機械学習 docker 説明書

2020/10/14

木南　貴志

流れ

1. Dockerをインストール
2. 自分のPCに搭載されているGPUの，NVIDIA-driver（基本最新でOK）をインストール
3. nvidia-container-toolkitをインストール（1.～3.は初回のみ）
4. Dockerfileを記述
5. Dockerfile ➡ image-fileを作成（ビルド）
6. image-file ➡ containerを作成（container =実際に作業する環境）

※ 環境構築が完了するまで，4.～6.を繰り返す

7. 環境構築が完了したら，container内で作業（プログラム実行等）

Dockerをインストール

UbuntuにDockerEngineをインストールする

https://docs.docker.com/engine/install/ubuntu/

※Dockerは頻繁に更新されるため，上記の公式のページを参考にしてインストールした方が無難

dockerをsudoなしでログイン

https://qiita.com/DQNEO/items/da5df074c48b012152ee

NVIDIA driver インストール

```
$ lspci | grep -i nvidia
```
自分のPCに搭載のGPUを調べる

- 調べたGPUに対応するNVIDIA driverの最新verを確認
  https://www.nvidia.co.jp/Download/index.aspx?lang=jp

- 調べたGPUに対応するNVIDIA driverの最新verのインストールファイルをダウンロード
  https://www.nvidia.co.jp/Download/index.aspx?lang=jp

```
$ chmod +x <ダウンロードしたファイル>
```
実行できるように

```
$ sudo ./<ダウンロードしたファイル>
```
ドライバインストール

```
$ reboot
```
再起動

```
$ nvidia-smi
```
ドライバがインストールされているか確認

NVIDIA driver インストール

※ nvidia-dimが使用中というエラーが出たら以降を実行

$ systemctl isolate multi-user.target	CUIモードに変更

CUIモードになったら alt+F1 → ユーザ名、パスワードを入力でログイン

$ modprobe -r nvidia-drm	無効化
$ modprobe -r nvidia-modeset	無効化
$ sudo ./<ダウンロードしたファイル>	ドライバインストール
$ reboot	再起動
$ nvidia-smi	**ドライバがインストールされているか確認**

nvidia-driver 対応CUDA確認

```
$ nvidia-smi
```

GPUの最新driverがCUDAのどのverまで対応しているか確認

CUDA 10.0以下まで対応

```
Tue Oct 22 15:46:59 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.104      Driver Version: 410.104      CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce RTX 2080    Off  | 00000000:01:00.0  On |                  N/A |
| 29%   34C    P8    24W / 225W |   5409MiB /  7949MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1440      G   /usr/lib/xorg/Xorg                           297MiB |
|    0      1652      G   /usr/bin/gnome-shell                         199MiB |
|    0      2920      G   /usr/lib/firefox/firefox                       6MiB |
```

最新ドライバーが「使用したいCUDA」に対応していない場合GPUを新しいものに買い換える必要あり

*nvidia-container-toolkit*をインストール

nvidia-containar-toolkitをインストール

https://github.com/NVIDIA/nvidia-docker

※Dockerは頻繁に更新されるため，上記の公式のページを参考にしてインストールした方が無難

機械学習の際のDockerfileの主な構成

```
FROM nvidia/cuda:9.0-cudnn7-devel-ubuntu16.04
RUN apt-get update && apt-get install -y /
    sudo /
    wget /
    vim
WORKDIR /opt
RUN wget
https://repo.anaconda.com/archive/Anaconda3-2020.02-
Linux-x86_64.sh && /
    sh Anaconda3-2020.02-Linux-x86_64.sh -b -p
/opt/anaconda3 && /
    rm -f Anaconda3-2020.02-Linux-x86_64.sh
ENV PATH /opt/anaconda3/bin:$PATH
RUN pip install --upgrade pip && pip install /
cupy-cuda90==5.3.0 /
    chainer==5.3.0
WORKDIR /
CMD ["/bin/bash"]
```

