DOCKER WORDPRESS APPLICATION

Table of Contents

This is an application for `WordPress` and how it was made step by step. We are going to need the application itself, a database, in this case MariaDB and finally an Nginx proxy server, for load-balancing the containers etc. You can install all the files by yourself, following the guide that follows or you can download the application itself from the GitHub. https://github.com/VissarionM/docker-wordpress. There is always the option to pull the latest images in the `docker-composer` file.

This guide follows Windows and Linux installation, but the app will run in linux OS.

OS for docker installation

Windows

Download from `Microsoft Store`, WSL linux distributions. For Docker Desktop installation instructions, check Windows Docker Install. After installation, make sure to enable **WSL integration in Docker's Resources**. Also, enable each distribution you want to use in the resources `list`.

Linux

Either a VM or machine, download one of the following - Ubuntu 22.04.3 LTS - Ubuntu 22.04 LTS - Ubuntu 20.04.6 LTS - Ubuntu 18.04.6 LTS

For Docker Engine installation you can follow the instructions, check Ubuntu Docker Install.

Uninstall all conflicting packages

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker
containerd runc; do sudo apt-get remove $pkg; done
```

1. Set up Docker's `apt` repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

2.Install the Docker packages.

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

3.Verify that the Docker Engine installation is successful by running the `hello-world` image.

```
sudo docker run hello-world
```

Dockerfiles and dependencies

Windows

Create a folder for each image, then download all the files in the links below and place them inside.

MariaDB version:

- MariaDB dockerfile
- MariaDB docker-entrypoint.sh
- MariaDB healthcheck.sh

Wordpress version:

- Wordpress dockerfile
- Wordpress docker-entrypoint.sh
- Wordpress wp-config-docker.php

Nginx version:

- Nginx dockerfile

Ubuntu

Create the script `image-install.sh` or you can follow the next instructions or copy-paste in cli and run as commands.

```
#!/bin/bash

# Create a directory for the project
mkdir -p ~/downloads/Project2024

# MariaDB
mkdir -p ~/downloads/Project2024/MariaDB
cd ~/downloads/Project2024/MariaDB
wget https://raw.githubusercontent.com/MariaDB/mariadb-
docker/44ed2e231207787c08d56acc94b79d5f06efe006/11.2/Dockerfile[MariaDB dockerfile]
wget https://raw.githubusercontent.com/MariaDB/mariadb-
docker/44ed2e231207787c08d56acc94b79d5f06efe006/11.2/docker-entrypoint.sh[MariaDB
docker-entrypoint.sh]
wget https://raw.githubusercontent.com/MariaDB/mariadb-
docker/44ed2e231207787c08d56acc94b79d5f06efe006/11.2/healthcheck.sh[MariaDB
healthcheck.sh]
```

```
chmod +x docker-entrypoint.sh healthcheck.sh

# WordPress
mkdir -p ~/downloads/Project2024/WordPress
cd ~/downloads/Project2024/WordPress
wget https://raw.githubusercontent.com/docker-
library/wordpress/2eba26f088c12dea179623ee8af5cb52e6f60c4a/latest/php8.3/fpm/Dockerfil
e[Wordpress dockerfile]
wget https://raw.githubusercontent.com/docker-
library/wordpress/2eba26f088c12dea179623ee8af5cb52e6f60c4a/latest/php8.3/fpm/docker-
entrypoint.sh[Wordpress docker-entrypoint.sh]
wget https://raw.githubusercontent.com/docker-
library/wordpress/2eba26f088c12dea179623ee8af5cb52e6f60c4a/latest/php8.3/fpm/wp-
config-docker.php[Wordpress wp-config-docker.php]
chmod +x docker-entrypoint.sh

# Nginx
mkdir -p ~/downloads/Project2024/Nginx
cd ~/downloads/Project2024/Nginx
wget https://raw.githubusercontent.com/nginxinc/docker-
nginx/3180cdbec313dc4a9f6dd1109ae66adaf98f11fb/mainline/alpine/Dockerfile[Nginx
dockerfile]

cd ..
cd ..
cd ..

echo "Download complete."
```

Run the command:

