Docker Hands-on

Agenda

- Part 1. Docker の基礎 ← **今日はここまで**
- Part 2. Dockerfile の基礎
- Part 3. Docker Compose の基礎

目的

できるようになること

- コンテナを仮想マシンに代わる手軽な開発環境として使えるようになる
- 複数のコンテナが協調して動作する環境を立ち上げられるようになる

扱わないこと

- Docker の内部の仕組み
- Docker の本番環境での利用

※一部説明を省略・簡略しているところもある

用語

- **コンテナ**
  - プロセスとその実行環境 （filesystem, network, etc.） をカプセル化したもの
- **メインプロセス**
  - コンテナの中核となる単一のプロセス
- **イメージ**
  - コンテナをアーカイブし、データとしてやりとりできるようにしたもの
- **ホスト**
  - コンテナを動かすためのマシン（各自の Mac 等）

スライドのコマンド

実際に入力してもらう部分は $ または # から始まる

```
$ ...       // ホストで実行するコマンド

# ...       // コンテナ内部で実行するコマンド
```

特に重要な部分は太字にしている

```
$ ...

# ...
```

Agenda

- Part 1. Docker の基礎
  - **コンテナの起動・停止・破棄**
  - コンテナの様々な起動オプション
  - 実行中のコンテナに対する操作
- Part 2. Dockerfile の基礎
- Part 3. Docker Compose の基礎

イメージのダウンロード

イメージを取得する方法は２つ

1. Webからダウンロードする ← まずはこちらから
2. 自分で作る ← あとで解説

以下のコマンドでUbuntuのイメージをダウンロードできる

```
$ docker image pull ubuntu
```

このあと使う他のイメージもダウンロードしておく

```
$ docker image pull nginx
$ docker image pull python
```

コンテナの起動

次のコマンドでコンテナを起動できる

`docker container run <image> <command>`

先程 pull した ubuntu イメージからコンテナを起動してみる

`$ docker container run ubuntu echo hello`

`command` で起動したプロセス（この場合は `echo hello` ）がメインプロセス

メインプロセスの生死 = コンテナの起動・停止

この場合 `echo hello` はすぐ終了するので、コンテナはすぐ停止する

起動中コンテナの一覧

今度は sleep コマンドをメインプロセスとしてコンテナを起動してみる

```
$ docker container run ubuntu sleep infinity
```

このプロセスは自動では終了しない

他のターミナルで下記を実行すると、コンテナが実行中であることがわかる

```
$ docker container list
```

```
CONTAINER ID    IMAGE     COMMAND             CREATED          STATUS          PORTS     NAMES
944883bf35c7    ubuntu    "sleep infinity"    3 seconds ago    Up 8 seconds              nervous_kowalevski
```

コンテナの停止

IDを指定して先程起動したコンテナを停止する

$ docker container stop <container ID>

今度は一覧に何も表示されなない

$ docker container list

コンテナは停止したが、まだ破棄されていないので --all をつけると見れる

$ docker container list --all

STATUS が Exited... となっているはず

コンテナの再起動

IDを指定して先程停止したコンテナを再起動する

$ docker container start <container ID>

コンテナが再起動されていることを確認

```
$ docker container list
```

※メインプロセスは最初に指定したものから変えられない

もう一度コンテナを停止しておく

```
$ docker container stop <container ID>
```

コンテナの破棄

IDを指定して先程停止したコンテナを破棄する

$ docker container rm <container ID>

破棄したので --all をつけてもコンテナはない

```
$ docker container list --all
```

再起動もできない

```
$ docker start <container ID>    # エラー
```

ここまでのおさらい

`docker image pull`	イメージをダウンロード
`docker container run`	コンテナを起動
`docker container stop|start|rm`	コンテナを停止|再起動|破棄
`docker container list`	コンテナの一覧を表示

※今回はフルコマンドを用いたが、普通はショートカットコマンドを用いる

```
docker image pull                    =>  docker pull
docker container run|stop|start|rm   =>  docker run|stop|start|rm
docker container list                =>  docker ps    // !!
```

Agenda

- Part 1. Docker の基礎
  - コンテナの起動・停止・破棄
  - **コンテナの様々な起動オプション**
  - 実行中のコンテナに対する操作
- Part 2. Dockerfile の基礎
- Part 3. Docker Compose の基礎

--rm : 停止時に即コンテナを破棄

通常はコンテナが停止しても破棄されるわけではない

```
$ docker run ubuntu echo hello

$ docker ps --all    // 停止中のコンテナが見える
```

再利用しないようなコンテナは、停止後即破棄することができる

```
$ docker run --rm ubuntu echo hello

$ docker ps --all    // コンテナは破棄されたので存在しない
```

--detach : バックグラウンドで起動

以下のように起動すると docker コマンド自体はすぐ終了する

$ docker run --detach ubuntu sleep infinity

しかしコンテナはバックグラウンドで動いている

```
$ docker ps    // バックグラウンドで実行中のコンテナが見える
```

バックグラウンド実行中のコンテナは終了するのを忘れないように

```
$ docker stop <container ID>
```

--interactive --tty：キーボード入力を有効化

以下のようにすると・・・

```
$ docker run --interactive --tty ubuntu bash
```

プロンプトが表示され、キーボード入力を受け付ける普通のシェルとして動作

```
# echo hello    // => hello
```

exit するとシェル（コンテナのメインプロセス）が終了してコンテナも停止

```
# exit
```

--interactive --tty：キーボード入力を有効化

これを応用すればPythonなどのREPLをコンテナ内で起動したりもできる

```
$ docker run --interactive --tty python python
```

