昨日，`systemd` と `systemctl` コマンドを `WSL` 上で実行できるようにしたが，`nvidia-docker` を実行することはできなかった．そこで，今日は再度 `NVIDIA CUDA toolkit` のインストールから，正常に動作しているかの確認を含めて環境を構築していく．

1. NVIDIA ドライバのインストール

NVIDIA Docker って今どうなってるの？ (20.09 版) を参考に以下のコマンドを実行し，`CUDA` をインストールする.

```
$ wget https://developer.download.nvidia.com/compute/cuda/repos/wsl-ubuntu/x86_64/cuda-wsl-ubuntu.pin
$ sudo mv cuda-wsl-ubuntu.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ wget https://developer.download.nvidia.com/compute/cuda/11.2.2/local_installers/cuda-repo-wsl-ubuntu-11-2-local_11.2.2-1_amd64.deb
$ sudo dpkg -i cuda-repo-wsl-ubuntu-11-2-local_11.2.2-1_amd64.deb
$ sudo apt-key add /var/cuda-repo-wsl-ubuntu-11-2-local/7fa2af80.pub
$ sudo apt update
$ sudo apt-get -y install cuda
```

そして，『もう依存関係に悩まない！ nvidia-docker でクリーンな GPU 環境を作る』を参考に，以下のコマンドによって `CUDA` が正常に動作するかを確認する.

```
$ nvcc -V
```

しかし，実際に実行してみると以下のような結果となった.

```
nvcc -V

Command 'nvcc' not found, but can be installed with:

sudo apt install nvidia-cuda-toolkit
```

そこで，表示に従い `nvidia-cuda-toolkit` を `sudo apt install nvidia-cuda-toolkit` コマンドを用いてインストールした．その結果，

```
nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:07:16_PDT_2019
Cuda compilation tools, release 10.1, V10.1.243
```

という出力が得られた．ここで，`nvcc` については『NVIDIA® Deep Learning（深層学習）開発環境構築情報 NVIDIA® CUDA Toolkit』に以下のような説明があった.

> CUDA は、NVIDIA® が提供する並列コンピューティング アーキテクチャ、統合開発環境です。
> nvcc（NVIDIA CUDA Compiler）というコンパイラや、ライブラリなどから構成されています。

すなわち，`nvcc` とは CUDA プログラムのコンパイラらしく，`-V` オプションはそのバージョンを表示するためのものらしい．出力にも `nvcc: NVIDIA (R) Cuda compiler driver` としっかり表示されていた．

ここまでの疑問点
CUDA Driver と CUDA-Toolkit は同一のもの？
まず，次のサイト『NVIDIA Docker って今どうなってるの？ (20.09 版)』では，

> `cuda-drivers` パッケージをインストールというのをもう少し具体的に示しますと、例えば Ubuntu 18.04 に CUDA 11.2 をインストールするのであれば、`CUDA Toolkit` の Web サイトで下記のようなコマンドが提示されると思います。

と説明されており，また

> この最後の行を `sudo apt-get -y install cuda-drivers` とすれば、最新のドライバだけが綺麗にインストールされます。これがオススメの方法です。

と説明されていることから、これら 2 つのものは同一の可能性が高いと考えられる．また先ほどのサイトによると，以下のように `NVIDIA Driver` と `cuda-drivers` を同じように説明している．

> NVIDIA ドライバのインストール
>
> オススメの方法 (だけど案外知られていない気がする方法) はエヌビディア提供の cuda-drivers パッケージを使うことです。NVIDIA Docker の FAQ にもあるのです。

そこでこれら 3 つは同一の物であると考え，これまで `sudo apt install -y cuda` としていた部分を `sudo apt install -y nvidia-cuda-toolkit` に替えて以後実行していく．

2. systemd を PID 1 に

arkane-systems/genie の手順に従って `systemd-genie` をインストールしていく.

2.1. DNS サーバを 8.8.8.8 にする.

1. 以下のコマンドを実し, `vi` を立ち上げる.

```
$ sudo vi /etc/wsl.conf
```

2. `vi` に以下の手順で文字を入力する.
   - `i` を押す.
   - カーソル部分に以下のコマンドを貼り付け

```
[network]
generateResolvConf = false
```

   - `Esc` ボタンを押して入力モードを終了
   - `:wq` コマンドを入力して、ファイルに変更を保存して `vi` を閉じる.
3. 以下のコマンドで `/etc/resolv.conf` ファイルを削除して DNS の設定を登録する.

```
$ sudo rm /etc/resolv.conf
$ sudo sh -c "echo 'nameserver 8.8.8.8' > /etc/resolv.conf"
```

この処理は、再起動をすることなく有効となる.

