WSL + docker + GPU (4)

そろそろ環境構築を終わらせたいので、一度いままでの内容をまとめようと思う。

参考

- nvidia-docker コンテナ Deep Learning
- nvidia docker って今どうなってるの？(19.11 版)
- nvidia docker って今どうなってるの？(20.09 版)

1. 環境構築

1.1. Ubuntu の初期設定

まずパッケージのアップデートを行う。

```
sudo apt update

sudo apt upgrade
```

1.2. WSL kernel update

`WSL2` のカーネル `4.19.121-microsoft` 以上で動作しているかを確認する。

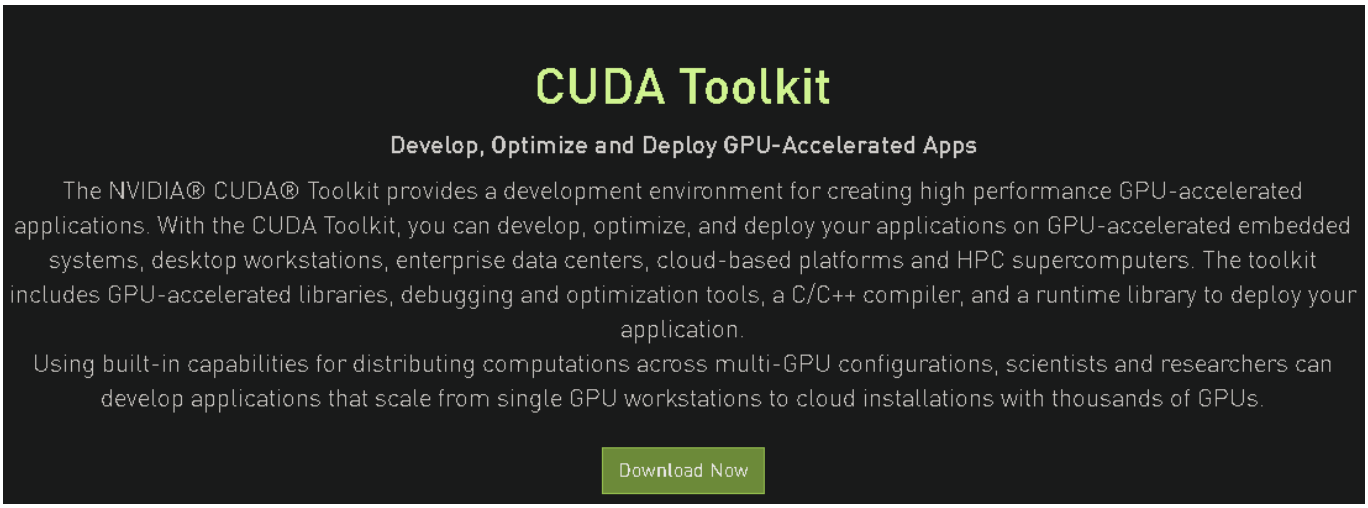
```
uname -r
```

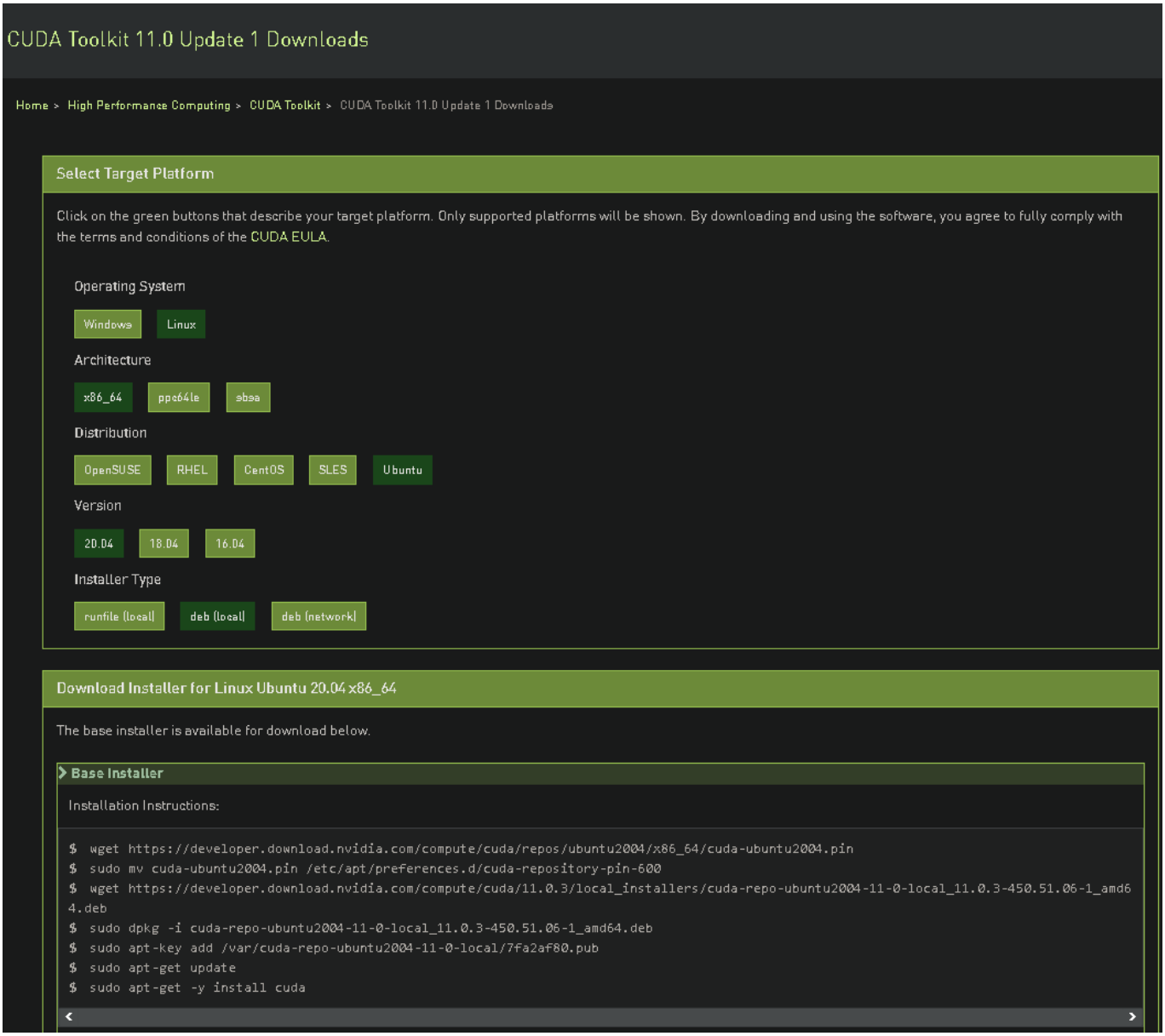
```
kohei@DESKTOP-Q123T6P:~$ uname -r
4.19.128-microsoft-standard
```

もし、`4.19.121-microsoft` より低い version で WSL が動作している場合は、このページを参考に、kernel を version up する。

