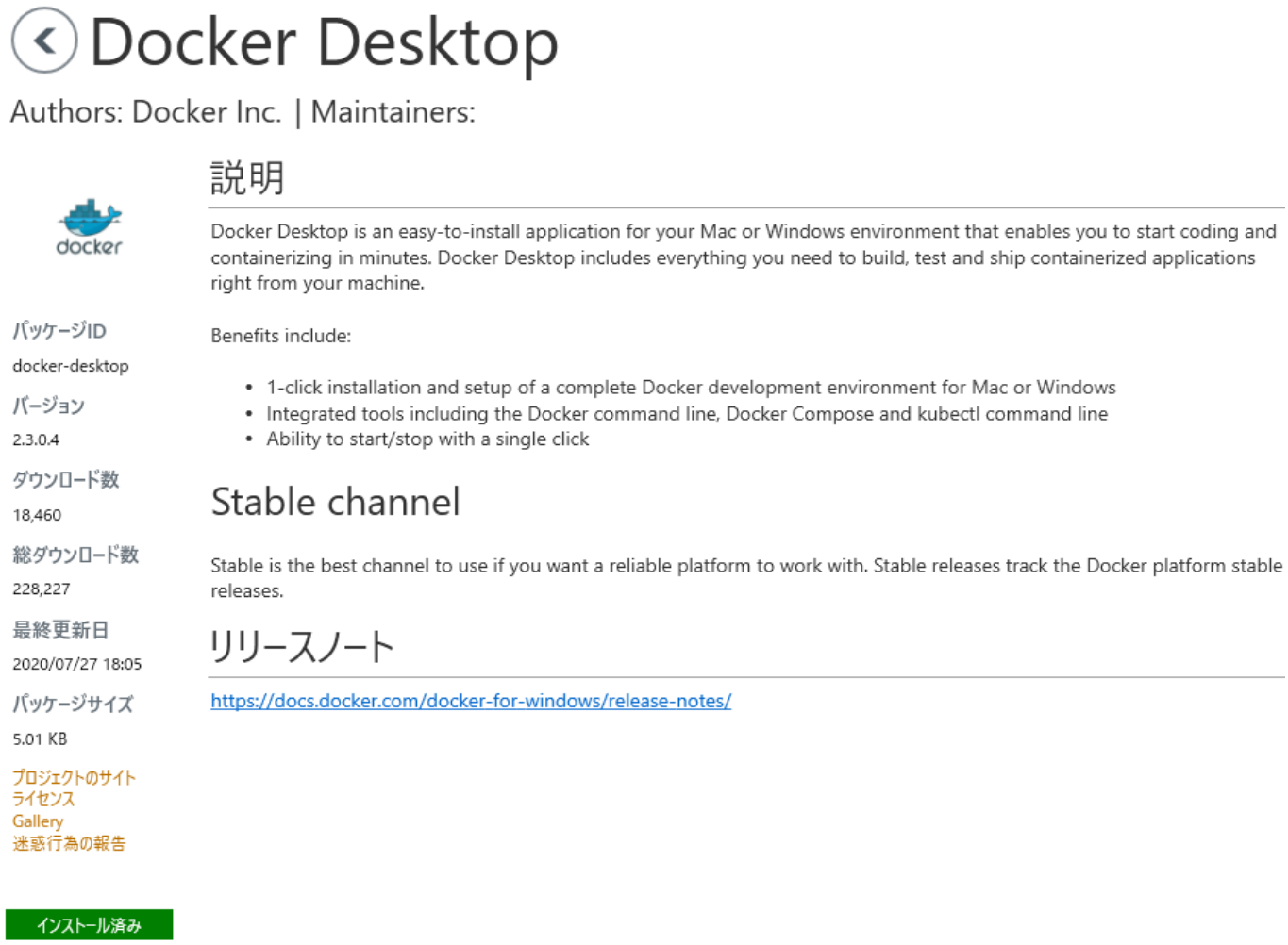
WSL + Docker の開発環境

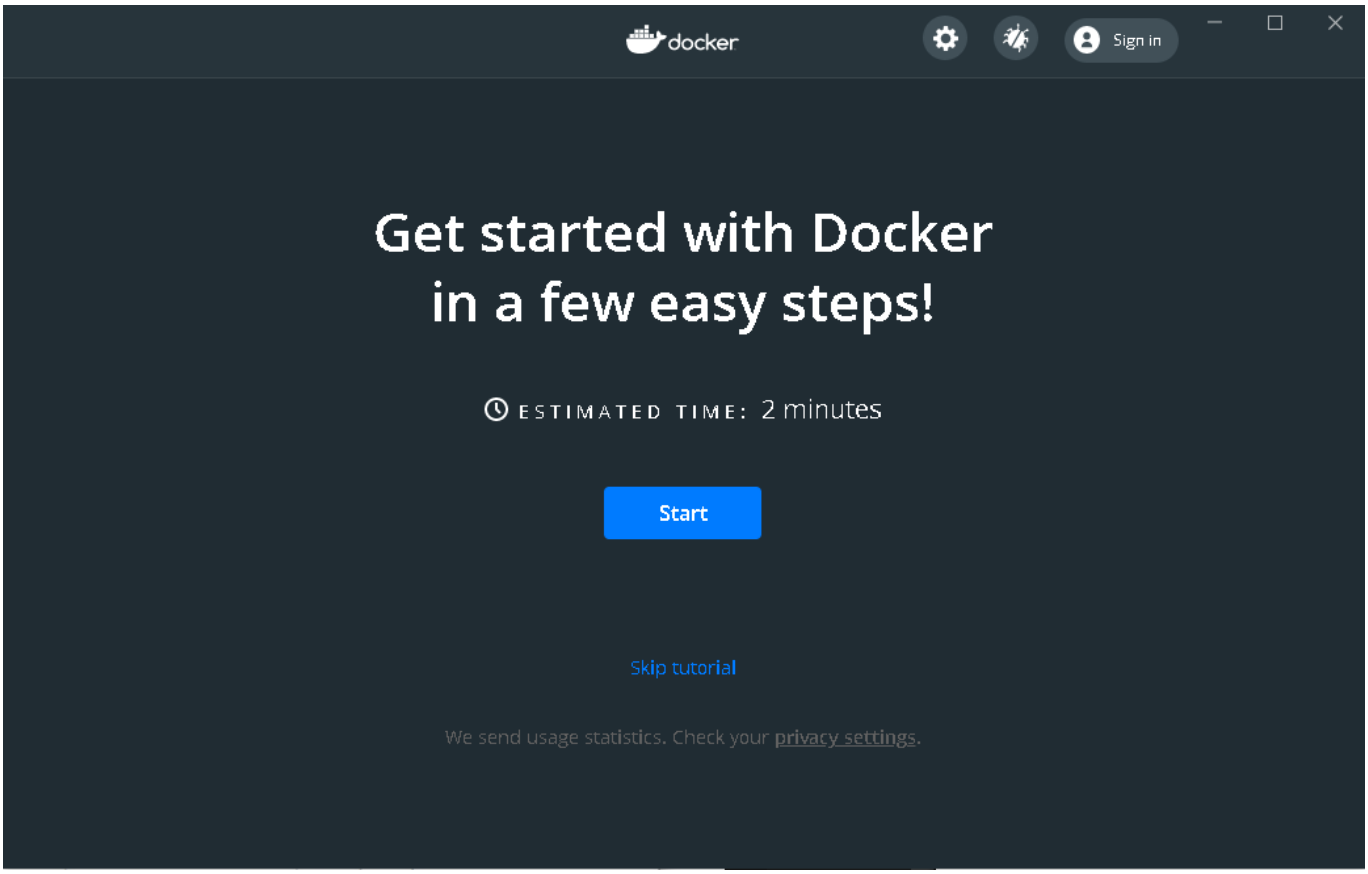
先日まで、WSL +Docker + GPU で開発環境構築を試みてみたが、`Windows Insider Preview` の `Dev チャネル` で新しくインストールされる `Build version` がうまく動作しなかったので、今回は、`Docker Desktop` を用いて構築を行う。