2.2. dotnet-runtime-5.0 のインストール

1. 以下の 2 つのコマンドを実行し，パッケージリポジトリを追加する．

```
$ wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
$ sudo dpkg -i packages-microsoft-prod.deb
```

2. 以下のコマンドを実行し，`dotnet-SDK` をインストールする．

```
$ sudo apt update
$ sudo apt install -y apt-transport-https
$ sudo apt update
$ sudo apt install -y dotnet-sdk-5.0
```

`Welcome to .NET!` と表示されれば成功．

2.3. 残りのパッケージをインストールする．

以下のコマンドで残りのパッケージをまとめてインストールする．

```
sudo apt -y install \
daemonize \
dbus \
dotnet-runtime-5.0 \
gawk \
libc6 \
libstdc++6 \
policykit-1 \
systemd
```

2.4. wsl-translinux リポジトリを追加

https://arkane-systems.github.io/wsl-transdebian/の手順に従って，`wsl-translinux` リポジトリを追加する．

1. まず次のコマンドを実行していく．

```
$ sudo -s apt install apt-transport-https
$ sudo -s wget -O /etc/apt/trusted.gpg.d/wsl-transdebian.gpg \
https://arkane-systems.github.io/wsl-transdebian/apt/wsl-transdebian.gpg
$ sudo -s chmod a+r /etc/apt/trusted.gpg.d/wsl-transdebian.gpg
```

2. 次に，以下のコマンドを実行し，使用している`Ubuntu` のディストリビューションコードネームを確認する．

```
$ lsb_release -c
```

3. 1.で作成したファイルを `vi` で開く.

```
$ sudo vi /etc/apt/sources.list.d/wsl-transdebian.list
```

4. そして，ここに以下の情報を書き込む.

```
$ deb https://arkane-systems.github.io/wsl-transdebian/apt/ <distro> main
$ deb-src https://arkane-systems.github.io/wsl-transdebian/apt/ <distro> main
```

ここで， は先ほど `lsb_release -c` コマンドを実行した際に表示された文字列であり，例えば，`Ubuntu 20.04LTS` を使用している場合には以下のようになる.

```
$ deb https://arkane-systems.github.io/wsl-transdebian/apt/ focal main
$ deb-src https://arkane-systems.github.io/wsl-transdebian/apt/ focal main
```

これで `wsl-translinux` の追加は終了した.

2.5. systemd-genie のインストール

最後に，以下のコマンドを実行して `systemd-genie` をインストールする.

```
$ sudo -s apt update
$ sudo apt install -y systemd-genie
```

2.6. systemd-genie の実行

以下のコマンドで `genie` を実行する.

```
$ genie -s
```

以下の結果が表示された場合

```
Timed out waiting for systemd to enter running state.
This may indicate a systemd configuration error.
Attempting to continue.
```

一度 `systemctl status` を実行する.

```
 State: degraded
  Jobs: 0 queued
Failed: 2 units
```

その結果，上記のように `Failed:` が 0 以上カウントされた場合，以下のコマンドを実行して強引にエラーを取り除く.

```
$ sudo systemctl reset-failed
```

再度 `systemctl status` を実行して `Failed: 0 units` と表示されれば成功．もう一度 `genie -s` コマンドを実行した後 `ps aux` で一番上に `systemd` の文字かあることを確認する．

```
$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.0 174900 12900 ?        Ss   11:21   0:00 systemd
```

これで，`systemd-genie` のセットアップは完了

3. Docker のインストールと動作確認

3.1. Docker をインストール

1. まず，以下のコマンドを実行し，Docker をインストールする.

```
$ curl https://get.docker.com | sh
```

2. 以下のコマンドを実行し， Docker が正常にインストールされていることを確認.

```
$ docker --version
```

3.2. 一般ユーザで Docker を実行できるようにする

Docker を実行するたびに sudo をしなくて済むように一般ユーザを docker グループに所属させる.

