WSL 2 MICHELIN UBUNTU 24.04

(This documentation follows some instructions from this link, if you have some trouble while installing WSL maybe some issues are resolved here)

Pre-requisites

- You must have a Windows 10-11 end user operating system up to date
- You must have administrator permission on your operating system. If this is not case, you must follow this procedure on service now: link for admin right procedure

WSL2 install

Open a Windows terminal as administrator and run:

```
wsl.exe --set-default-version 2

wsl.exe --install -d Ubuntu-24.04

wsl.exe –-update
```

/!\ During setup, Windows will force you to reboot your computer.

At the end of the installation, open an Ubuntu24.04 Terminal to set your Ubuntu credentials.

WSL configuration File

You can personalize the way your sub-systems work in the .wslconfig file. The file is usually in your user folder (e.g.: C:\Users\E123456). If the file doesn't exist, create it (just name it .wslconfig). Make these changes in the file, they are essential to the DNS and GPU configurations:

```
[wsl2]

gpuSupport=true

guiApplications=true
```

```
networkingMode=mirrored
dnsTunneling=true
dhcp=false

[experimental]
sparseVhd=true
hostAddressLoopback=true
bestEffortDnsParsing=true
```

You will need to restart your WSL afterward.

Ensure good communication of your subsystem

For some obscure reasons, when you use WSL behind a VPN, you are not able to have a correct DNS resolution. A not closed issue is published on Microsoft github here.

Fortunately, there is a solution based on vpnkit. This tool is able to modify WSL routing DNS rules to forward them to the host system. It is not a beautiful solution but the best yet.

A project, based on WSL distribution, can help us to provide a ready-to-use service on this github repository.

The steps to install wsl-vpn kit are:

- Create a working directory on your windows workspace and download inside the v0.3.8 release wsl-vpnkit.tar.gz asset from this page.
- Now, open a powershell and go to the location of wsl-vpnkit.tar.gz, downloaded during the previous step
- On your powershell terminal, launch:

(NOTE

In the wsl --import wsl-vpnkit $env:USERPROFILE\wsl-vpnkit wsl-vpnkit.tar.gz, $env:USERPROFILE\wsl-vpnkit is the destination where vpn-kit will be installed. If you want to install vpn-kit in another folder, replace **$env:USERPROFILE\wsl-vpnkit** by the desired destination.)

```
wsl --import wsl-vpnkit $env:USERPROFILE\wsl-vpnkit wsl-vpnkit.tar.gz
wsl -d wsl-vpnkit
wsl -d wsl-vpnkit service wsl-vpnkit start
```

To ensure all wsl reboot good communication, we will write in your .profile file of your ubuntu user the wsl-vpnkit initialization command:

```
echo 'wsl.exe -d wsl-vpnkit service wsl-vpnkit start' >> ~/.profile
```

Relaunch your WSL Terminal.

Configure DNS

By default, WSL uses the windows host as DNS upstream. For many usages, this configuration works but a lot of people have sent support requests because they had DNS problems. That is why, it is safer to force WSL system to directly communicate with Michelin's DNS servers. To do it, in first time we will indicate to WSL to not generate ***resolv.conf*** file by adding the generateResolvConf to false in ***/etc/wsl.conf*** file. Run in your ubuntu:

NOTE : You have to remove the additional line breaks between these lines when pasting them in your terminal

```
cat << EOF | sudo tee /etc/wsl.conf > /dev/null
[network]
generateResolvConf = false
[boot]
Systemd = true
EOF
```

Next, we will delete rewrite the ***resolv.conf*** and with Michelin nameserver

```
sudo rm -f /etc/resolv.conf
```

Then type :

NOTE : You have to remove the additional line breaks between these lines when pasting them in your terminal

```
cat << EOF | sudo tee /etc/resolv.conf > /dev/null

nameserver 10.255.255.254

search adgroup.michelin.com lad.michelin.com

EOF
```

Next in ***resolved.conf*** (***/etc/systemd/resolved.conf***), make the following changes:

```
[Resolve]

DNS=10.225.1.1 10.216.84.130

FallbackDNS=8.8.8.8
```

Configure git with a ssh key

In order to import the Michelin CA certificates, you need to configure git with an SSH key.

Here is how to create and add the key to your account:

Go into your home directory:

```
cd ~
```

Generate the key :

```
ssh-keygen -t rsa -b 4096 -C "firstname.familyname@michelin.com"
```

You can choose the file in which the key is saved. You can choose a passphrase (empty is fine).

Go to your Gitlab Michelin account, go to User Settings / SSH Keys. Click Add new key.

In your terminal, paste:

```
cat ~/.ssh/id_rsa.pub
```

Copy the reply (it should start with ssh-rsa and end with firstname.familyname@michelin.com) in the key field of your github page. Add a Title.

Then Click Add key. You can now follow the next step.

Add Michelin CA certificates

Michelin SI is behind a ssl proxy with his proper PKI for certificates delivering. That is why, your subsystem must add this PKI to her recognized authorities.

To do this, we will clone a repository with the certificates in the subsystem and copy them to ca-certificates:

```
rm -rf /tmp/certificates

git clone git@gitlab.michelin.com:DEV/bib-certificates.git /tmp/certificates

sudo cp /tmp/certificates/* /usr/local/share/ca-certificates/

sudo update-ca-certificates
```

If everything is ok, terminal notify you that certificates have been added.

You can now clean the temp working folder:

```
rm -rf /tmp/devops_environment
```

Configure APT

The last step, to have a subsystem ready to use, is to have an apt with Michelin trusted sources configured. Ubuntu based package repositories can't be used behind Michelin proxy.

In ***ubuntu.sources*** (***/etc/apt/sources.list.d/ubuntu.sources***) make the following changes:

```
Types: deb
URIs: https://artifactory.michelin.com/artifactory/ubuntu-archive-remote
Suites: noble noble-updates noble-backports
Components: main universe restricted multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg

## Ubuntu security updates. Aside from URIs and Suites,
## this should mirror your choices in the previous section.
Types: deb
URIs: https://artifactory.michelin.com/artifactory/ubuntu-security-remote
Suites: noble-security
Components: main universe restricted multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
```

Try:

```
sudo apt-get update
```

GPU Configuration

This configuration will only work if you have an NVIDIA GPU.

You will need the NVIDIA driver on Windows: https://www.nvidia.com/en-us/drivers/

Look for your GPU and download the driver.

Afterward, in your WSL, at the end of the file ***~/.profile***, add:

```
. "$HOME/.local/bin/env"
export MESA_D3D12_DEFAULT_ADAPTER_NAME=NVIDIA
```

Then you need to install the nvidia toolkit, in your Ubuntu terminal, enter:

```
cd /tmp &&
 wget https://github.com/NVIDIA/nvidia-container-toolkit/releases/download/v1.17.5/nvidia-container-toolkit_1.17.5_deb_amd64.tar.gz && \
  tar -xvf nvidia-container-toolkit_1.17.5_deb_amd64.tar.gz && \
 cd /tmp/release-v1.17.5-stable/packages/ubuntu18.04/amd64 && \
  sudo dpkg -i *.deb && \
  sudo nvidia-ctk runtime configure --runtime=docker && \
```

Relaunch your WSL Terminal.

You can check if your NVIDIA is used as principal GPU with: (install the dependencies if needed)

```
glxinfo -B | grep Device
```