`Chocolatey` で `Docker Desktop` をインストール

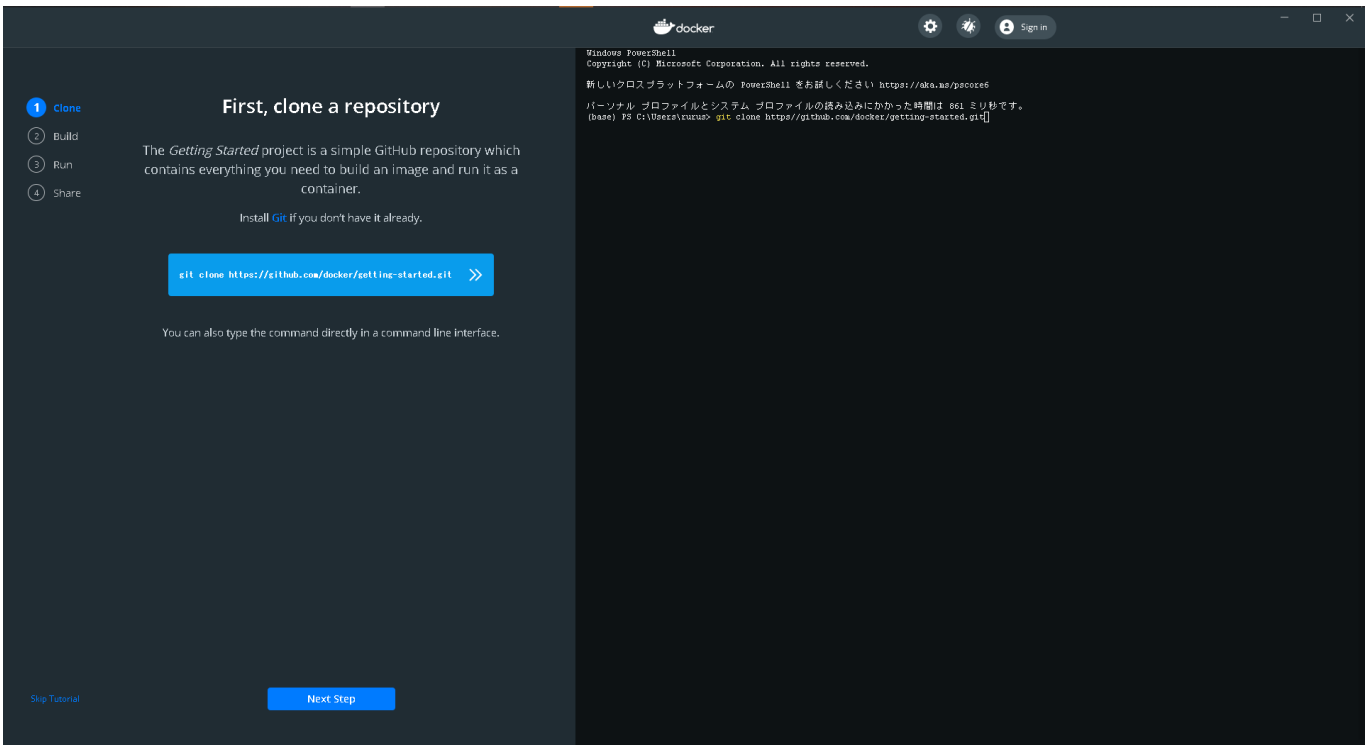
`Chocolatey` で、`Docker Desktop` をインストールする。