1.3. NVIDIA ドライバのインストール

CUDA Toolkit の web サイト にアクセスし、自分が欲しいディストリビューションに合わせたダウンロードファイルを選択する。





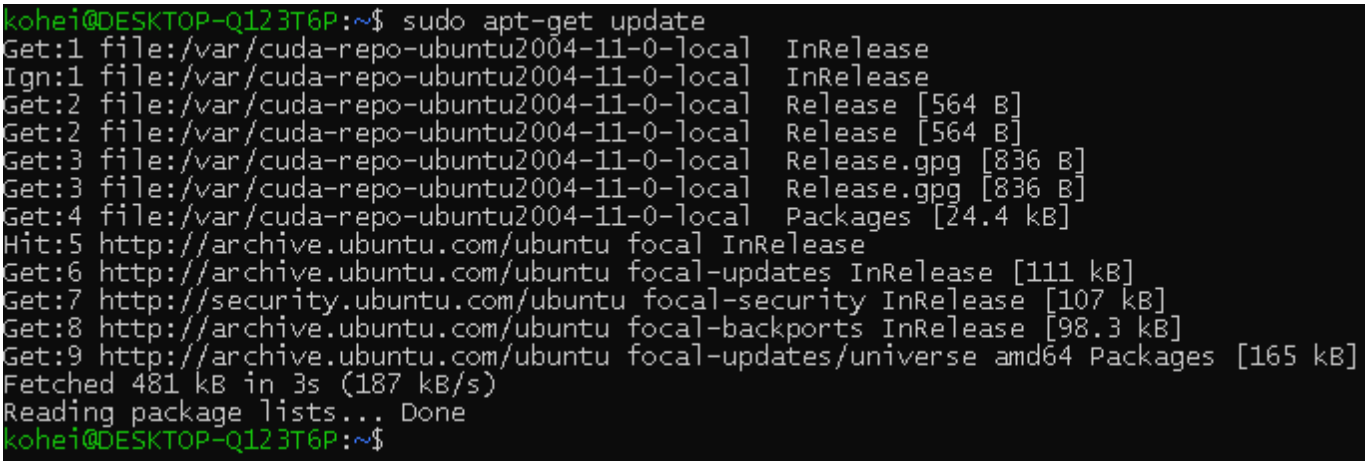
ここで、最後の行を

```
sudo apt-get -y install cuda-drivers
```

とすると、最新のドライバだけがきれいにインストールされる。

以下、CUDA Toolkit に記された通りにインストールする。

```
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64/cuda-
ubuntu2004.pin

sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600

wget
https://developer.download.nvidia.com/compute/cuda/11.0.3/local_installers/cuda-
repo-ubuntu2004-11-0-local_11.0.3-450.51.06-1_amd64.deb

sudo dpkg -i cuda-repo-ubuntu2004-11-0-local_11.0.3-450.51.06-1_amd64.deb

sudo apt-key add /var/cuda-repo-ubuntu2004-11-0-local/7fa2af80.pub

sudo apt update

sudo apt -y install cuda-drivers
```



1.4. Docker のインストール

Docker 公式のサイトの手順に従って、Docker をインストールする。

1. まず、古い version の Docker をアンインストールする。

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

2. 以下のコマンドで必要なリポジトリを追加する。

```
sudo apt update

sudo apt install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

```
kohei@DESKTOP-Q123T6P:~$ sudo apt install \
> apt-transport-https \
> ca-certificates \
> curl \
> gnupg-agent \
> software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20190110ubuntu1.1).
ca-certificates set to manually installed.
curl is already the newest version (7.68.0-1ubuntu2.2).
curl set to manually installed.
software-properties-common is already the newest version (0.98.9.2).
software-properties-common set to manually installed.
The following NEW packages will be installed:
  apt-transport-https gnupg-agent
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 6944 B of archives.
After this operation, 206 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-transport-https all 2.0.2ubuntu0.1 [1708 B]
Get:2 http://archive.ubuntu.com/ubuntu focal/universe amd64 gnupg-agent all 2.2.19-3ubuntu2 [5236 B]
Fetched 6944 B in 1s (6589 B/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 93373 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.0.2ubuntu0.1_all.deb ...
Unpacking apt-transport-https (2.0.2ubuntu0.1) ...
Selecting previously unselected package gnupg-agent.
Preparing to unpack .../gnupg-agent_2.2.19-3ubuntu2_all.deb ...
Unpacking gnupg-agent (2.2.19-3ubuntu2) ...
Setting up apt-transport-https (2.0.2ubuntu0.1) ...
Setting up gnupg-agent (2.2.19-3ubuntu2) ...
kohei@DESKTOP-Q123T6P:~$
```

3. Docker の official GPG key を追加する。

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

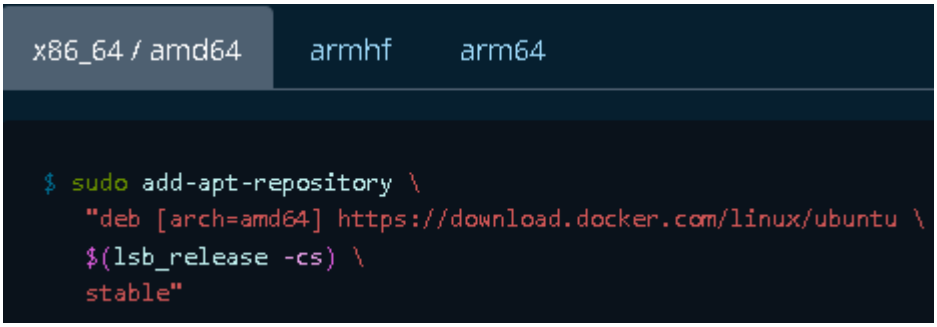
4. fingerprint の最後の 8 文字を検索して、手元の key が `9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88` であることを確認する。

```
sudo apt-key fingerprint 0EBFCD88
```

```
kohei@DESKTOP-Q123T6P:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
kohei@DESKTOP-Q123T6P:~$ sudo apt-key fingerprint 0EBFCD88
pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid           [ unknown] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]

