

Corentin Froger
Antoine Fontanez
IL 2

Compte-Rendu Virtualisation avancée

https://github.com/Corentin-Froger/Virtualisation_Avancee

TP 1.....	3
TP 2.....	5
TP 3.....	6
TP 4.....	7
TP 5.....	10
TP 6.....	11

TP 1

Etape 1 : Installer docker desktop

<https://www.docker.com/get-started/>

Etape 2 : Télécharger un ou plusieurs exemples depuis

<https://github.com/docker/awesome-compose>

dans notre cas, on prendra l'exemple de minecraft, car étant le plus attrayant

Etape 3 : Se déplacer à la racine du fichier minecraft et lancer

```
docker compose up -d
```

Toute information utile peut se trouver dans le fichier README.md fourni à l'intérieur pour voir

Etape 4 : Vérifier que tout fonctionne correctement

```
docker ps
```

```
PS C:\Users\CORENTIN\Documents\TRUCS PERSONNELS\IUT\Semestre 5\R5.09 - Virtualisation avancée\minecraft> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
483590a00a94   itzg/minecraft-server   "/start"              11 minutes ago    Up 11 minutes (healthy)    0.0.0.0:25565->25565/tcp    minecraft-minecraft-1
```

Le processus tourne bien

```
docker compose logs
```

```
minecraft-1 | [12:38:38] [ServerMain/INFO]: No existing world data, creating new world
minecraft-1 | [12:38:39] [ServerMain/INFO]: Loaded 1337 recipes
minecraft-1 | [12:38:39] [ServerMain/INFO]: Loaded 1448 advancements
minecraft-1 | [12:38:40] [Server thread/INFO]: Starting minecraft server version 1.21.3
minecraft-1 | [12:38:40] [Server thread/INFO]: Loading properties
minecraft-1 | [12:38:40] [Server thread/INFO]: Default game type: SURVIVAL
minecraft-1 | [12:38:40] [Server thread/INFO]: Generating keypair
minecraft-1 | [12:38:40] [Server thread/INFO]: Starting Minecraft server on *:25565
minecraft-1 | [12:38:40] [Server thread/INFO]: Using epoll channel type
minecraft-1 | [12:38:40] [Server thread/INFO]: Preparing level "world"
minecraft-1 | [12:38:43] [Server thread/INFO]: Preparing start region for dimension minecraft:overworld
minecraft-1 | [12:38:43] [Worker-Main-4/INFO]: Preparing spawn area: 2%
minecraft-1 | [12:38:44] [Worker-Main-5/INFO]: Preparing spawn area: 2%
minecraft-1 | [12:38:44] [Worker-Main-13/INFO]: Preparing spawn area: 18%
minecraft-1 | [12:38:45] [Worker-Main-10/INFO]: Preparing spawn area: 20%
minecraft-1 | [12:38:45] [Worker-Main-7/INFO]: Preparing spawn area: 51%
minecraft-1 | [12:38:46] [Server thread/INFO]: Time elapsed: 2274 ms
minecraft-1 | [12:38:46] [Server thread/INFO]: Done (5.468s)! For help, type "help"
minecraft-1 | [12:38:46] [Server thread/INFO]: Starting remote control listener
minecraft-1 | [12:38:46] [Server thread/INFO]: Thread RCON Listener started
minecraft-1 | [12:38:46] [Server thread/INFO]: RCON running on 0.0.0.0:25575
minecraft-1 | [12:39:45] [Server thread/INFO]: Server empty for 60 seconds, pausing
```

On peut voir que le serveur s'est bien démarré normalement, on peut même s'y connecter directement

```
docker compose down -v
```

Pour arrêter le serveur, l'option -v supprime les données

TP 2

Pour chacun des types d'isolation suivants : Mount, UTS, IPC, PID, user, time.

- Isoler 2 processus bash différents avec unshare
- Trouver la commande prouvant que les 2 processus sont bien isolés

Exemple :

```
## Terminal 1
### start namespaced process
$ unshare --fork --pid --mount-proc bash

### launch long running process
root@namespace:~# perl -MPOSIX -e '$0="hello"; pause' &

### check process running
root@namespace:~# ps aux

## Terminal 2
$ unshare --fork --pid --mount-proc ps
```

Les options suivantes permettent à 2 processus d'être séparés dans différents namespace selon :

-mount : Les fichiers systèmes
-uts : UTS (UNIX Time-Sharing) permet d'avoir différents noms d'hôte et de domaine
-ipc : (Inter-Process Communication) la communication entre les processus, comme POSIX, System V
-pid : Les numéros de processus,
-user : UIDs et GIDs, sépare différents utilisateurs entre eux
-time : Permet à 2 processus d'avoir 2 heures système différentes

-fork : faire un fork du programme plutôt que l'exécuter directement
-mount-proc <nom programme> : Avant de lancer le programme, monte le processus fichier système au point de montage

TP 3

cgcreate : créer un nouveau groupe de contrôle

cgexec : exécute la commande fournie dans un groupe de contrôle donné

cgcreate -g *:test a:b

create control group student in all mounted hierarchies and create control group teacher in hierarchy containing controller devices.

cgexec -g *:test ls

runs command ls in control group test1 in all mounted controllers.