WSL2 を導入した状態で `Docker Desktop` をインストールすると、次の画面が表示される。



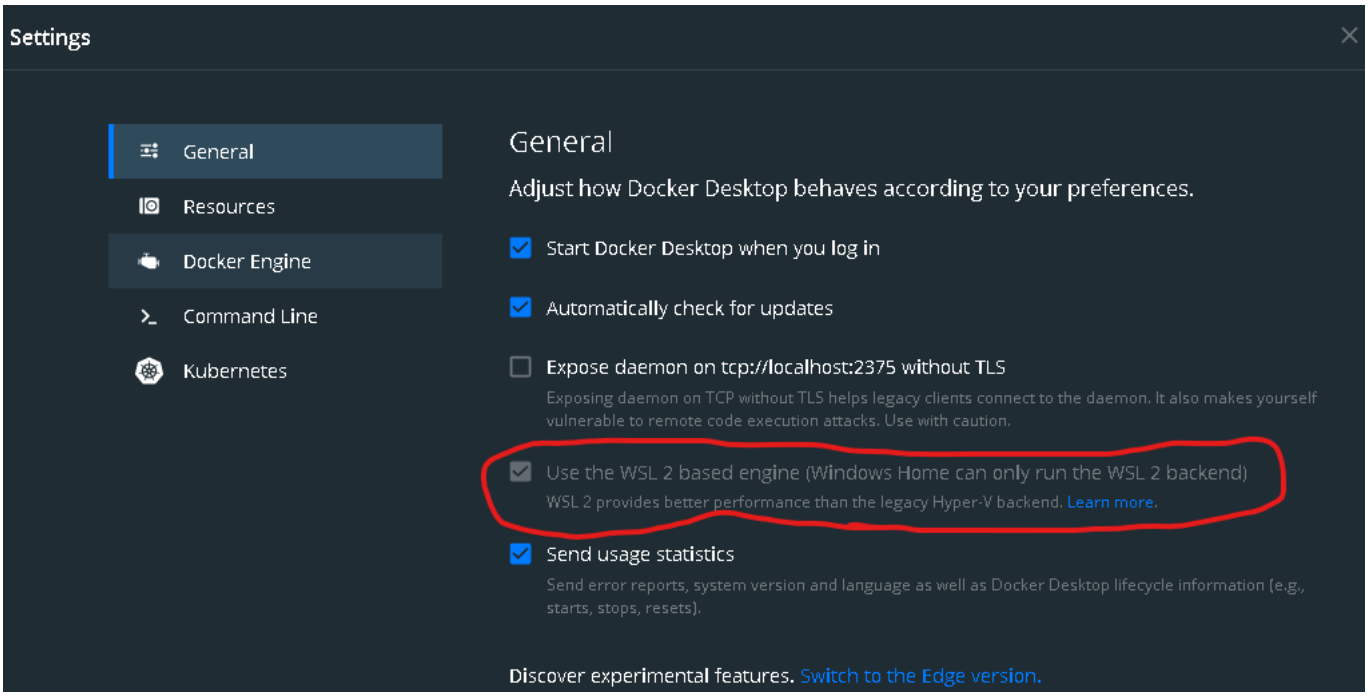
スタートを押すと、チュートリアルが開始される。

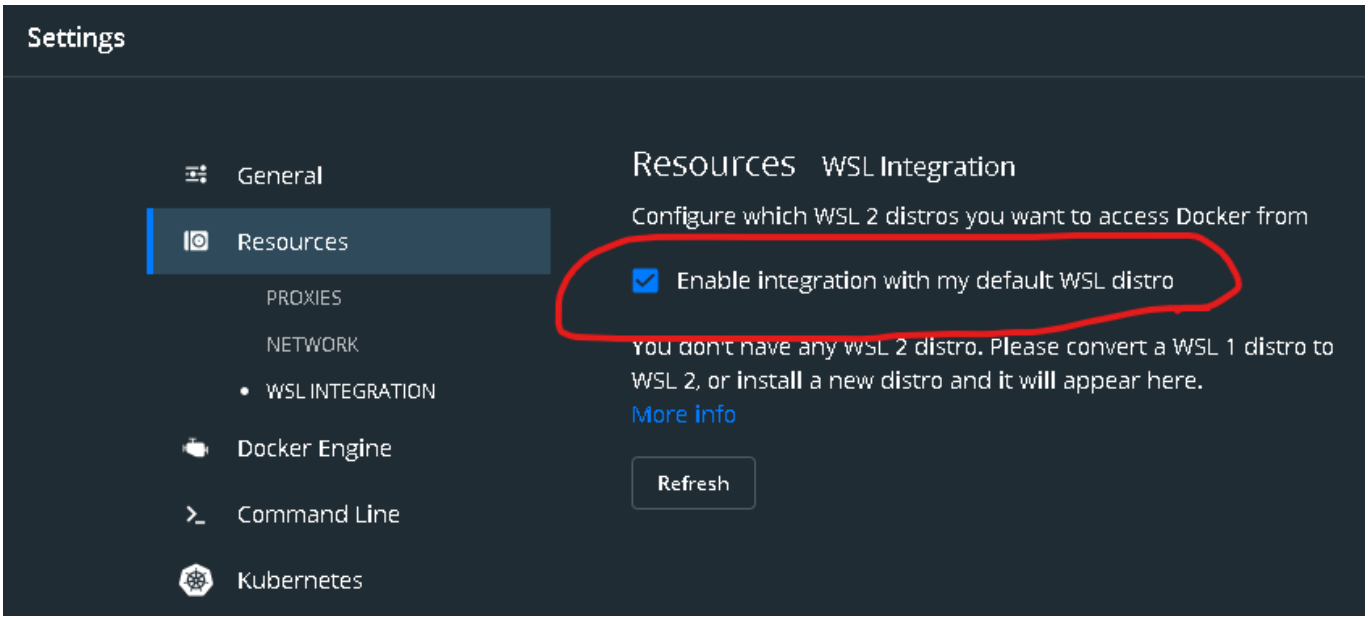


チュートリアルはスキップすることができる。

Docker for Desktop の確認

次の設定が有効になっていることを確認する。





Docker image を取得して動かす

今回は `Docker Hub` にある `Ubuntu` リポジトリからコンテナイメージを取得する。

```
docker pull ubuntu
```

```
Using default tag: latest
latest: Pulling from library/ubuntu
54ee1f796a1e: Pull complete
f7bfea53ad12: Pull complete
46d371e02073: Pull complete
b66c17bbf772: Pull complete
Digest: sha256:31dfb10d52ce76c5ca0aa19d10b3e6424b830729e32a89a7c6eee2cda2be67a5
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
PS C:\Users\rurus>
```

再度、イメージを確認する。

```
docker images
```

めっちゃあるやん！！！！！

今日一の衝撃！！！

```
PS C:\Users\rurus> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker101tutorial   latest              8ae76877fbf5        4 hours ago         27.3MB
<none>              <none>              75184c724eb8        4 hours ago         85.5MB
<none>              <none>              14fff1a2290a        4 hours ago         72MB
<none>              <none>              29c7ead0c654        4 hours ago         224MB
python              alpine              0f03316d4a27        6 days ago          42.7MB
ubuntu              latest              4e2eef94cd6b        3 weeks ago         73.9MB
nginx               alpine              6f715d38cfe0        4 weeks ago         22.1MB
node                12-alpine           18f4bc975732        6 weeks ago         89.3MB
```

必要ないイメージをアンインストールする。

Docker image のアンインストール

Dockerイメージとコンテナの削除方法 を参考に `Docker Container` をアンインストールする。

動いているコンテナの確認

```
docker ps
```

```
PS C:\Users\rurus> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

停止しているコンテナの確認