kohei@DESKTOP-Q123T6P:~$
```

5. 次のコマンドを使用して、安定したリポジトリを追加する。nightly または test リポジトリを追加するには、以下のコマンドで stable という単語の後に nightly または test (または両方) という単語を追加する。

今回は、`x86_64/amd64` を使用する。



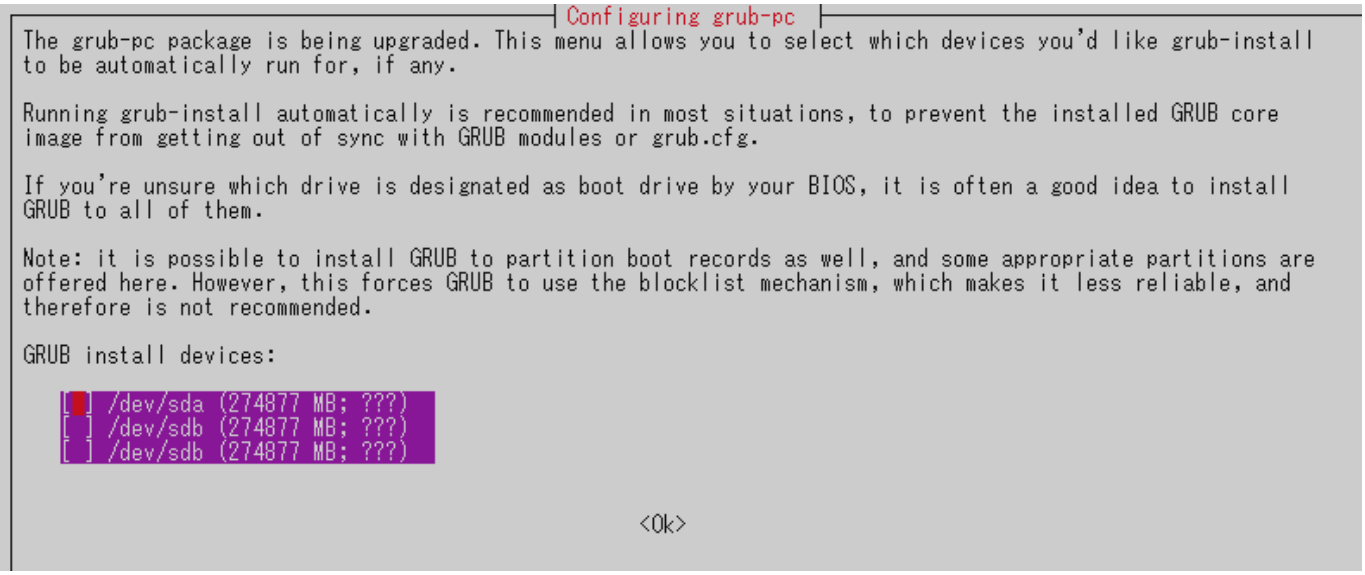

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

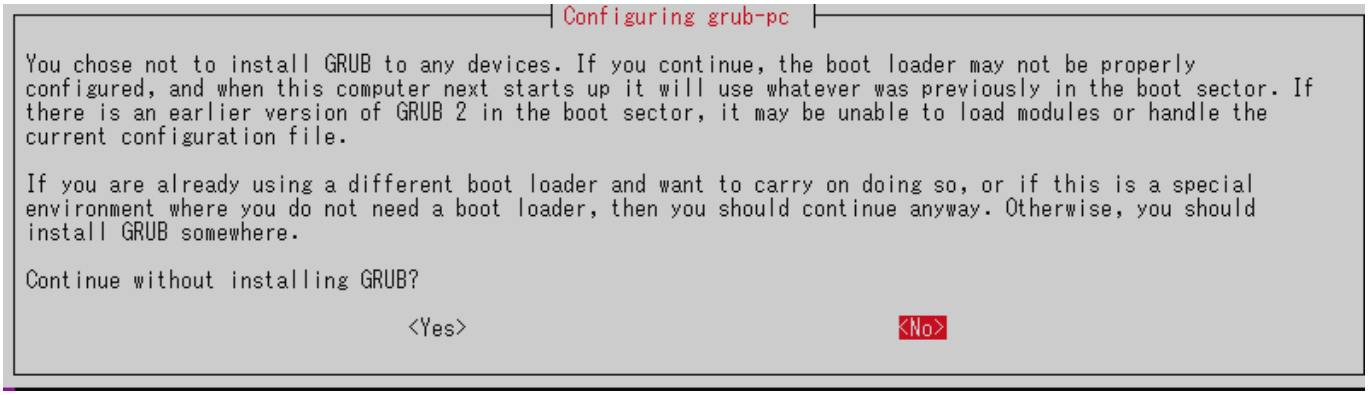
```
kohei@DESKTOP-Q123T6P:~$ sudo add-apt-repository \
>    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
>    $(lsb_release -cs) \
>    stable"
Get:1 file:/var/cuda-repo-ubuntu2004-11-0-local  InRelease
Ign:1 file:/var/cuda-repo-ubuntu2004-11-0-local  InRelease
Get:2 file:/var/cuda-repo-ubuntu2004-11-0-local  Release [564 B]
Get:2 file:/var/cuda-repo-ubuntu2004-11-0-local  Release [564 B]
Get:4 https://download.docker.com/linux/ubuntu focal InRelease [36.2 kB]
Get:5 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [3056 B]
Hit:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:7 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:8 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:9 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Fetched 39.2 kB in 1s (33.2 kB/s)
Reading package lists... Done
```

6. apt パッケージインデックスを更新し、Docker Engine と containerd の最新バージョンをインストールするか、7. の手順に進んで特定のバージョンをインストールする。

```
sudo apt update

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

先ほどのコマンドを実行すると、次のような画面が表示された。





初めて見るエラーだったので、少し調べてみた。

> GRUB
>
> GRUB とは、コンピュータの起動時に最初に読み込まれ、ストレージなどからオペレーティングシステム（OS）を読み込んで起動するブートローダの一つ。GNU プロジェクトが開発・公開しているオープンソースソフトウェアで、よく Linux と組み合わせて用いられる。 [e-Words より]

画面については、以下を参考にした。

Windows10 Update が来たので WSL2 で Ubuntu 20.04 LTS を起動し Docker をインストール

7. Docker Engine の特定のバージョンをインストールするには、利用可能なバージョンをリポジトリにリストしてから、選択してインストールします。

- まず、リポジトリで利用可能なバージョン一覧を表示する。

```
apt-cache madison docker-ce
```

```
kohei@DESKTOP-Q123T6P:~$ apt-cache madison docker-ce
 docker-ce | 5:19.03.12~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.11~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.10~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
 docker-ce | 5:19.03.9~3-0~ubuntu-focal | https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

- 2 番目の列のバージョン文字列 `5:19.03.11~3-0~ubuntu-focal` を使用して、特定のバージョンをインストールする場合は、以下のコマンドの `<VERSION_STRING>` 部分をそのバージョン文字列に置き換えて使用する。

```
sudo apt install docker-ce=<VERSION_STRING> docker-ce-cli=<VERSION_STRING>
containerd.io
```

8. docker イメージを実行するために、docker deamon を起動します。

```
sudo service docker start
```

```
kohei@DESKTOP-Q123T6P:~$ sudo service docker start
 * Starting Docker: docker                                                    [ OK ]
```

9. hello-world イメージを実行して、Docker エンジンが正しくインストールされていることを確認します。

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

```
kohei@DESKTOP-Q123T6P:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:4cf9c47f86df71d48364001ede3a4fcd85ae80ce02ebad74156906caff5378bc
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

kohei@DESKTOP-Q123T6P:~$
```

10. `hello-world` のメッセージが表示されたことを確認したら、docker グループにユーザを追加しておく。docker はデフォルトだと root 権限がないユーザからの操作を受け付けないが、こうすることで、

`sudo` なしに、`docker run` コマンドを使用できるようになる。

- ユーザ追加前の、docker グループを確認する。

```
grep -i docker /etc/group
```

- ユーザを dokcer グループに追加する。ここで、`$USER` は自分がログイン時に使っているユーザ名に変更する必要がある。

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

```
kohei@DESKTOP-Q123T6P:~$ grep -i docker /etc/group
docker:x:999:
kohei@DESKTOP-Q123T6P:~$ sudo usermod -aG docker kohei
kohei@DESKTOP-Q123T6P:~$ grep -i docker /etc/group
docker:x:999:kohei
```

11. グループに入ったことを確認した後、一旦ログアウトを行い、再度ログインする。

```
logout
```

12. docker サービスの開始と自動起動設定をする。

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

sudo systemctl start docker && sudo systemctl enable docker
```

これを実行すると、最後に次のような文章が表示される。

> If you would like to use Docker as a non-root user, you should now consider adding your user to the "docker" group with something like:
>
> sudo usermod -aG docker kohei
>
> Remember that you will have to log out and back in for this to take effect!

適当訳：

Docker を非 root ユーザーとして使用したい場合は、ここで検討する必要があります。 次のようにして、ユーザーを「docker」グループに追加します。

`sudo usermod -aG docker kohei`

これを有効にするには、ログアウトしてから再度ログインする必要があることに注意してください。

> WARNING: Adding a user to the "docker" group will grant the ability to run containers which can be used to obtain root privileges on the docker host. Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface for more information.

適用訳：

警告：「docker」グループにユーザーを追加すると、Docker ホストで root 権限を取得するために使用できるコンテナーを実行する機能が付与されます。 詳細については、
https：//docs.docker.com/engine/security/security/#docker-daemon-attack-surface を参照してください。

```
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

  sudo usermod -aG docker kohei

Remember that you will have to log out and back in for this to take effect!

WARNING: Adding a user to the "docker" group will grant the ability to run
         containers which can be used to obtain root privileges on the
         docker host.
         Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
         for more information.
```

このような警告が表示されたため、一応 docker グループにユーザ登録されているかを確認する。

```
grep -i docker /etc/group
```

```
kohei@DESKTOP-Q123T6P:~$ grep -i docker /etc/group
docker:x:999:kohei
kohei@DESKTOP-Q123T6P:~$
```

しっかりと、登録されたままの状態となっていることを確認。

13. 起動確認

最後に起動確認を行う。

```
docker -ps
```

このようにエラーが出なければ OK

```
kohei@DESKTOP-Q123T6P:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
kohei@DESKTOP-Q123T6P:~$
```

1.5. NVIDIA Container Toolkit のインストール

nvidia の公式ドキュメントのインストールガイド に従い、NVIDIA Container Toolkit のインストールを行う。

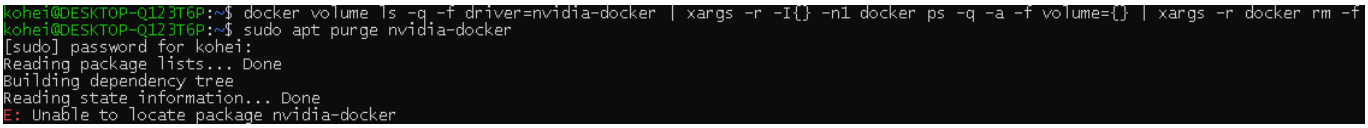
1. Version 1.0 の `nvidia-docker` がインストールされていないかを確認する。

```
docker volume ls -q -f driver=nvidia-docker | xargs -r -I{} -n1 docker ps -q -a -f
volume={} | xargs -r docker rm -f
```

インストールされていれば、以下のコマンドでアンインストールする。

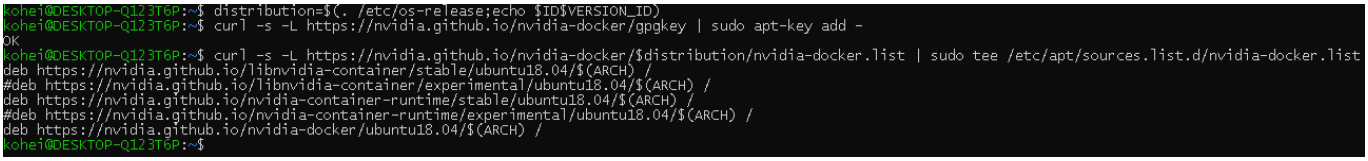
```
sudo apt purge nvidia-docker
```

一応、確認のため実行してみたが、インストールされていなかったので、エラーが出た。



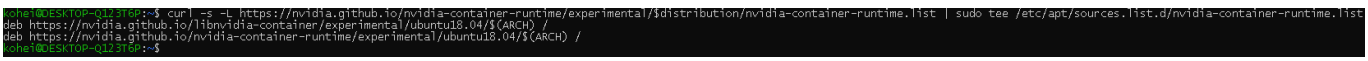
2. 安定したリポジトリと GPG キーを設定する。

```
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
```



> WSL の CUDA や A100 の新しい MIG 機能などの実験的機能にアクセスするには、実験的ブランチをリポジトリリストに追加できます。

```
curl -s -L https://nvidia.github.io/nvidia-container-
runtime/experimental/$distribution/nvidia-container-runtime.list | sudo tee
/etc/apt/sources.list.d/nvidia-container-runtime.list
```



3. パッケージリストを更新した後、nvidia-docker2 パッケージ (および依存関係) をインストールします。

```
sudo apt update

sudo apt install -y nvidia-docker2
```

```
kohei@DESKTOP-Q123T6P:~$ sudo apt update
Get:1 file:/var/cuda-repo-ubuntu2004-11-0-local  InRelease
Ign:1 file:/var/cuda-repo-ubuntu2004-11-0-local  InRelease
Get:2 file:/var/cuda-repo-ubuntu2004-11-0-local  Release [564 B]
Get:2 file:/var/cuda-repo-ubuntu2004-11-0-local  Release [564 B]
Hit:4 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:5 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:6 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:8 https://download.docker.com/linux/ubuntu focal InRelease
Get:9 https://nvidia.github.io/libnvidia-container/experimental/ubuntu18.04/amd64  InRelease [1158 B]
Get:10 https://nvidia.github.io/nvidia-container-runtime/experimental/ubuntu18.04/amd64  InRelease [1149 B]
Get:11 https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/amd64  InRelease [1139 B]
Get:12 https://nvidia.github.io/nvidia-container-runtime/stable/ubuntu18.04/amd64  InRelease [1136 B]
Get:13 https://nvidia.github.io/nvidia-docker/ubuntu18.04/amd64  InRelease [1129 B]
Get:14 https://nvidia.github.io/libnvidia-container/experimental/ubuntu18.04/amd64  Packages [3392 B]
Get:15 https://nvidia.github.io/nvidia-container-runtime/experimental/ubuntu18.04/amd64  Packages [804 B]
Get:16 https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/amd64  Packages [8496 B]
Get:17 https://nvidia.github.io/nvidia-container-runtime/stable/ubuntu18.04/amd64  Packages [5828 B]
Get:18 https://nvidia.github.io/nvidia-docker/ubuntu18.04/amd64  Packages [4180 B]
Fetched 28.4 kB in 3s (9554 B/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
W: Target CNF (stable/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:50 and /etc/apt/sources.list.d/docker.list:1
kohei@DESKTOP-Q123T6P:~$
```

インストール完了！やっとここまで来た……

```
kohei@DESKTOP-Q123T6P:~$ sudo apt install -y nvidia-docker2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnvidia-container-tools libnvidia-container1 nvidia-container-runtime nvidia-container-toolkit
The following NEW packages will be installed:
  libnvidia-container-tools libnvidia-container1 nvidia-container-runtime nvidia-container-toolkit nvidia-docker2
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 1471 kB of archives.
After this operation, 4684 kB of additional disk space will be used.
Get:1 https://nvidia.github.io/libnvidia-container/experimental/ubuntu18.04/amd64  libnvidia-container1 1.3.0~rc.1-1 [66.8 kB]
Get:2 https://nvidia.github.io/libnvidia-container/experimental/ubuntu18.04/amd64  libnvidia-container-tools 1.3.0~rc.1-1 [20.4 kB]
Get:3 https://nvidia.github.io/nvidia-container-runtime/experimental/ubuntu18.04/amd64  nvidia-container-toolkit 1.3.0~rc.2-1 [763 kB]
Get:4 https://nvidia.github.io/nvidia-container-runtime/stable/ubuntu18.04/amd64  nvidia-container-runtime 3.3.0-1 [615 kB]
Get:5 https://nvidia.github.io/nvidia-docker/ubuntu18.04/amd64  nvidia-docker2 2.4.0-1 [5792 B]
Fetched 1471 kB in 1s (1024 kB/s)
Selecting previously unselected package libnvidia-container1:amd64.
(Reading database ... 100906 files and directories currently installed.)
Preparing to unpack .../libnvidia-container1_1.3.0~rc.1-1_amd64.deb ...
Unpacking libnvidia-container1:amd64 (1.3.0~rc.1-1) ...
Selecting previously unselected package libnvidia-container-tools.
Preparing to unpack .../libnvidia-container-tools_1.3.0~rc.1-1_amd64.deb ...
Unpacking libnvidia-container-tools (1.3.0~rc.1-1) ...
Selecting previously unselected package nvidia-container-toolkit.
Preparing to unpack .../nvidia-container-toolkit_1.3.0~rc.2-1_amd64.deb ...
Unpacking nvidia-container-toolkit (1.3.0~rc.2-1) ...
Selecting previously unselected package nvidia-container-runtime.
Preparing to unpack .../nvidia-container-runtime_3.3.0-1_amd64.deb ...
Unpacking nvidia-container-runtime (3.3.0-1) ...
Selecting previously unselected package nvidia-docker2.
Preparing to unpack .../nvidia-docker2_2.4.0-1_all.deb ...
Unpacking nvidia-docker2 (2.4.0-1) ...
Setting up libnvidia-container1:amd64 (1.3.0~rc.1-1) ...
Setting up libnvidia-container-tools (1.3.0~rc.1-1) ...
Setting up nvidia-container-toolkit (1.3.0~rc.2-1) ...
Setting up nvidia-container-runtime (3.3.0-1) ...
Setting up nvidia-docker2 (2.4.0-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...
kohei@DESKTOP-Q123T6P:~$
```

