第1部

第3章 コンテナの実行方法/独自イメージの作り方/複数コンテナの扱い

Dockerコンテナの操作に慣れる

山田 英伸

コンテナをソフトウェア開発で活用する方法について紹介します．想定している動作環境はWindowsです．コンテナ・フレームワークとしてDocker Desktop for Windowsがインストールされているものとします．

Dockerコンテナ実行の流れ

● 手順1：コンテナ・イメージの取得

コンテナの基本的な操作を解説します．ここでは，Pythonプログラムを実行できるコンテナ・イメージを使います．これはDocker Hubから取得します．

Docker Hubはコンテナ・レジストリとも呼ばれ，さまざまなコンテナ・イメージが登録されています．Windowsのコマンド・プロンプトまたはWindows Terminalなどで次のコマンドを実行することで，コンテナ・イメージを取得できます．

```
$ docker image pull python:3.12
```

リスト1　Pythonの実行環境を持つコンテナをコマンドラインから起動する

```
$ docker container run -v .:/workspace -it
                                    python:3.12 bash
root@cb172c661b22:/#
```

これ以降はコンテナ内のシェル操作になる

実行するとコンテナ内のシェルに切り替わり，ホスト名やユーザ名の表示が変化する

表1　コンテナ起動時の代表的なコマンドライン・オプション

オプション	意　味
-v，--volume	マウント（バインド・マウント，ボリューム・マウント）の指示をする
-p，--publish	コンテナのポートをホストに公開する
-e，--env	環境変数を設定する
-d，--detach	コンテナをバックグラウンド実行する
--rm	コンテナ終了時に自動的に削除
-t，--tty	疑似TTYを割り当てる
-i，--interactive	アタッチしていなくても，標準入力を開き続ける
--name	コンテナに名前を割り当てる
-w，--workdir	コンテナ内の作業ディレクトリ

● 手順2：コンテナ起動

Pythonのコンテナを起動し，その中でbashシェルを操作するためのコマンドを**リスト1**に示します．

実行するとコンテナが起動し，直接コンテナ内に入ることができます．-vオプションを付けると，現在のフォルダがコンテナ内の/workspaceディレクトリにバインド・マウントされます．これにより，ホストOS上のファイルをコンテナ内で読み書きできるようになります．

また，コンテナ内で対話的なシェル操作を行うために，-itオプションを設定しています．**表1**にコンテナ起動時に指定できる主なオプションを示します．

コンテナ内で生成したファイルは，コンテナ本体が削除されると消えてしまいます．残しておきたいデータについては，次のいずれかの方法で保存（データの永続化）します．

▶方法1：バインド・マウント

ホスト・マシン上のファイルやフォルダを対象として，コンテナ内にマウントします．

▶方法2：ボリューム・マウント

Dockerが管理している場所にボリュームという単位で領域を用意し，それをコンテナ内にマウントします．ファイルのI/O速度やファイルの権限周りの観点からは，ボリューム・マウントの方が優れています．

ここでは，Windowsのメモ帳などでPythonのプログラムを用意するのが簡単なことから，バインド・マウントを選択します．これはコンテナを起動させる際に，コマンド・オプションで指定します．

● 手順3：コンテナでPythonプログラムを実行

一度，コンテナのシェルを離れて，Windows環境でPythonプログラムを作成します．任意のテキスト・エディタを使って，先ほどDockerコマンドを実行したカレント・フォルダにmain.pyというファイル名で保存します（**リスト2**）．

▶方法1：コンテナ内のシェルからプログラムを起動

コンテナ内のシェルに戻って作業します．コンテナ内の/workspaceディレクトリでは，Windows側