```
docker ps -a
```

Build されたイメージはなかったので、元のイメージの削除に移る。

```
PS C:\Users\rurus> docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

コンテナの削除

Build されたイメージがあった場合、次のコマンドを使用し、コンテナを削除する。

```
docker rm [コンテナ ID]
```

```
PS C:\Users\rurus> docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
5979003bcdff        ubuntu              "/bin/bash"         49 minutes ago      Exited (127) 27 minutes ago                     frosty_jepsen
PS C:\Users\rurus> docker rm 5979003bcdff
5979003bcdff
PS C:\Users\rurus> docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

イメージの削除

- まず次のコマンドで、インストールされているイメージを確認する。

```
docker images
```

```
PS C:\Users\rurus> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker101tutorial   latest              8ae76877fbf5        4 hours ago         27.3MB
<none>              <none>              75184c724eb8        4 hours ago         85.5MB
<none>              <none>              14fff1a2290a        4 hours ago         72MB
<none>              <none>              29c7ead0c654        4 hours ago         224MB
python              alpine              0f03316d4a27        6 days ago          42.7MB
ubuntu              latest              4e2eef94cd6b        3 weeks ago         73.9MB
nginx               alpine              6f715d38cfe0        4 weeks ago         22.1MB
node                12-alpine           18f4bc975732        6 weeks ago         89.3MB
```

- 次に以下のコマンドで、イメージを削除する。

```
docker rmi [イメージ ID]
```

実際にイメージの削除を行う。

```
PS C:\Users\rurus> docker rmi 8ae76877fbf5
Untagged: docker101tutorial:latest
Deleted: sha256:8ae76877fbf5febfdb428703772d9e42d86e70d4a44137f986732e5e24503bb7
Deleted: sha256:bc962b50e38c2d138f94c9ddb278685b66e77d03ee20b1e9e48ef4ae32674d9d
Deleted: sha256:aa44d5e8cab6f20d0d95b72bbb90e7f4a8e710548929d0f7ce87702cbab2726c
Deleted: sha256:3f5d9452f317be3641f991bec3f86c37319929b4172833ffb2b42db342d5be49
```

- 最後に、イメージが削除されたかを確認する。

```
PS C:\Users\rurus> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
python              alpine              0f03316d4a27        6 days ago          42.7MB
ubuntu              latest              4e2eef94cd6b        3 weeks ago         73.9MB
nginx               alpine              6f715d38cfe0        4 weeks ago         22.1MB
node                12-alpine           18f4bc975732        6 weeks ago         89.3MB
```

この手順を繰り返し、全てのイメージを削除する。

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
PS C:\Users\rurus>
```

再度イメージをダウンロードし、セットアップする

```
PS C:\Users\rurus> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              4e2eef94cd6b        3 weeks ago         73.9MB
```

取得したイメージから、コンテナを作成する。

```
docker run -it ubuntu
```

上記のコマンドを実行すると作成と同時にコンテナも起動し、そのタイミングで自動的にログインする。

ログイン状態を抜けるには `control` を押した状態で `P, Q` を順番に押す。この操作を「dettach (デタッチ)」という。

デタッチした後に再度 `docker ps` コマンドで稼働状況を確認する。

```
PS C:\Users\rurus> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
5979003bcdff        ubuntu              "/bin/bash"         5 minutes ago       Up 5 minutes                            frosty_jepsen
```

STATUS が UP になっていることが確認できる。

先ほど dettach した ubuntu コンテナに再度ログインするには、以下のコマンドを実行する。

```
docker attach <CONTAINER ID または NAME>
```

```
PS C:\Users\rurus> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
5979003bcdff        ubuntu              "/bin/bash"         20 minutes ago      Up 20 minutes                           frosty_jepsen
PS C:\Users\rurus> docker attach frosty_jepsen
root@5979003bcdff:/#
```