4. デフォルトのランタイムを設定した後、Docker デーモンを再起動してインストールが完了する。

```
sudo systemctl restart docker
```

しかし、ここにきてエラー….

```
kohei@DESKTOP-Q123T6P:~$ sudo systemctl restart docker
System has not been booted with systemd as init system (PID 1). Can't operate.
Failed to connect to bus: Host is down
```

1.6. PID 1 を `systemd` に変更する

インストールは上手くいってそうなので、別の理由かと思いつつ、調べてみた。

参考：【WSL2】systemctl が動かない問題をきちんと解決する

```
System has not been booted with systemd as init system (PID 1). Can't operate.
Failed to connect to bus: Host is down
```

機械翻訳：

システムが systemd で init システム（PID 1）として起動されていません。 操作できません。 バスに接続できませんでした：ホストがダウンしています。

これは、`systemd` が `PID 1 (init system)` ではないので操作できないよ。みたいな感じ。

参考サイトによると

> Linux が起動するとき、すべてのプロセスの最初として「systemd」が起動し、すべてのプロセスの親としてふるまう。

実際に以下のコマンドで、WSL2 上のプロセスを確認してみると、PID 1 が `/init` になっており `systemd` ではないことが確認できる。

```
ps aux
```

これが、原因で `systemd` が動作していない模様...

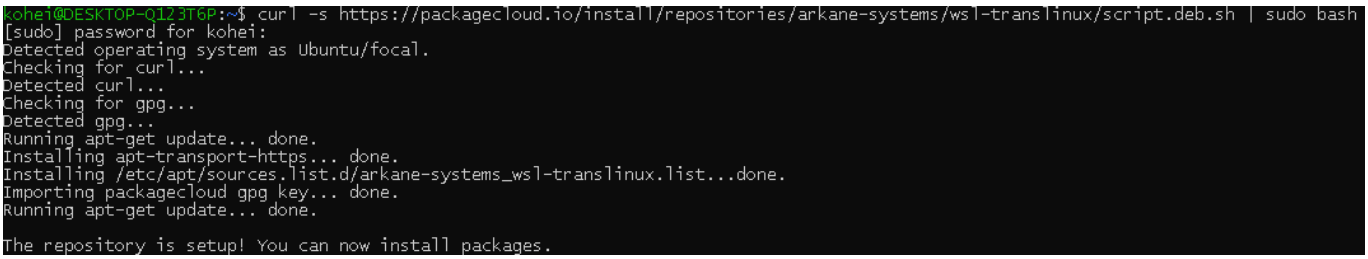
そこで、以下のリポジトリを用いてこの問題を解決する。

https://github.com/arkane-systems/genie

※ この方法は、WSL2 でのみ適用可能。

1. パッケージを追加する。

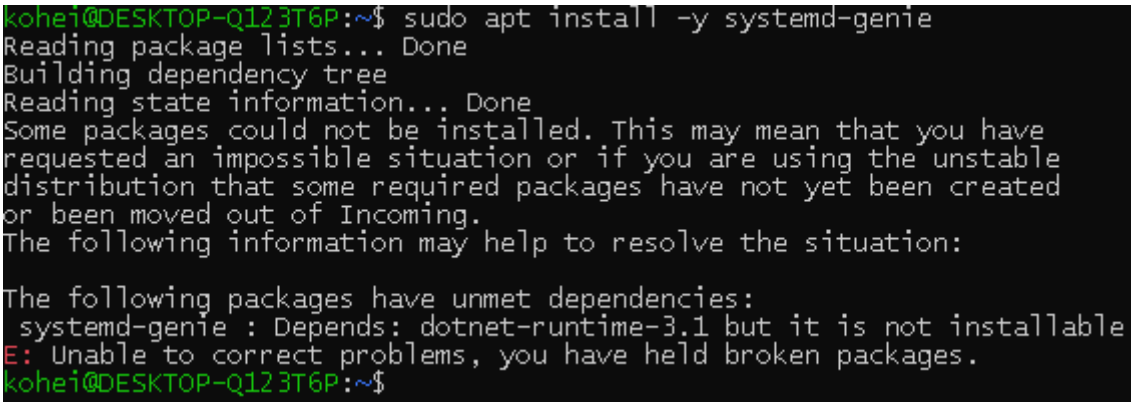
```
curl -s https://packagecloud.io/install/repositories/arkane-systems/wsl-translinux/script.deb.sh | sudo bash
```



2. 以下のコマンドでインストールする。

```
sudo apt install -y systemd-genie
```

すると、次のようなエラーが表示される。




```
The following packages have unmet dependencies:
systemd-genie : Depends: dotnet-runtime-3.1 but it is not installable

E: Unable to correct problems, you have held broken packages.
```

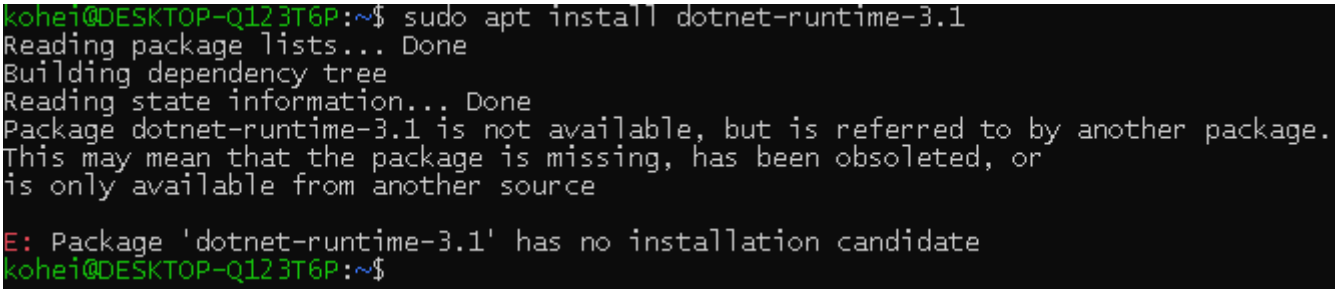
機械翻訳：

> 次のパッケージには、満たされていない依存関係があります。
>
> systemd-genie：依存：dotnet-runtime-3.1 ですが、インストールできません。
>
> E：問題を修正できません。パッケージが壊れています。

そこで、依存関係の `dotnet-runtime-3.1` をインストールする。

```
sudo apt install dotnet-runtime-3.1
```

しかし、次のようなエラーが表示される。



```
Package dotnet-runtime-3.1 is not available, but is referred to by another
package.
This may mean that the package is missing, has been obsoleted, or is only
available from another source

E: Package 'dotnet-runtime-3.1' has no installation candidate
```

機械翻訳：

> パッケージ dotnet-runtime-3.1 は使用できませんが、別のパッケージによって参照されています。 これは、パッケージが見つからないか、廃止されているか、別のソースからしか入手できないことを意味します。
>
> E：パッケージ 'dotnet-runtime-3.1'にはインストール候補がありません。

なんで、みんな落とせてるのよ....