```
./dockerfile-install.sh
```

Set up

Windows

For Windows you need to install `Git Bash` to run scripts and `Openssl` for self-generated certificate and key for https. You can get Git Bash at Git Bash Download. You can get at Openssl Download.

Ubuntu

Here, we have pre-installed OpenSSL. You can install the "ifconfig command" by, RUN [sudo apt install net-tools]. We log in using "$ docker login" and entering our username and password.

GUI/CLI

Installing CLI Tools

Install `lynx` or `links` for the command-line interface (CLI) to run Wordpress:

```
sudo apt-get install lynx  // For Lynx
// OR
sudo apt-get install links  // For Links
```

Installing GUI for Browser

Install a GUI to run Wordpress. For example, to install Firefox, run:

```
sudo apt-get install firefox
```

SSL keys (self-generated)

Step 1: Use Linux or Git Bash on Windows. RUN [openssl req -newkey rsa:2048 -nodes -keyout name_of_key.key -x509 -days 365 -out name_of_certificate.crt]

Breakdown of the command:

- openssl req: This command invokes the certificate request and certificate generating utility.
- newkey rsa:2048: This option generates a new RSA key of 2048 bits.
- nodes: This option tells OpenSSL to not encrypt the private key.
- keyout localhost.key: This specifies the output file for the private key.
- x509: This option outputs a self-signed certificate instead of a certificate request.
- days 365: This option sets the certificate validity period to 365 days.
- out localhost.crt: This specifies the output file for the certificate.

Step 2: Provide Certificate Details After running the command, you will be prompted to provide information for your certificate. Since this is for localhost testing, you can fill in the details as you wish. Current certificate is filled as follows:

```
    Country Name (2 letter code) [XX]:GR
    State or Province Name (full name) []:Attica
    Locality Name (eg, city) [Default City]:Athens
    Organization Name (eg, company) [Default Company Ltd]:Uniwa
    Organizational Unit Name (eg, section) []:CloudComputingLab
    Common Name (eg, your name or your server's hostname) []:localhost
    Email Address []:admin@localhost
```

Make a folder named `ssl` in the main folder and place them inside.

Build images

Create a script for the script to build the docker images.

```
#!/bin/bash

docker build -t mariadb ./mariadb
docker build -t wordpress ./wordpress
docker build -t nginx ./nginx
```

Run the command:

```
./build.sh
```

Compose images

First, open the `docker-composer.yaml`. We will create services, which means the creation of the containers that will run.

- All the containers belong to the same `network`.
- All containers have their `storage`.
- All containers have their restrictions for `Out of Memory Exceptions(OOME)`.
- Expose `ports` in which they listen.

We need the database to be created first, then wordpress and then nginx, so that's why we use `depends_on`, which instrcuts the container to wait for their turn during creation.

Also, we create an example for user and password in mariadb and wordpress. The variables in the file `wp-config-docker.sh` of wordpress directory must be the same with wordpress.

Run the command:

```
docker-compose up -d
```

Run application

Create the script `App.sh` to execute the wordpress app. This script looks first the OS of the machine, then if it has `GUI`. If it has then open a `browser with http://localhost` to open Wordpress. If it is Windows it will open the browser, if it is ubuntu check for `lynx`. If it not not installed, the install it. Finally run it in `CLI` with:

```
lynx http://localhost
```

For GUI, you can always install the browser you wish.

App.sh

```
#!/bin/bash

# Function to open URL based on OS
open_url() {
    local url=$1

    # Detect the operating system
    if [[ "$OSTYPE" == "darwin"* ]]; then
        open $url
    elif [[ "$OSTYPE" == "linux-gnu"* ]]; then
        if [ -n "$DISPLAY" ]; then
            xdg-open $url
        else
            if command -v lynx &>/dev/null; then
                lynx $url
            else
                echo "CLI browser Lynx is not installed. Installing..."
                sudo apt update
                sudo apt install lynx
                lynx $url
            fi
        fi
    elif [[ "$OSTYPE" == "msys" || "$OSTYPE" == "win32" ]]; then
        # Use PowerShell to open the URL on Windows
        powershell.exe Start-Process $url
    else
        echo "Unsupported operating system"
        exit 1
    fi
}