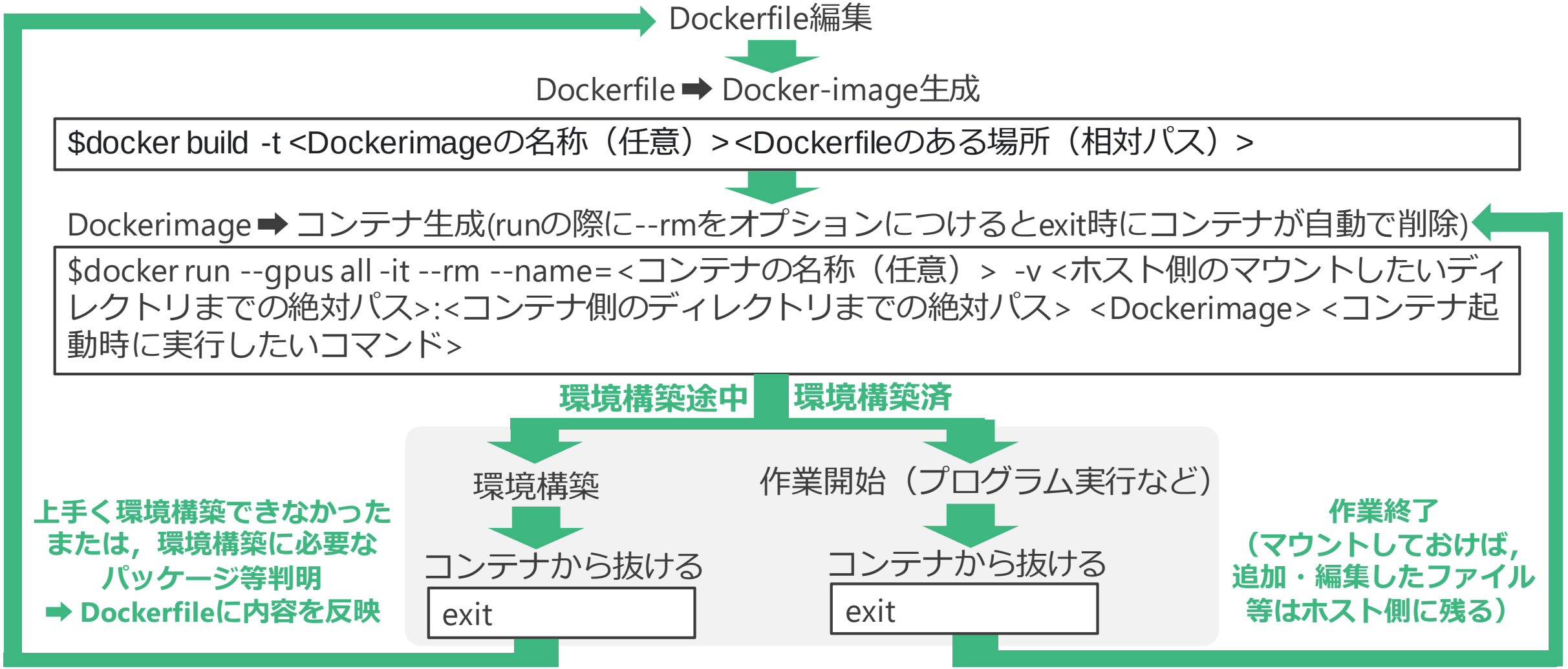
使用したいubuntu, cuda, cudnnのverを指定

anacondaのインストール
https://repo.anaconda.com/archive
で検索して任意のverのanacondaをダウンロード&
インストールが終わったらshファイルは削除
これで，python関連（numpy, matplotlib等）は
ほとんどインストールされる