Microsoft の Ubuntu に .NET Core SDK または .NET Core ランタイムをインストールする を参考に `dotnet-runtime-3.1` をインストールする。

- .Net をインストールする前に、次のコマンドを実行して、信頼された key の一覧に Microsoft パッケージ署名 key を追加し、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
```

- `.Net Core` ランタイムを使用すると、ランタイムを含まない `.Net Core` を使用して作成されたアプリを実行することができる。次のコマンドを実行すると、 `.Net Core` の最も互換性の高いランタイムである `ASP .Net Core` ランタイムをインストールできる。

```
sudo apt update; \
  sudo apt install -y apt-transport-https && \
  sudo apt update && \
  sudo apt install -y aspnetcore-runtime-3.1
```

これは、目的の `.Net Core runtime-3.1` ではないが、動作するか確認したかったので、一度インストールを試みた。

インストールは成功。

```
Selecting previously unselected package dotnet-host.
(Reading database ... 100944 files and directories currently installed.)
Preparing to unpack .../dotnet-host_3.1.8-1_amd64.deb ...
Unpacking dotnet-host (3.1.8-1) ...
Selecting previously unselected package dotnet-hostfxr-3.1.
Preparing to unpack .../dotnet-hostfxr-3.1_3.1.8-1_amd64.deb ...
Unpacking dotnet-hostfxr-3.1 (3.1.8-1) ...
Selecting previously unselected package dotnet-runtime-deps-3.1.
Preparing to unpack .../dotnet-runtime-deps-3.1_3.1.8-1_amd64.deb ...
Unpacking dotnet-runtime-deps-3.1 (3.1.8-1) ...
Selecting previously unselected package dotnet-runtime-3.1.
Preparing to unpack .../dotnet-runtime-3.1_3.1.8-1_amd64.deb ...
Unpacking dotnet-runtime-3.1 (3.1.8-1) ...
Selecting previously unselected package aspnetcore-runtime-3.1.
Preparing to unpack .../aspnetcore-runtime-3.1_3.1.8-1_amd64.deb ...
Unpacking aspnetcore-runtime-3.1 (3.1.8-1) ...
Setting up dotnet-host (3.1.8-1) ...
Setting up dotnet-runtime-deps-3.1 (3.1.8-1) ...
Setting up dotnet-hostfxr-3.1 (3.1.8-1) ...
Setting up dotnet-runtime-3.1 (3.1.8-1) ...
Setting up aspnetcore-runtime-3.1 (3.1.8-1) ...
Processing triggers for man-db (2.9.1-1) ...
kohei@DESKTOP-Q123T6P:~$
```

先ほど失敗した、`genie` のインストールも試してみる。

```
sudo apt install -y systemd-genie
```

結果、インストールに成功した。

```
kohei@DESKTOP-Q123T6P:~$ sudo apt install -y systemd-genie
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  daemonize
The following NEW packages will be installed:
  daemonize systemd-genie
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 208 kB of archives.
After this operation, 705 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 daemonize amd64 1.7.8-1 [11.9 kB]
Get:2 https://packagecloud.io/arkane-systems/wsl-translinux/ubuntu focal/main amd64 systemd-genie amd64 1.28 [197 kB]
Fetched 208 kB in 1s (205 kB/s)
Selecting previously unselected package daemonize.
(Reading database ... 101296 files and directories currently installed.)
Preparing to unpack .../daemonize_1.7.8-1_amd64.deb ...
Unpacking daemonize (1.7.8-1) ...
Selecting previously unselected package systemd-genie.
Preparing to unpack .../systemd-genie_1.28_amd64.deb ...
Unpacking systemd-genie (1.28) ...
Setting up daemonize (1.7.8-1) ...
Setting up systemd-genie (1.28) ...
Processing triggers for man-db (2.9.1-1) ...
kohei@DESKTOP-Q123T6P:~$
```

3. インストールした `genie` を使って PID 1 を変更する。

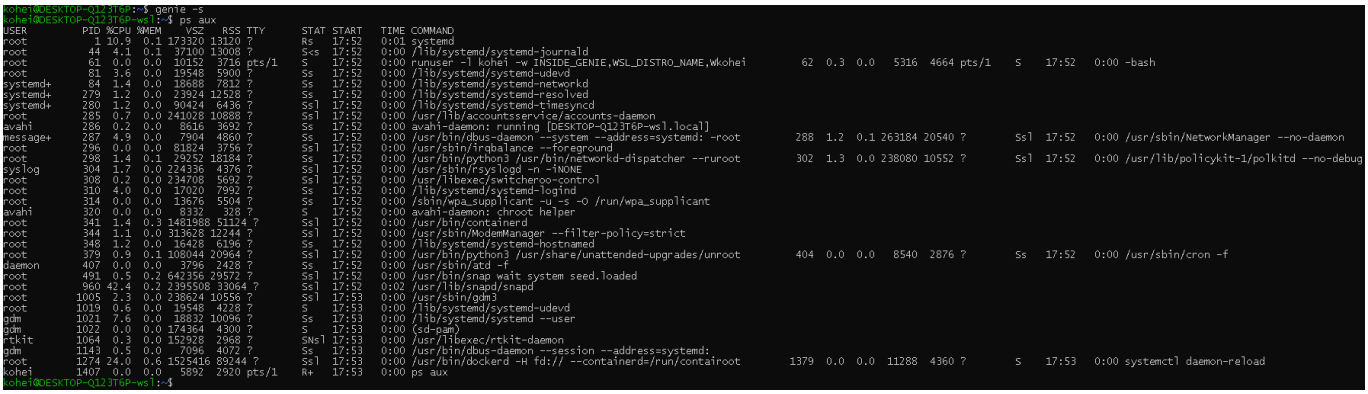
`genie` をインストールしただけでは、PID に変化はなかった。

```
kohei@DESKTOP-Q123T6P:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0    892   572 ?        Sl   10:40   0:00 /init
root          49  0.0  0.0    900    88 ?        S    10:41   0:00 /init
root        1174  0.5  0.7 1895772 101856 ?      Sl   15:35   0:48 /usr/bin/dockerd -p /var/run/docker.pid
root        1193  0.5  0.4 1704560 55508 ?       Ssl  15:35   0:43 containerd --config /var/run/docker/containerd/conta
root        2516  0.0  0.0    900    80 ?        Ss   15:57   0:00 /init
root        2517  0.0  0.0    900    88 ?        R    15:57   0:00 /init
kohei       2518  0.0  0.0  10056  5148 pts/1    Ss   15:57   0:00 -bash
kohei       8616  0.0  0.0  10612  3364 pts/1    R+   17:49   0:00 ps aux
kohei@DESKTOP-Q123T6P:~$
```

そのために、以下のコマンドを実行する。

```
genie -s
```

このコマンドを実行すると、PID 1 がしっかりと `systemd` に変化した。

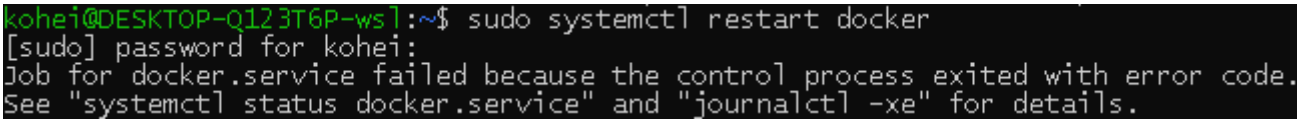


1.7. Docker daemon を再起動

先ほどできなかった、`docker daemon`の再起動を行う。