1. 以下のコマンドを実行し，現在 docker グループに所属しているユーザ名の中に自分が含まれていないことを確認(初期状態の場合，以下の出力のように docker グループには誰も所属していない).

```
$ getent group | grep docker
docker:x:998:
```

2. 以下のコマンドを実行し， docker グループに現在ログインしているユーザを登録する．

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

3. 再度 1. のコマンドを実行し，ユーザが docker グループに所属したことを確認する．

```
$ getent group | grep docker
docker:x:998:miki
```

> ちょっと脱線
> ここまでのセットアップを終えた後，以下のコマンドを実行すると
>
> ```
> $ docker run hello-world
> ```
>
> 以下のようなエラーが発生したので，『docker コマンド実行時の「Got permission denied while trying to connect to the Docker daemon socket」』 を参考に，以下のコマンドで `docker.sock` にグループでの書き込み権限を付与する．
>
> ```
> $ sudo chgrp docker /var/run/docker.sock
> $ sudo service docker restart
> ```
>
> 最後にログアウトし，再度ログインして `genie -s` コマンドから実行していくと，先ほどのコマンドを走らせることができた．

4. NVIDIA ContainerToolkit のセットアップ

1. まず，リポジトリと GPG キーを設定する．

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
   && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \
   && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
```

続いて，`nvidia-docker2` をインストールする．

```
$ sudo apt update
$ sudo apt install -y nvidia-docker2
```

そして，以下のコマンドで `nvidia-docker2` コンテナが動作するかを確認する．

```
$ docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

しかし，やはり下記のエラーが吐かれた．

```
docker: Error response from daemon: OCI runtime create failed: container_linux.go:367: starting
container process caused: process_linux.go:495: container init caused: Running hook #0:: error running
hook: exit status 1, stdout: , stderr: nvidia-container-cli: initialization error: nvml error: driver
not loaded: unknown.
```

いくつかの試行錯誤

1. `install cuda-toolkit` & `install cuda`
   `cuda` のダウンロードが 92% で停止した．そのため，Ubuntu を再インストールし逆手順でインストールを行ったがやはり同様にインストールが停止した．
2. Ubuntu をインストールせず，直接 windows 上 にインストールした Docker Desktop から，nvidia-container を起動してみたが，以下のエラーが発生し，停止した．

```
docker: Error response from daemon: OCI runtime create failed: container_linux.go:367: starting
container process caused: process_linux.go:495: container init caused: Running hook #0:: error running
hook: exit status 1, stdout: , stderr: nvidia-container-cli: initialization error: nvml error: driver
not loaded: unknown.
```

このエラー番号(go:367, go:495)と一致するものとして，Redditの OCI Runtime error (I think) に質問に対する回答に解決策になりそうな投稿があった．

> コンテナを起動し ネストを 有効にし ましたか？

どうも、コンテナのネストを有効にするという方法があるらしい．この方法について microsoft 掲示板 には，

> 仮想マシンで WSL 2 を実行できますか?
> はい。 仮想マシンで入れ子になった仮想化が有効なことを確認する必要があります。 これを親 Hyper-V ホストで有効にするには、管理者特権を使用して PowerShell ウィンドウで次のコマンドを実行します。
> `Set-VMProcessor -VMName <VMName> -ExposeVirtualizationExtensions $true`
> " `<VMName>` " は、実際の仮想マシンの名前に置き換えてください。

上記のコマンドの `<VMName>` を以下のように変更し `powershell` にて実行した()ところ，

```
$ Set-VMProcessor -VMName Ubuntu-20.04 -ExposeVirtualizationExtensions $true
```

以下の結果が帰ってきた．

```
Set-VMProcessor: The term 'Set-VMProcessor' is not recognized as a name of a cmdlet,
function, script file, or executable program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
```

そのため，どこかから `Set-VMProcessor` コマンドをダウンロードできるのかと思い調べてみると，Windows PowerShell を使用して Azure Stack HCI 上の VM を管理する に以下のような説明があった．

> VM の仮想プロセッサを設定する
> VM の仮想プロセッサを構成するには、Set-VMProcessor コマンドレットを使用します。 その使用方法の詳細については、「Set-VMProcessor」のリファレンス ドキュメントを参照してください。

ここでまず気付くことは，このコマンドが Azure 上で動作する VM に対して使用するコマンドであるということである．そのため，現在 Azure を使用していない自分の PC では，使用することができない．よって，現場 `WSL` をネストする方法がないということになる．しかし，今まで見てきたサイトには，このことに関する情報がなかった．そこで考えられることが2つある．1つは，**Docker を wsl と ホスト側に 1つずつインストールしているため問題が発生している可能性**．もう1つは，**ネストするコマンドが他にもある可能性**である．