# URL to open
url="http://localhost"

# Call function to open URL
open_url $url
```

```
sudo ./App.sh
```

Structure of project

- Project2024
  - mariadb
    - docker-entrypoint.sh
    - Dockerfile
    - healthcheck.sh **nginx
    - Dockerfile
  - nginx-conf
    - nginx.conf
    - nginxHTTPS.conf
  - ssl
    - certificate.crt
    - private.key
  - wordpress
    - docker-entrypoint.sh
    - Dockerfile
    - wp-config-docker.php
  - build.sh
  - docker-compose.yaml
  - docker-composeHTTPS.yaml
  - README.md
  - RunApp.sh

Services

For Wordpress, we use `php-fpm` that listens to port:9000 because it runs the best with Nginx. We use an Nginx-alpine because it is lightweight, stable and runs smoothly with fpm. For nginx.conf, copy or download the file in the GitHub repo link that was given earlier. Finally, we use a database, in this case `MariaDB`.

Make sure that all needed ports have access through the firewall. Command is *$ sudo ufw allow <port>*.

- port:22/tcp (ssh/Openssh)
- port:80/tcp, 8080/tcp, 443/tcp (Nginx)
- port:9000/tcp (php-fpm)
- port:3306/tcp (MariaDB)

- port:2377/tcp, 4789/udp, 7946/tcp and udp (Docker swarm)

Run containers

Execute the command:

```
docker-compose up -d
```

Show running containers

Execute the command:

```
docker ps
```

Stop containers

Execute the command:

```
docker down
```

Additional information

If want to use `https` then we change the `docker-composer.yaml` with the `docker-composerHTTPS.yaml` and the `nginx.conf` with the `nginxHTTPS.conf` file. When we run the app, it will not be trusted and we will be given a warning for not being secure. This is why we use self-generated but we can proceed since it's our machine and it is **ONLY for testing purposes**.

For windows users

Eventually you will need to use linux for Docker Swarm. Windows can't see the `docker0` network of Docker Engine. If you use a `WSL distribution`, even though it can recognize the docker0 network, due to `vEthernet (WSL (Hyper-V firewall))`, you can't communicate outside the machine. Windows can communicate with VM and vice versa, but VMs can't go further. Upload all your project to a repository in Git Hub or use the Author's Repository.

If you want to transition to Ubuntu, then start Ubuntu, log in Git Hub, clone your repository and pull all the files. - Execute "$ sudo apt install git-hub". - git clone https://github.com/username/repository.git - Change directory to the one that was created by git. - Give permission to execute to all scripts by going through all directories and using "$ chmod +x ./script_name.sh" for each script.

In case you encounter any problem in scripts then run: `$ sudo apt install dos2unix`, then `dos2unixApp.sh`.

Docker swarm

Create a docker in a server to use as `manager`. Use the command `docker swarm init --advertise-addr <IP address>` Our temporary server is a VM - Ubuntu Jammy Cloud LTS, Author's temporary Server. We find our IPv4 address using:

```
ifconfig
```

IP Address:`83.212.72.64`

Execute the command:

```
sudo docker swarm init --advertise-addr 83.212.72.64
```

To add a `worker` to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-18zeatvpqvrc4hjde4cj5vvi9s7crtsh6a8zplvzinh6ozrhbl-
dl014dsg124zuuxb5v01yea6z 83.212.72.64:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

We find the name of the host with:

```
sudo docker node ls
```

Guide and Install

You can use an online editor to make the adoc to pdf files.

- asciiDoc editor

In order to make the files in Ubuntu give the commands that follow.

For guide pdf:

```
docker run -it --rm -v ${PWD}:/documents/ asciidoctor/docker-asciidoctor asciidoctor-
pdf --safe -a toc -o /documents/guide-asciidoc.adoc.pdf /documents/guide-asciidoc.adoc
```

For install pdf:

```
docker run -it --rm -v ${PWD}:/documents/ asciidoctor/docker-asciidoctor asciidoctor-
pdf --safe -a toc -o /documents/install-asciidoc.adoc.pdf /documents/install-
```

```
asciidoc.adoc
```

If you don’t have the image, it will be pulled by giving the command.

Explanation of each part of the command:

- docker run -it --rm: Runs a Docker container in interactive mode and removes the container after it exits.
- -v $(pwd):/documents/: Mounts the current working directory (where your .adoc file is located) to /documents/ inside the container.
- asciidoctor/docker-asciidoctor: Uses the Asciidoctor Docker image.
- asciidoctor-pdf --safe -a toc -o /documents/guide.adoc.pdf /documents/guide.adoc: Runs the asciidoctor-pdf command inside the container to convert the Asciidoc file to a PDF with the specified options.

That’s it, you are ready to go.