```
sudo systemctl restart docker
```

結果は、別のエラー....



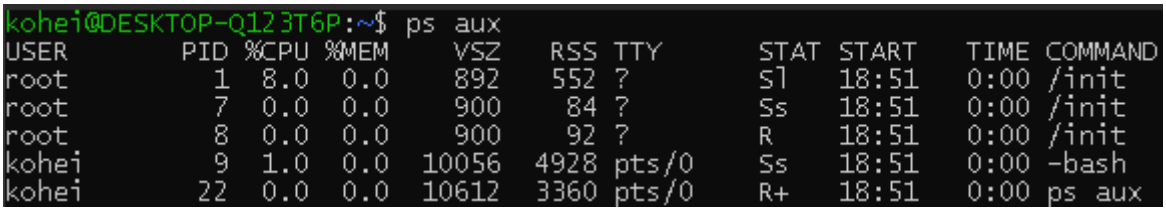

```
Job for docker.service failed because the control process exited with error code.
See "systemctl status docker.service" and "journalctl -xe" for details.
```

機械翻訳：

> 制御プロセスがエラーコードで終了したため、docker.service のジョブが失敗しました。
>
> 詳細については、「systemctl status docker.service」および「journalctl -xe」を参照してください。

docker のリスタートが行いたかったので、ここで一度再起動してみることに。

しかし、再起動すると PID 1 がまた `init` に戻ることが判明！



そのため、WSL2 では起動の度に

```
genie -s
```

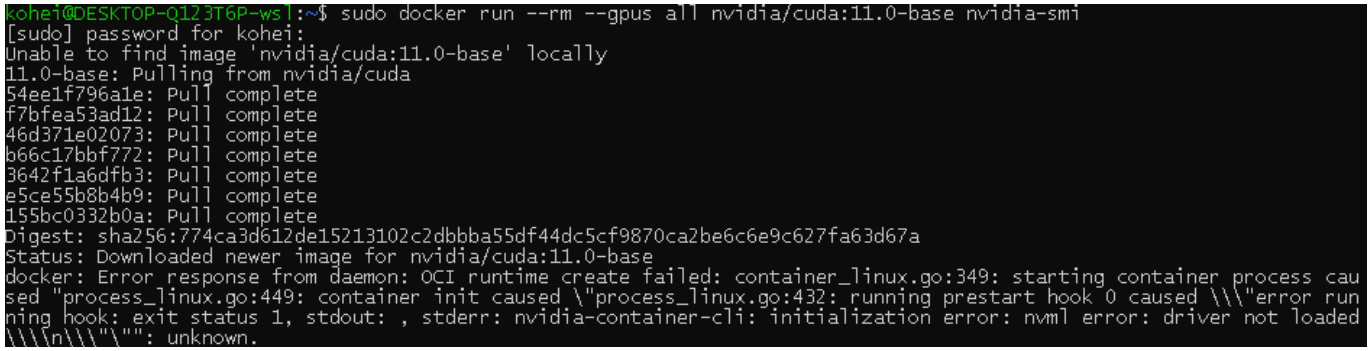
コマンドを打たなければならない。

1.8. CUDA コンテナの実行

ベースの CUDA コンテナーを実行することで、機能しているセットアップをテストする。

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

しかし、次のようなエラーが発生。




```
docker: Error response from daemon: OCI runtime create failed:
container_linux.go:349: starting container process caused "process_linux.go:449:
container init caused \"process_linux.go:432: running prestart hook 0 caused
\\\"error running hook: exit status 1, stdout: , stderr: nvidia-container-cli:
initialization error: nvml error: driver not loaded\\\\n\\\"\"": unknown.
```

機械翻訳：

> docker：デーモンからのエラー応答：OCI ランタイムの作成に失敗しました：container_linux.go：
>
> 349：コンテナープロセスの開始により「process_linux.go：
>
> 449：コンテナーの初期化により\ "process_linux.go：
>
> 432：実行前開始フック 0 により\"エラーが発生しました。
>
> 実行中のフック：終了ステータス 1、標準出力：、標準エラー：nvidia-container-cli：初期化エラー：nvml エラー：ドライバーが読み込まれていません\\ n \\ "" "：不明です。

ドライバーが読み込まれていませんという文があることから、`nvidia-driver` が必要なのではないかと考えた。

そこで、以下のコマンドを実行する。

```
nvidia-sim
```

すると、次のようなメッセージが表示された。

```
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make
sure that the latest NVIDIA driver is installed and running.
```

機械翻訳：

> NVIDIA-SMI は、NVIDIA ドライバーと通信できなかったために失敗しました。 最新の NVIDIA ドライバーがインストールされ、実行されていることを確認します。

1.9. NVIDIA Driver のインストール

参考

- ubuntu18.04 に nvidia ドライバを入れるの苦労した
- Ubuntu 20.04 セットアップ

NVIDIA Driver のインストールには、いくつか方法がある。

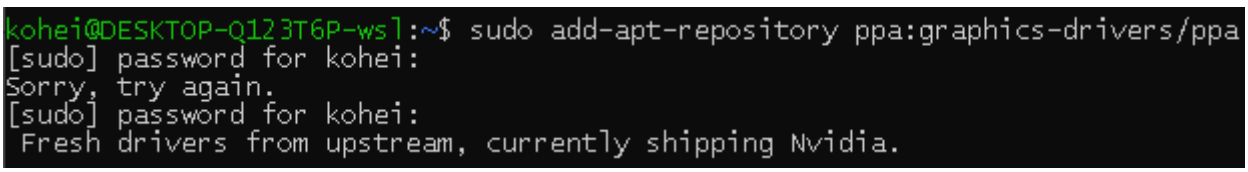
1. `ubuntu-drivers` コマンドでインストールする
2. `software & Updates` の `Additional Drivers` からインストールする
3. NVIDIA 公式からドライバをダウンロードしてインストールする。

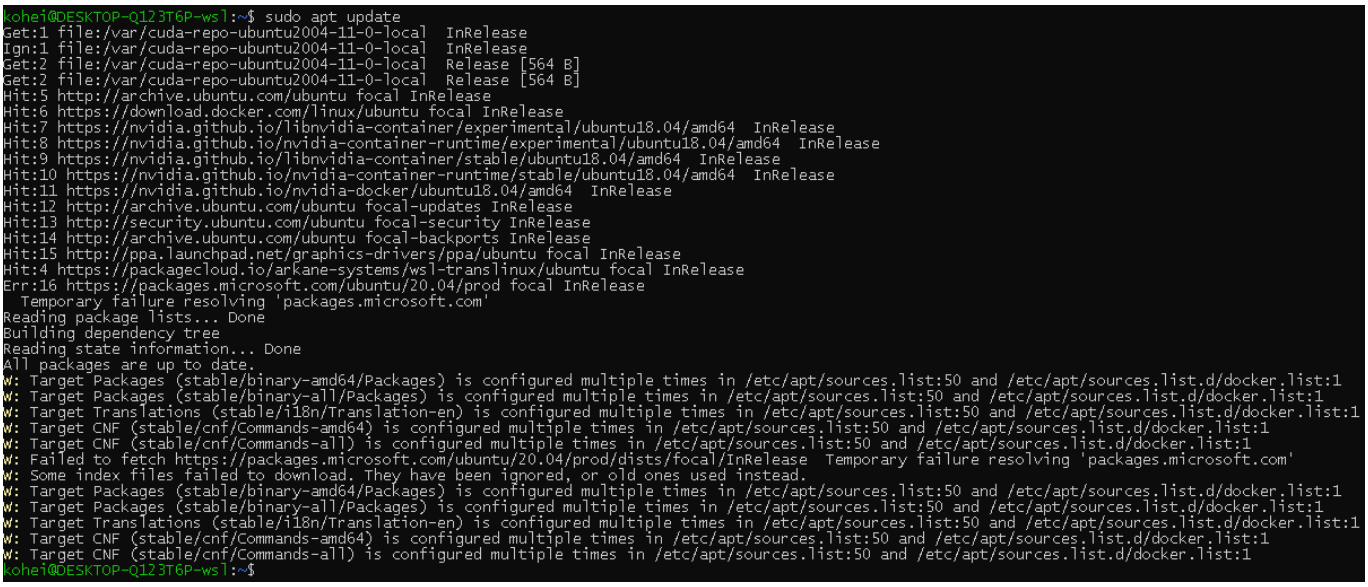
ここでは、1. の方法でインストールする。

(Optional) PPA を追加する