TP 4

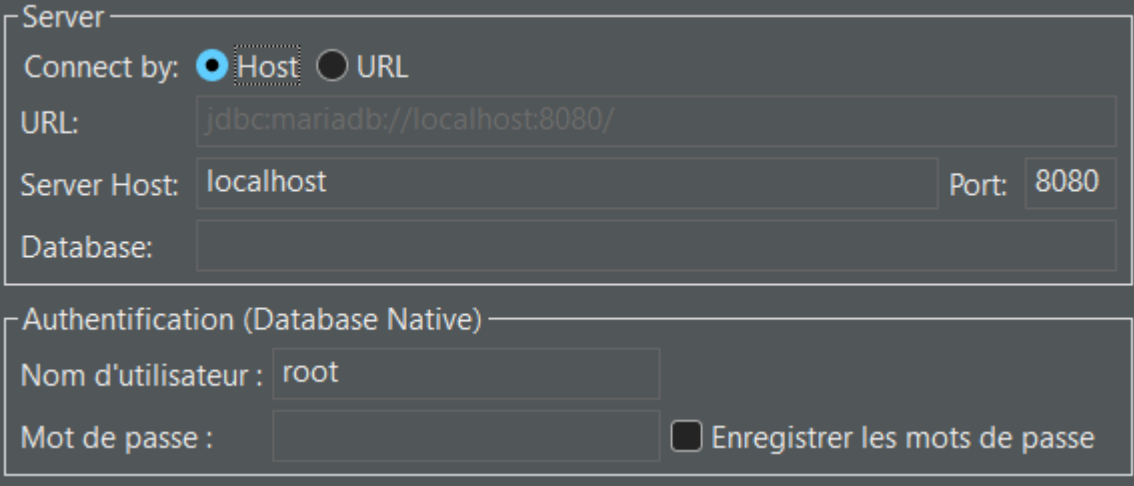
Etape 1 : `docker pull mariadb`

Etape 2 : exécuter mariaDB, nous n'avons pas besoin de mot de passe pour ce TP

bien faire pointer le port 8080 de la machine vers le port 3306 du container avec -p

```
docker run -p 8080:3306 --detach --name serveur_maria --env  
MARIADB_ALLOW_EMPTY_ROOT_PASSWORD=1 mariadb:latest
```

Etape 3 : Tester la connection avec DBeaver, bien préciser qu'il s'agit de MariaBD et qu'on se connecte au port 8080



The screenshot shows the DBeaver connection configuration window. Under the 'Server' tab, 'Connect by:' is set to 'Host'. The 'URL' field contains 'jdbc:mariadb://localhost:8080/'. The 'Server Host' field contains 'localhost' and the 'Port' field contains '8080'. The 'Database' field is empty. Under the 'Authentification (Database Native)' tab, the 'Nom d'utilisateur' field contains 'root' and the 'Mot de passe' field is empty. There is a checkbox labeled 'Enregistrer les mots de passe' which is unchecked.

Etape 4 : Créer une table







```
USE sys;  
  
create table projects(  
    project_id int auto increment,  
    project_name varchar(255) not null,  
    begin_date date,  
    end_date date,  
    cost decimal(15,2) not null,  
    created_at timestamp default current_timestamp,  
    primary key(project_id)  
);
```

Il faut bien préciser d'utiliser **sys** comme base de données

`docker ps` indique ceci :

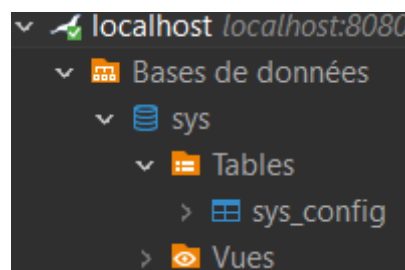
```
PS C:\Users\CORENTIN\Documents\TRUCS PERSONNELS\IUT\Semestre 5\R5.09 - Virtualisation avancée> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
4895c3d89611   mariadb:latest "docker-entrypoint.s..." 5 minutes ago  Up 5 minutes  0.0.0.0:8080->3306/tcp   serveur_maria
PS C:\Users\CORENTIN\Documents\TRUCS PERSONNELS\IUT\Semestre 5\R5.09 - Virtualisation avancée> |
```

La table créée :

Nom de la colonne	#	Type de donnée
 project_id	1	int(11)
 project_name	2	varchar(255)
 begin_date	3	date
 end_date	4	date
 cost	5	decimal(15,2)
 created_at	6	timestamp

Etape 5 : On stoppe le container, puis on le supprime (on peut utiliser `docker stop` et `docker rm` ou simplement utiliser docker desktop)

La table n'est plus présente (il devrait y avoir `projects` en dessous de `sys_config`)



Etape 6 :

Il faut donc créer un volume avec -v : `-v /maria_tp:/var/lib/mysql:Z`

```
docker run -p 8080:3306 -v /maria_tp:/var/lib/mysql:Z --detach
--name serveur_maria --env MARIADB_ALLOW_EMPTY_ROOT_PASSWORD=1
mariadb:latest
```


On remarque effectivement que les données sont bien stockées

```
PS C:\Users\CORENTIN> docker volume ls
DRIVER      VOLUME NAME
local       2aa38177af67d68fce12bf47fbb7213350287474a0ed91f08f234b664add1ae9
```

Etape 7 : On fait une variable d'environnement pour se connecter avec un mot de passe

ajouter `-e MARIADB_ROOT_PASSWORD=1234` (On a pas besoin d'un mdp complexe)

```
docker run -p 8080:3306 -v /maria_tp:/var/lib/mysql:Z --detach
--name serveur_maria --env MARIADB_ALLOW_EMPTY_ROOT_PASSWORD=1
mariadb:latest
```

TP 5

- 1) Création d'un fichier index.php affichant les informations sur php :

```
<?php
phpinfo();
?>
```

- 2) On crée un fichier dockerfile qui nous permettra de créer notre image ! :

```
FROM ubuntu:latest
RUN apt-get update
RUN apt install -y apache2
RUN apt install -y php
RUN apt clean
COPY /index.php /var/www/html/index.php
EXPOSE 8000
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Ce dockerfile installe apache2 et php, puis on copie le fichier index.php créé précédemment pour le mettre dans un répertoire accessible.

- 3) Création du build de l'image avec le dockerfile :

docker build -t monimage:latest .

- 4) Run de l'image dans un container qu'on nomme "myapache" sur le port 8000 :

docker run --name myapache -d -p 8000:80 monimage:latest

Ainsi, on peut maintenant accéder à apache2 sur localhost:8000, et on a notre fichier affichant les informations php sur localhost:8000/index.php

PHP Version 8.3.6	
	
System	Linux c731d8f674fc 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x86_64
Build Date	Sep 30 2024 15:17:17
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d
Additional .ini files parsed	/etc/php/8.3/apache2/conf.d/10-opcache.ini, /etc/php/8.3/apache2/conf.d/10-pdo.ini, /etc/php/8.3/apache2/conf.d/20-calendar.ini, /etc/php/8.3/apache2/conf.d/20-ctype.ini, /etc/php/8.3/apache2/conf.d/20-exif.ini, /etc/php/8.3/apache2/conf.d/20-ffi.ini, /etc/php/8.3/apache2/conf.d/20-fileinfo.ini, /etc/php/8.3/apache2/conf.d/20-ftp.ini, /etc/php/8.3/apache2/conf.d/20-gettext.ini, /etc/php/8.3/apache2/conf.d/20-iconv.ini, /etc/php/8.3/apache2/conf.d/20-phar.ini, /etc/php/8.3/apache2/conf.d/20-posix.ini, /etc/php/8.3/apache2/conf.d/20-readline.ini, /etc/php/8.3/apache2/conf.d/20-shmop.ini, /etc/php/8.3/apache2/conf.d/20-sockets.ini, /etc/php/8.3/apache2/conf.d/20-sysmsg.ini, /etc/php/8.3/apache2/conf.d/20-sysvsem.ini, /etc/php/8.3/apache2/conf.d/20-sysvshm.ini, /etc/php/8.3/apache2/conf.d/20-tokenizer.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API20230831,NTS
PHP Extension Build	API20230831,NTS
Debug Build	no

TP 6

On créer un fichier compose.yml et on le lance avec `docker compose up -d`

```
services:
  frontend:
    build:
      dockerfile: apache/Dockerfile
    container_name: frontend
    ports:
      - 80:80

  adminer:
    image: adminer:latest
    container_name: adminer
    restart: always
    ports:
      - 8080:8080

  database:
    image: mariadb:latest
    container_name: database
    restart: always
    environment:
      MARIADB_ROOT_PASSWORD: root
    ports:
      - 3306:3306
    volumes:
      - volume_db:/data/db

volumes:
  volume_db:
    name: "volume_db"

#networks:
# network1:
#   # config
#network2:
#   # config
```

Les images et containers sont ceux ci avec leurs ports

<input type="checkbox"/>	Name	Container ID	Image	Port(s)
<input type="checkbox"/>	tp6	-	-	-
<input type="checkbox"/>	frontend	cc3834772851	tp6-frontend	80:80 ↗
<input type="checkbox"/>	database	96253bc19880	mariadb:latest	3306:3306 ↗
<input type="checkbox"/>	adminer	1def6ce0b45b	adminer:latest	8080:8080 ↗

La base de données est mariaDB sur le port 3306, on lui donne un volume comme au TP 4

Adminer permet d'accéder à une base de données en allant sur le port 8080 du localhost :

← → ↻

localhost:8080/?server=database

Langue: Français ▼

Adminer 4.8.1

(MySQL) root@database - database

Authentification

Système	MySQL ▼
Serveur	database
Utilisateur	root
Mot de passe	••••
Base de données	

☒ Authentification ☐ Authentification permanente

dans compose.yaml, on a précisé que le nom du serveur mariaDB est database et que le mot de passe est root, on utilise ici l'utilisateur par défaut root

← → ↻ localhost:8080/?server=database&username=root

Langue: Français MySQL » database

Adminer 4.8.1

DB:

[Requête SQL](#) [Importer](#) [Exporter](#)

Sélectionner la base de données

[Créer une base de données](#) [Privilèges](#) [Liste des processus](#) [Variables](#) [Statut](#)

Version de MySQL : **11.6.2-MariaDB-ubu2404** via l'extension PHP **MySQLi**

Authentifié en tant que : **root@172.18.0.3**

	Base de données - Rafraichir	Interclassement	Tables	Taille - Calcul
<input type="checkbox"/>	information_schema	utf8mb3_general_ci	?	?
<input type="checkbox"/>	mysql	utf8mb4_uca1400_ai_ci	?	?
<input type="checkbox"/>	performance_schema	utf8mb3_general_ci	?	?
<input type="checkbox"/>	sys	utf8mb3_general_ci	?	?

Sélectionnée(s) (0)

[Supprimer](#)

On peut voir que l'on peut bien accéder à la base de données

Le serveur est créé grâce à ce fichier Dockerfile, comme dans le TP 5 :

```
FROM ubuntu:latest
RUN apt-get update
RUN apt install -y apache2
RUN apt install -y php
RUN apt clean
COPY index.php /var/www/html/index.php
EXPOSE 8000
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Le script PHP du serveur :

```
<?php

$databaseHost = $_GET["server"];
$databaseUsername = $_GET["user"];
$databasePassword = "root";
$databaseName = $_GET["dbname"];
$port = $_GET["port"];

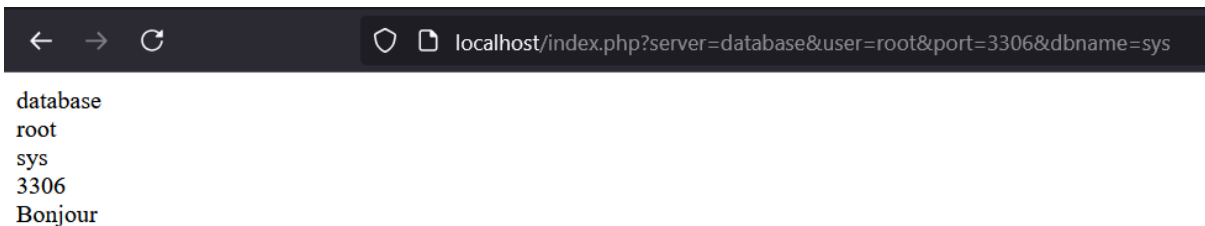
echo $databaseHost . "<br>";
echo $databaseUsername . "<br>";
echo $databaseName . "<br>";
echo $port . "<br>";

// Connect to the database
echo "Bonjour";

$mysqli = mysqli_connect($databaseHost, $databaseUsername, $databasePassword, $databaseName, $port);

if ($mysqli -> connect_errno) {
    echo "Failed to connect to MySQL: " . $mysqli -> connect_error;
    exit();
}
echo "connected";
?>
```

Nous n'avons malheureusement pas réussi à se connecter à la base de données créée grâce à notre propre script PHP, mais nous avons vu que nous l'avons fait avec adminer



```
← → ↻ localhost/index.php?server=database&user=root&port=3306&dbname=sys

database
root
sys
3306
Bonjour
```