それぞれのオプションの意味

- --interactive：コンテナの標準入力を開いておく
- --tty：コンテナの標準入力にターミナル（キーボード入力）を接続する

この２つのオプションを省略して -it と指定することが多い

```
$ docker run -it ubuntu bash
```

--env：環境変数を設定

コンテナ内で参照する環境変数を設定

```
$ docker run -it --env HOGE=fuga ubuntu bash

# echo $HOGE    // => fuga
```

ファイルから読み込むことも可能

```
$ docker run -it --env-file vars.env ubuntu bash
```

vars.env の例：

```
HOGE=fuga
PIYO=poyo
```

--volume : ホストとコンテナのディレクトリを同期

まずはホストに適当なディレクトリとファイルを作成

```
$ mkdir data && echo hello > data/hello.txt
```

作成したディレクトリをコンテナ内のディレクトリと同期する

```
$ docker run -it --volume $PWD/data:/tmp/data ubuntu bash
```

コンテナの中で同期したファイルを読み書きできる

```
# ls /tmp/data                          // => hello.txt が見える
# echo bye > /tmp/data/hello.txt        // => hello.txt に書き込めり
```

--publish : ホストのポートをコンテナに接続

以下を実行してブラウザで http://localhost:8080/ にアクセスすると・・・

```
$ docker run --publish 8080:80 nginx nginx -g 'daemon off;'
```

Welcome to nginx!

ターミナルに戻ると標準出力にログが出ている

```
172.17.0.1 - - [03/Apr/2018:11:23:37 +0000] "GET / ..." ...
```

MySQLやRailsも同じようにポートを接続して起動できる

デフォルトコマンドを実行

通常は docker run <image> <command> としてコマンドを実行する

```
$ docker run -it python python
```

この <command> を省略するとイメージのデフォルトコマンドが実行される

```
$ docker run -it python    // PythonのREPLが起動
```

イメージにはたいてい何らかのデフォルトコマンドが設定されている

```
$ docker run -it ubuntu                    // => bash が起動
$ docker run --publish 8080:80 nginx       // => nginx が起動
```

ここまでのおさらい

- `docker run` のオプションでコンテナの起動方法を変えられる
- `docker run <image>` でデフォルトコマンドを実行できる

主要なオプションには省略形が用意されている (`docker run --help` で確認できる)

```
--detach        =>  -d
--env           =>  -e
--interactive   =>  -i
--volume        =>  -v
--tty           =>  -t
--publish       =>  -p
```

Agenda

- Part 1. Docker の基礎
  - コンテナの起動・停止・破棄
  - コンテナの様々な起動オプション
  - **実行中のコンテナに対する操作**
- Part 2. Dockerfile の基礎
- Part 3. Docker Compose の基礎

実行中のコンテナ内で別のコマンドを実行

コンテナをバックグラウンドで起動

```
$ docker run -d ubuntu sleep infinity
```

ホストからこのコンテナの中に新しく別のコマンドを実行できる

```
$ docker container exec <container ID> ps
```

```
  PID TTY          TIME CMD
    1 ?        00:00:00 sleep
    5 ?        00:00:00 ps
```

ホストとコンテナの間でファイルをコピー

コンテナを起動

```
$ docker run -it ubuntu
```

ホストで適当にファイルを作成し、コンテナの中にコピー

```
$ echo hello > test.txt
```

$ docker container cp test.txt <container ID>:/tmp

コンテナの中でファイルを確認してみる

```
# ls /tmp    // => test.txt があるはず
```

メインプロセスの出力を取得

コンテナをバックグラウンドで起動

```
$ docker run -d -p 8080:80 nginx
```

ブラウザで http://localhost:8080/ にアクセスしてログを残す

コンテナの標準出力・標準エラー出力に記録されたログを確認

```
$ docker container logs <container ID>
```

```
172.17.0.1 - - [03/Apr/2018:11:23:37 +0000] "GET / ..." ...
```

おさらい

`docker container exec`	コンテナ内で別コマンドを実行
`docker container cp`	コンテナとホストの間でファイルをコピー
`docker container logs`	メインプロセスの出力を取得

もちろんショートカットコマンドも使える

```
docker container exec  =>  docker exec
docker container cp    =>  docker cp
docker container logs  =>  docker logs
```

Agenda

- Part 1. Docker の基礎 ← **今日はここまで**
  - コンテナの起動・停止・破棄
  - コンテナの様々な起動オプション
  - 実行中のコンテナに対する操作
- Part 2. Dockerfile の基礎
- Part 3. Docker Compose の基礎

おわりに

起動したままのコンテナがあると思うので自分で停止してください

方法はこれまでのスライドを参考に・・・

一通り停止したら下記コマンドで停止済みコンテナを一括破棄できます

```
$ docker container prune
```