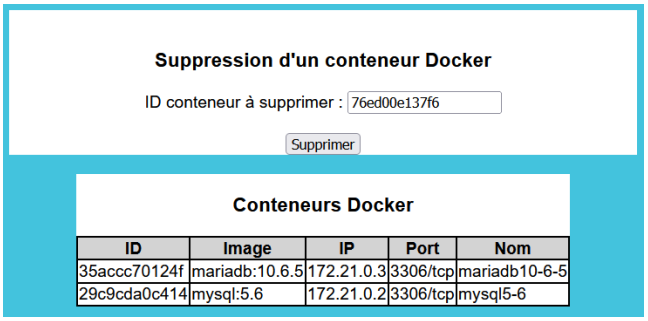
Suppression sans avoir renseigné d'ID :



Suppression d'un conteneur inexistant :



Suppression d'un conteneur :



Le conteneur a bien été supprimé :

```
root@bdd:/var/www/html# docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS      NAMES
35accc70124f   mariadb:10.6.5  "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes   3306/tcp   mariadb10-6-5
29c9cda0c414   mysql:5.6       "docker-entrypoint.s…"   29 minutes ago   Up 29 minutes   3306/tcp   mysql5-6
```

Les conteneurs sont bien accessible à l'extérieur de l'hôte: