Install docker :

Documentation by BASTOS Quentin

1. **Remove any incomplete installations**: To clean up the partially installed Docker Desktop, run:

sudo dpkg --remove --force-remove-reinstreq docker-desktop

2. **Set up the Docker repository**: Ensure that you have the Docker repository added to your APT sources. You can do this by following these steps:
    - **Install Required Packages**: First, make sure you have the necessary tools installed:

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

    - **Add Docker's GPG Key**:

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -

    - **Add the Docker APT Repository**: Since you are using Debian Bookworm, add the repository as follows:

echo "deb [arch=amd64] https://download.docker.com/linux/debian bookworm stable" | sudo tee /etc/apt/sources.list.d/docker.list

3. **Update your package index**: Now that the Docker repository is added, update the package index:

sudo apt-get update

4. **Install Docker Engine and Related Packages**: Now you can try to install Docker Engine and its components:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

5. **Install Missing Dependencies**: If you still face issues with missing dependencies (like qemu-system-x86, pass, or uidmap), install them manually:

sudo apt-get install qemu-system-x86 pass uidmap

6. **Install Docker Desktop Again** (Optional): If you want Docker Desktop specifically, after installing Docker Engine, you can try installing Docker Desktop again:

sudo dpkg -i /root/Téléchargements/docker-desktop-amd64.deb

7. **Fix Any Remaining Dependency Issues**: If you encounter any dependency problems again, use the following command:

sudo apt-get install -f

Set up proxy :

1. **Create proxy service file (only for IUT Lyon 1 proxy)**

mkdir /etc/systemd/system/docker.service.d

2. **Write in this file**

nano /etc/systemd/system/docker.service.d/http-proxy.conf

[Service]

Environment="HTTP_PROXY=http://proxy.iutbourg.univ-lyon1.fr:3128/"

Environment="HTTPS_PROXY=http://proxy.iutbourg.univ-lyon1.fr:3128/"

3. **Reload systemctl**

systemctl daemon-reload

systemctl restart docker

Test docker :

1. **Docket install test**

Docker run hello-world

2. **Remove test image**

docker images

// get the id of hello world

docker rmi id d2c9 -f // to force the delete

PHP container :

1. **Create a folder to work into**
2. **Create phpinfo file**

nano info.php

3. **Write into it**

echo "<?php phpinfo(); ?>" > info.php

4. **Check docker hub to search info for php container**
5. **Create a php container**

docker run --rm -p 8000:80 -v "$PWD":/var/www/html php:8.2-apache

-rm to delete the container after the stop

-p to set the port 80 for the port 8000 of the container

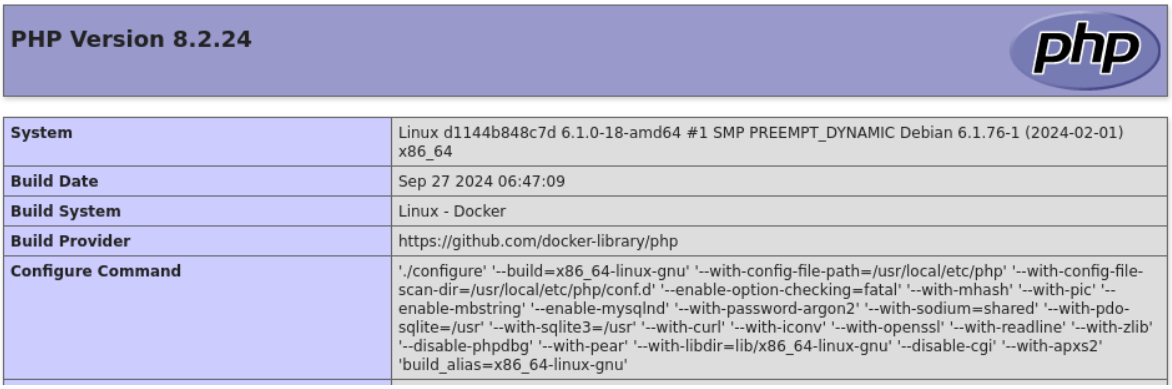
-v "$PWD":/var/www/html : set the work repository where apache2 will search the files and folders

We can use php:8.1 or 8.0-apache to set the version of php

6. **Load a .ini file**

docker run -p 8000:80 \ -v "$PWD":/var/www/html \ -v /etc/php/8.2/apache2/php.ini:/usr/local/etc/php/php.ini \ php:8.2-apache

// here, the php ini file of the computer is load in the container and use into it



PHP Version 8.2.24

System	Linux d1144b848c7d 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.76-1 (2024-02-01) x86_64
Build Date	Sep 27 2024 06:47:09
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' 'build_alias=x86_64-linux-gnu'

7. **Create a .ini file**

nano php.ini

8. **Basic configuration**

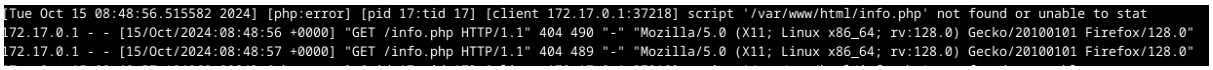
display_errors = On

error_reporting = E_ALL

memory_limit = 128M

upload_max_filesize = 10M

post_max_size = 10M



```
[Tue Oct 15 08:48:56.515582 2024] [php:error] [pid 17:tid 17] [client 172.17.0.1:37218] script '/var/www/html/info.php' not found or unable to stat
172.17.0.1 - - [15/Oct/2024:08:48:56 +0000] "GET /info.php HTTP/1.1" 404 490 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
172.17.0.1 - - [15/Oct/2024:08:48:57 +0000] "GET /info.php HTTP/1.1" 404 489 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
```

Now we have a message for every connection and we have that :

Start MySQL Server

1. **Create a container**

docker run --name my-mysql \ -e MYSQL_ROOT_PASSWORD=root \ -v mysql-data:/var/lib/mysql \ -p 3306:3306 \ -d mysql:latest

I have an error :

docker: Error response from daemon: driver failed programming external connectivity on endpoint my-mysql (7c70d36335eae611ef77ac11c9e0dc0d1cd7907eaa2fea09ecf09da929141d83): failed to bind port 0.0.0.0:3306/tcp: Error starting userland proxy: listen tcp4 0.0.0.0:3306: bind: address already in use.

Need to check with container use this port :

Lsof -I :3306

Close port:

Systemctl stop my-mysql

Authorize proxy to docker:

Need to add the proxy in the config.json in the /docker folder

mkdir ~/.docker

nano ~/.docker/config.json

{ "proxies": { "default": { "httpProxy": " http://proxy.iutbourg.univ-lyon1.fr:3128/

", "httpsProxy": " http://proxy.iutbourg.univ-lyon1.fr:3128/

", "noProxy": "localhost,127.0.0.1" } } }

Add iutbgdin and give permission

1. Create the docker group.

sudo groupadd docker

2. Add your user to the docker group.

sudo usermod -aG docker $iutbgdin

create mysql container now the configuration is fine

docker run --name my-mysql -e MYSQL_ROOT_PASSWORD=root -d mysql:latest

docker exec -it my-mysql mysql -u root -p

SHOW DATABASES;

```
➔ +--------------------+
➔ | Database           |
➔ +--------------------+
➔ | information_schema |
➔ | mysql              |
➔ | performance_schema |
➔ | sys                |
➔ +--------------------+
```

➔ 4 rows in set (0.00 sec)

Test

1. **do a volume alpine**

docker run -it --rm --volumes-from my-mysql alpine sh

2. **Create a bash script inside alpine**

apk add --no-cache mysql-client

2.1. **ERROR permission denied**

/ # *apk update*

/ # apk add --no-cache mysql-client

fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/main/x86_64/APKINDEX.tar.gz

WARNING: fetching https://dl-cdn.alpinelinux.org/alpine/v3.20/main: Permission denied

Try to ping alpine to see if internet network work

Use : ping dl-cdn.alpinelinux.org

PING dl-cdn.alpinelinux.org (146.75.118.132): 56 data bytes

--- dl-cdn.alpinelinux.org ping statistics ---

5 packets transmitted, 0 packets received, 100% packet loss

Run the container by using a public DNS and not the one of iut lyon 1 to avoid permission denied

docker run -it --rm --dns 8.8.8.8 --dns 8.8.4.4 alpine sh

Write in the container *vi /etc/apk/repositories*

Then replace the link by :

https://mirror.clarkson.edu/alpine/v3.20/main

https://mirror.clarkson.edu/alpine/v3.20/community

3. **Create a script**

(when a file is created, we can add a “chmod +x create_db.sh” for example to add the executable permissions if needed)

echo '#!/bin/sh' > create_db.sh

echo 'mysql -u root -p"root" -e "CREATE DATABASE my_database;"' >> create_db.sh

echo 'mysql -u root -p"root" -e "CREATE TABLE my_database.my_table (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255));"' >> create_db.sh

chmod +x create_db.sh

4. **Backup database**

echo '#!/bin/sh' > backup_db.sh

echo 'mysqldump -u root -p"root" --all-databases > /var/lib/mysql/backup.sql' >> backup_db.sh

chmod +x backup_db.sh

5. **Use a backup**

mysql -h my-mysql -u root -p"${MYSQL_ROOT_PASSWORD}" < /var/lib/mysql/backup.

chmod +x create_db.sh

6. **Execute script (put ./ or sh in front of the command)**

./create_db.sh

./backup_db.sh

./restore_db.sh

7. **Common problem and error**

The configuration of the container and docker is almost the same for every instance but we can encounter some problems like DNS, proxy and connection problem because the connection of the iut lyon 1 debian OS are related to a DNS and a proxy, to avoid that, some exemple in this documentation can be useful, like adding a public DNS (namespace 8.8.8.8) or change the proxy like in the user configuration but also in the docker/config.json file.

All these problems are related to the proxy of the IUT Lyon 1.