AERO-WEBB
Technical Direction Docs

# How to install wsl2 docker

# How to install wsl2 docker

## Resource

- Microsoft Official Website - WSL install
- Microsoft Official Website - Basic CLI
- Install Docker on WSL

## WSL install

Just follow the Microsoft tutorial (link above):

- Verify that your PC runs with Windows 10/11 and up to date (Windows 10 versions 2004 and more (build 19041 and more) or Windows 11.)
- Prefer recommended Linux distribution : Ubuntu (without version number behind)
- When install is finished, open a Powershell window and do:

```
wsl --status
```

If WSL version is 1, do:

```
wsl --set-default-version 2
```

If WSL version 2 and linux kernel version are given you are ready to install Docker.

## Docker install

**Prerequisites**

- WSL2
- Linux distro (the user defined at installation is a root one).

You can follow the tutorial (link above) or if you have installed Ubuntu execute commands listed below, (simplified version from the Bowman tutorial).

In the context of the AeroWebb project, Docker allows the execution of unit tests that perform operations on the database, by simulating this database.

### Enable Systemd

By default, Systemd is disabled in your distro in WSL, so in order to enable it you'll have to do the following :

- open your `/etc/wsl.conf` file with your favourite text editor (here it's vi) `sudo vi /etc/wsl.conf`

- add the following block

```
[boot]
systemd=true
```

- save and quit (for vi press Escape and type `:x` then press Enter, for nano it's `ctrl+x`)
- shutdown your wsl session in powershell `wsl --shutdown`
- relaunch a new WSL session
- for checking that Systemd is now up you just have to enter the following command `systemctllist-unit-files --type=service`
- You should see a list of all services installed in Linux. Next, press `q` to quit

## Update/upgrade packages

```
sudo apt update && sudo apt upgrade
```

As we use no proxy, no network connectivity problem should arise during update.

## Prepare for Docker installation: remove residue

```
sudo apt remove docker docker-engine docker.io containerd runc
```

## Prepare for Docker installation: install dependencies

```
sudo apt install --no-install-recommends apt-transport-https ca-certificates curl gnupg2
```

## Debian/Ubuntu package repository configuration

First temporarily set some OS-specific variables:

```
source /etc/os-release
```

Then, make sure that apt will trust the repo:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/trusted.gpg.d/docker.asc
```

Then add and update the repo information so that apt will use it in the future:

```
echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu ${VERSION_CODENAME} stable" |
sudo tee /etc/apt/sources.list.d/docker.list
```

```
sudo apt update
```

## Install Docker

Now we can install the official Docker Engine and client tools:

```
sudo apt install docker-ce docker-ce-cli containerd.io
```

### Add user to docker group

Now we can install the official Docker Engine and client tools:

```
sudo usermod -aG docker $USER
```

Then close that WSL window, and launch WSL again. You should see docker when you run the command groups to list group memberships.

```
groups
```

### Launch Docker

To launch Docker, you'll only need Systemd, use the following command in order to do it :

```
systemctl start docker
```

To test it, you will have to test the following commands :

```
systemctl status docker
docker run --rm hello-world
```

If the pull is successful you're done with basic configuration! Now you can work directly on WSL with docker. If some of your docker instances need to work with 2 MoRO network, some further configuration is needed, check below.

## WSL2, Docker and 2MoRO

To be able to pull artefacts from Nexus, Docker must have access.

Create required files (as root)

```
sudo su
cd /etc/docker
touch daemon.json
touch config.json
```

Edit (press `i`) and fill daemon.json (press Escape then type `:x` to save file / press Escape then type `:q!` to exit without saving)

```
vi daemon.json
```

with following JSON

```
{
    "insecure-registries": [
        "nexus-ext.2moro.fr:8088"
    ]
}
```

Edit (press `i`) and fill config.json (press Escape then type `:x` to save file / press Escape then type `:q!` to exit without saving)

```
vi config.json
```

with following JSON

```
{
    "auths": {
        "nexus-ext.2moro.fr:8088": {
            "auth": "ZG9ja2VyOkFVTWQ3em1weXBRN0tTZkxpMHdVRA=="
        }
    }
}
```

Restart the `docker` service :

```
$ systemctl restart docker
```

Now you can pull from 2MoRO Nexus! And you're all done!

## Add Portainer in my docker

If you want to still have a GUI for your Docker like you had with Docker Desktop, then Portainer will be your future best friend.

First enter this in your WSL

```
docker run -d -p 8111:8000 -p 9113:9443 --name portainer \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data \
portainer/portainer-ce:latest
```

**Note**

Here, the default Portainer ports 8000 and 9443 are remapped to 8111 and 9113 respectively to avoid any conflict with other (default) ports from other tools.

Then, once your Docker with Portainer is installed and launched, go to this URL https://localhost:9113/, then you just have to create your login and password for Portainer, and it's all good.

## Troubleshooting

## If you get an IPTables error when launching `dockerd`, do the following

```
update-alternatives --config iptables
```

Then select iptables-legacy instead of iptables-nft, it should work fine after this.

## To allow docker pull from default WSL profile

Restart WSL then run the following command:

```
ln -s /etc/docker ~/.docker
```

## To start Docker automatically at WSL startup

Once you have installed Docker and enabled systemd, you'll just have to do the following command

```
systemctl enable docker
```

It will start automatically Docker when you start your session, and also you will be able to check that Docker is up with this command

```
systemctl status docker
```