コンテナを停止するには、コンテナ上で以下のコマンドを使用する。

```
exit
```

稼働中のコンテナー覧に表示されなくなった。

```
root@5979003bcdff:/# exit
exit
PS C:\Users\rurus> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

bash の有無

Docker image を run するときに bash を付けるか付けないかによって、コマンドウインドウが変化したので、記録する。

bash 有り

`docker run` コマンドの最後に `bush` を付けると、linux ターミナルが起動する。

```
PS C:\Users\rurus> docker run -it python:3.8-slim bash
root@c2a4dca82f99:/#
```

bash 無し

`docker run` コマンドの最後に `bush` を付けないと、`python` が起動する。

```
PS C:\Users\rurus> docker run -it python:3.8-slim
Python 3.8.5 (default, Sep 10 2020, 16:58:22)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

理由

これは、`bash` を付けなかった場合、 python:3.8-slim の docker image の最後にある `CMD ["python3"]` が起動するためである。

Docker で GPU を使うには....

やはり、`nvidia docker` を使うのが一番らしい...

そこで、nvidia の Docker Hub より、以下のコマンドで、イメージをダウンロードする。

```
docker pull nvidia/cuda:11.0-base-ubuntu20.04
```

そして、次のコマンドでイメージを `build` する。

```
docker run -it nvidia/cuda:11.0-base-ubuntu20.04
```

```
PS C:\Users\rurus> docker pull nvidia/cuda:11.0-base-ubuntu20.04
11.0-base-ubuntu20.04: Pulling from nvidia/cuda
54ee1f796a1e: Pull complete
f7bfea53ad12: Pull complete
46d371e02073: Pull complete
b66c17bbf772: Pull complete
3642f1a6dfb3: Pull complete
e5ce55b8b4b9: Pull complete
155bc0332b0a: Pull complete
Digest: sha256:774ca3d612de15213102c2dbbba55df44dc5cf9870ca2be6c6e9c627fa63d67a
Status: Downloaded newer image for nvidia/cuda:11.0-base-ubuntu20.04
docker.io/nvidia/cuda:11.0-base-ubuntu20.04
PS C:\Users\rurus> docker run -it nvidia/cuda:11.0-base-ubuntu20.04
root@10f02c47c13d:/#
```

方針転換

このページ によると、

次の環境変数をセットしておけば、`TensorFlow GPU` を使用できるらしい。

よって、こちらの導入方法に移行する。

```
FROM continuumio/miniconda3

RUN conda install -y tensorflow-gpu


ENV NVIDIA_VISIBLE_DEVICES all
ENV NVIDIA_DRIVER_CAPABILITIES utility,compute
```

Dockerfile の build

Dockerfile を保存しているフォルダに移動し、以下の部分に `cmd` と入力しコマンドプロンプトを立ち上げる。

そして、以下のコマンドで `Dockerfile` を `build` する。

```
docker build . -t deeplearning
```

```
D:¥My_programing¥VirtualEnvironment¥Docker>docker build . -t deeplearning
Sending build context to Docker daemon   2.56kB
Step 1/4 : FROM continuumio/miniconda3
latest: Pulling from continuumio/miniconda3
38ced04f60ab: Pull complete
9c388eb6d33c: Pull complete
96cf53b3a9dd: Pull complete
Digest: sha256:456e3196bf3ffb13fee7c9216db4b18b5e6f4d37090b31df3e0309926e98cfe2
Status: Downloaded newer image for continuumio/miniconda3:latest
 ---> b4adc22212f1
Step 2/4 : RUN conda install -y tensorflow-gpu
 ---> Running in 2cb96b8b6db6
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying with flexible solve.
Solving environment: ...working... failed with repodata from current_repodata.json, will retry with next repodata source
```

最後に、以下のコマンドでコンテナを立ち上げる。

```
docker run -it deeplearning
```

```
D:¥My_programing¥VirtualEnvironment¥Docker>docker run -it deeplearning
(base) root@0d44784258a1:/#
```

しかし、現状 WSL2 では GPU が使用できない ので、今回はここで断念することに…。