```
sudo add-apt-repository ppa:graphics-drivers/ppa

sudo apt update
```

途中で一度 `Enter` を押す場所がある。





(Optional) 推奨ドライバを確認する。

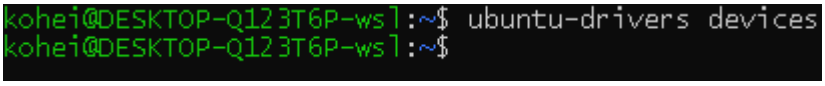
推奨ドライバを自動的に確認するために、依存パッケージを先にインストールする。

```
sudo apt install -y ubuntu-drivers-common
```

推奨ドライバを確認する。

```
ubuntu-drivers devices
```

このコマンドを実行することで、推奨ドライバを確認することができるはずだが、自分のパソコンでは表示されなかった。



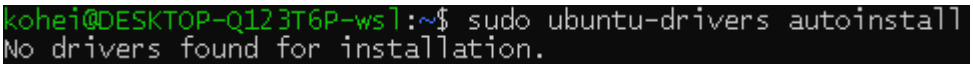
しかし、今回はこれを無視して次の手順に進む。

ドライバをインストールする

自動で推奨ドライバをインストールする。

```
sudo ubuntu-drivers autoinstall
```

結果は、ドライバを発見することができなかった。



既存のドライバが入ってるから…？

既存ドライバのアンインストール

以前インストールしていたドライバ・CUDA をアンインストールする。

```
sudo apt --purge remove nvidia-*

sudo apt-get --purge remove cuda-*
```

どうも、nvidia driver はインストールされていなかったみたい....

```
kohei@DESKTOP-Q123T6P-wsl:~$ sudo apt --purge remove nvidia-*
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package nvidia-*
kohei@DESKTOP-Q123T6P-wsl:~$ sudo apt-get --purge remove cuda-*
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package cuda-repo-ubuntu2004-11-0-local_11.0.3-450.51.06-1_amd64.deb
E: Couldn't find any package by glob 'cuda-repo-ubuntu2004-11-0-local_11.0.3-450.51.06-1_amd64.deb'
E: Couldn't find any package by regex 'cuda-repo-ubuntu2004-11-0-local_11.0.3-450.51.06-1_amd64.deb'
kohei@DESKTOP-Q123T6P-wsl:~$
```

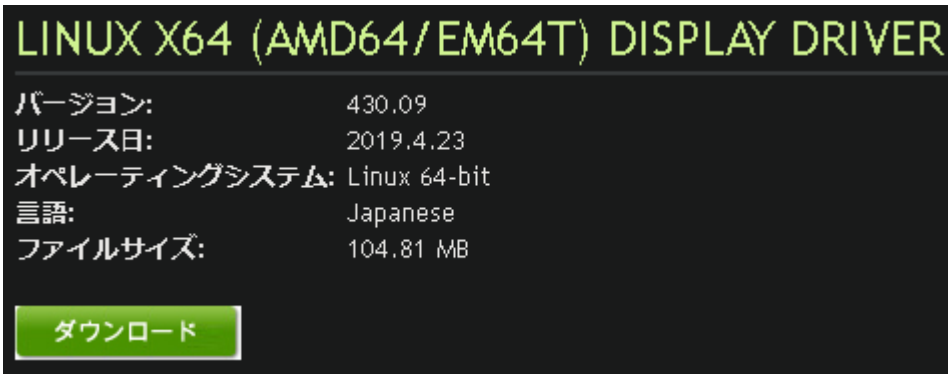
全てアンインストールしたが、`autoinstall` できなかった。

```
kohei@DESKTOP-Q123T6P-wsl:~$ sudo ubuntu-drivers autoinstall
No drivers found for installation.
```

NVIDIA の公式から `Driver` をインストールする

`NVIDIA` 公式 に自分の製品情報を入力する。

そして、「検索ボタン」を押すと、自分の製品に最適な Driver の version が表示される。



このバージョンを参考に、Driver をインストールする。

```
sudo add-apt-repository ppa:graphics-drivers/ppa

sudo apt update

sudo apt install nvidia-driver-430
```

インストールすることができた。

```
kohei@DESKTOP-Q123T6P-wsl:~$ sudo apt install nvidia-driver-430
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  nvidia-driver-440
The following NEW packages will be installed:
  nvidia-driver-430 nvidia-driver-440
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 12.9 kB of archives.
After this operation, 38.9 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 nvidia-driver-430 amd64 440.100-0ubuntu0.20.04.1 [7340 B]
Get:2 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu focal/main amd64 nvidia-driver-440 amd64 450.66-0ubuntu0.20.04.1 [5516 B]
Fetched 12.9 kB in 1s (23.8 kB/s)
Selecting previously unselected package nvidia-driver-440.
(Reading database ... 101468 files and directories currently installed.)
Preparing to unpack .../nvidia-driver-440_450.66-0ubuntu0.20.04.1_amd64.deb ...
Unpacking nvidia-driver-440 (450.66-0ubuntu0.20.04.1) ...
Selecting previously unselected package nvidia-driver-430.
Preparing to unpack .../nvidia-driver-430_440.100-0ubuntu0.20.04.1_amd64.deb ...
Unpacking nvidia-driver-430 (440.100-0ubuntu0.20.04.1) ...
Setting up nvidia-driver-440 (450.66-0ubuntu0.20.04.1) ...
Setting up nvidia-driver-430 (440.100-0ubuntu0.20.04.1) ...
kohei@DESKTOP-Q123T6P-wsl:~$
```

Driver をインストールしても、エラーは消えなかった。

```
kohei@DESKTOP-Q123T6P-wsl:~$ nvidia-smi
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running.
```

CUDA Toolkit を再インストールする

CUDA がインストールされていないかを確認する。

```
nvcc -V
```

```
kohei@DESKTOP-Q123T6P-wsl:~$ nvcc -V
Command 'nvcc' not found, but can be installed with:
sudo apt install nvidia-cuda-toolkit
kohei@DESKTOP-Q123T6P-wsl:~$
```

メッセージに書かれている通りに、インストールする

```
sudo apt install nvidia-cuda-toolkit
```

再インストール完了。

```
kohei@DESKTOP-Q123T6P-wsl:~$ sudo apt install nvidia-cuda-toolkit
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

先ほどのコマンドを再度実行し、`CUDA-Toolkit` がインストールできたかを確認。

```
nvcc -V
```

version が表示されたことから、インストールを確認。

```
kohei@DESKTOP-Q123T6P-wsl:~$ 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
kohei@DESKTOP-Q123T6P-wsl:~$
```

しかし、やはり動作しない。

```
kohei@DESKTOP-Q123T6P-wsl:~$ nvidia-smi
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running.
```

仕方がないので今日はここまで…