使用したい機械学習のフレームワークと
verを指定してインストール

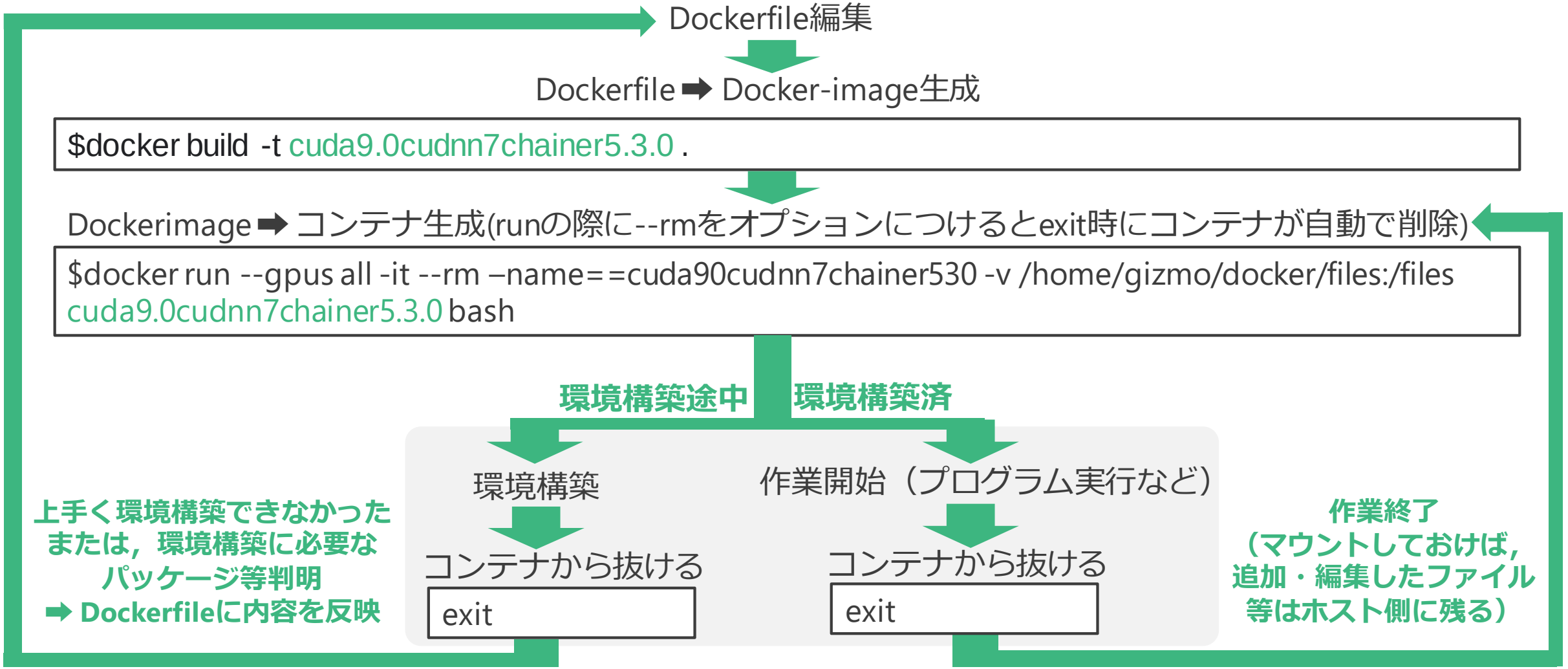
コンテナ作成の流れ　まとめ

基本的に，コンテナ作成 ➡ 破壊を繰り返すことが多い



コンテナ作成の流れ　例）

基本的に，コンテナ作成 ➡ 破壊を繰り返すことが多い



コンテナ内でのGPUの認識

```
$docker run --gpus all （以下略）
```

でコンテナを起動したときにエラーが発生した場合，NVIDIAドライバ，nvidia-containar-toolkitが正しくインストールされていない可能性あり

ホストとコンテナ内の両方で

```
$nvidia-smi
```

を実行した時，ホストとコンテナ内で内容が異なっていた場合（CUDAのバージョンにERRと表示されている等）も同様にドライバのインストールができていない可能性あり

6.dockerのコマンド

コマンド一覧

dockerで使用することの多いコマンド

(Dockerimage，containarを指定する場合，IDか名前のどちらかを指定)

$ docker build <path_to_Dockerfile>

Dockerfile ➡ Dockerimageを生成

options: -t：Dockerimageに名前をつける
-f <Dockerfilename>：Dockerfileに「Dockerfile」以外の名前をつけている場合

$ docker run <Dockerimage> <command>

Dockerimageからコンテナを生成

※ <command>を指定しない場合DockerfileのCMDで指定したコマンドを実行

exitしたらコンテナが停止する

options: -v <host>:<path>：指定したディレクトリをマウント
--name <container_name>： コンテナに名前をつける
-it： きれいに表示する（詳細は省略）
--gpus <number>：ホストのGPUをコンテナ内で使用（allでホストにある全てのGPUをコンテナで使用）
--rm：exitでコンテナから抜けた後にコンテナを削除（指定しない場合，停止はするが削除はされない）
-p <host_port>:<container_port>：ホストのポートとコンテナのポートを接続
-u <user_id><group_id>：ユーザIDとグループIDを指定（通常はrootでログインする）
※ -u $(id -u) $(id -g) と指定すれば直接IDを入力しなくて良い

コマンド一覧

dockerで使用することの多いコマンド

(Dockerimage，containarを指定する場合，IDか名前のどちらかを指定)

$ docker exec <container> <command> **コンテナに再度入る**

※ <command>を指定しない場合DockerfileのCMDで指定したコマンドを実行
options: -it： きれいに表示する（詳細は省略）
eixtした場合もコンテナが停止しない.

$ docker attach <container> <command> **コンテナに再度入る**

※ <command>を指定しない場合DockerfileのCMDで指定したコマンドを実行
options: -it： きれいに表示する（詳細は省略）
eixtしたらコンテナが停止する

$ docker restart <container> **停止したコンテナを再アクティブ化**

※ 停止している場合exec, attachができないため，停止したコンテナに入りたい場合はこれを実行してから

コマンド一覧

dockerで使用することの多いコマンド

(Dockerimage，containarを指定する場合，IDか名前のどちらかを指定)

$ docker commit <container> <new_Dockerimage> **コンテナ ➡ Dockerimageを生成**

$ docker tag <befor_Dockerimage> <after_Dockerimage> **Dockerimageの名前変更**

Dockerimageの名前は「リポジトリ名:タグ名」

$ docker push <Dockerimage> **DockerimageをDocker hubにpush**

pushする際は，Dockerimageの名前は「pushする先のリポジトリ名:タグ名」にすること

（docker hubにpushする際にDockerimageを参照して，そのリポジトリにpushするため）

コマンド一覧

dockerで使用することの多いコマンド

(Dockerimage，containarを指定する場合，IDか名前のどちらかを指定)

$ docker rm <container> **コンテナを削除**

停止していないコンテナは削除不可

$ docker stop <contaner> **コンテナを停止**

$ docker system prune **停止しているコンテナ，名前のついていないdockerimageを全て削除**

$ docker rmi <Dockerimage> **Dockerimageを削除**

コマンド一覧

コンテナ内で使用できるコマンド

$ exit	**コンテナから抜ける（抜けた後コンテナ停止）**
$ detatch	**コンテナから抜ける（抜けた後もコンテナ動作継続）**