Docker を用いた **Elixir/Phoenix** 開発環境の構築手順

黒田努　著

2020-02-02 版　　発行

目次

第 1 章

Docker Desktop for Mac を利用した環境構築の手順

この章では、は Docker Desktop for Mac を利用して Elixir/Phoenix 開発環境を整える手順を解説します。

1.1 Git のインストール

本章の手順に沿って Elixir/Phoenix の開発環境を構築するためには、あなたの Mac に **Git** がインストールされている必要があります。

ターミナルを開き、次のコマンドを実行してください。

```
% git --version
```

ここで、次のように表示されれば Git がインストール済みです。次の節に進んでください。

```
git version 2.7.4
```

ここで「git: command not found」あるいは「git: コマンドが見つかりません」と表示された場合は未インストールです。

ブラウザで次のページを開いてインストーラをダウンロードしてインストールしてください。

```
https://git-scm.com/downloads
```

Git をインストールする詳細な手順は省略します。

インストールが完了したら、ターミナルで git --version コマンドを実行してバージョン番号を確かめてください。

1.2 Dockerfile 等の取得

ターミナルで次のコマンドを実行してください。

```
% git clone https://github.com/oiax/phx-compose.git
% cd phx-compose
```

以後、この phx-compose ディレクトリを**作業ディレクトリ**と呼びます。

1.3 Docker と Docker Compose の状態を確認

本節では、**Docker**（ドッカー）と **Docker Compose**（ドッカー・コンポーズ）を用いて Linux の仮想環境を作成し、その上に Elixir/Phoenix の開発環境を構築します。

まず、あなたの Mac に Docker がインストールされているかどうか確認します。ターミナルで、次のコマンドを実行してください。

```
% docker --version
```

ここで、次のように表示されれば Docker がインストール済みです。

```
Docker version 19.03.5, build 633a0ea838
```

ここで「docker: command not found」あるいは「docker: コマンドが見つかりません」と表示された場合は未インストールです。また、Docker のバージョンが 18 未満の場合は、再インストールが必要です。

Docker バージョン 18 以上がインストールされている場合は、さらに Docker Compose の状態を確認します。ターミナルで、次のコマンドを実行してください。

```
% docker-compose --version
```

ここで、次のように表示されれば OK です。

```
docker-compose version 1.24.0, build 0aa59064
```

Docker Desktop for Mac には Docker Compose が同梱されていますので、Docker だけがインストールされている状況は通常ありません。

Docker と Docker Compose の最新バージョンがインストールされている場合、次の項をスキップしてください。

1.4 Docker Desktop for Mac のインストール

はじめて Docker を使う方は、まずブラウザで `https://hub.docker.com/signup` を開き、Docker Hub のアカウントを取得してください。そして、ブラウザで次のアドレスを開きます。

```
https://hub.docker.com/editions/community/
docker-ce-desktop-mac
```

そして、画面右側にある「Get Docker」ボタンをクリックして、インストーラをダウンロードします。

ダウンロードされたファイル `Docker.dmg` をダブルクリックすると、インストーラが開きます。Docker の鯨アイコンをドラッグして Application フォルダにドロップすればインストール完了です。

Finder の「アプリケーション」フォルダにある Docker アイコンをダブルクリックすると Docker Desktop for Mac が起動します。初回起動時は Mac のパスワード入力を求められます。デスクトップ最上部のステータスバーにある鯨のアイコンで Docker の状態が表示されます。起動直後は鯨のアイコンが動き続けます。Docker の準備が整うとアイコンが静止します。

ターミナルを開いてバージョン番号を確認します。

```
% docker --version
Docker version 19.03.5, build 633a0ea838
```

■コラム: Docker に割り当てるメモリの量の調整

初期状態で Docker に割り当てられているメモリの量は 2GB です。これでも Elixir/Phoenix の開発は可能ですが、やや足りないかもしれません。あなたの PC に 8GB 以上のメモリが搭載されているのであれば、4GB 以上のメモリを割り当てると、より快適に本書の学習を進められるでしょう。

ステータスバーの鯨アイコンをクリックして、ドロップダウンメニューから「Preferences」を選んでください。そして、「Advanced」タブを開いて「Memory」のスライドバーを調整してください。「Apply & Restart」ボタンをクリックすると、Docker が再起動して新しい設定が有効になります。

1.5 Elixir/Phoenix コンテナのビルド

ターミナルを開き、作業ディレクトリ（phx-compose ディレクトリ）において次のコマンドを実行してください。

```
% bin/setup.sh
```

本書で使用する Elixir/Phoenix コンテナのビルドが始まります。初回はかなり時間がかかりますので、気長にお待ちください。

第 2 章

VirtualBox を利用した Windows への環境構築手順

この章では、は VirtualBox を利用して Windows に Elixir/Phoenix 開発環境を整える手順を解説します。

2.1 VirtualBox を採用する理由

Docker Desktop for Windows というソフトウェアを利用すれば、Windows 上に直接 Docker をインストールできますが、いくつかのやや厳しい条件を満たす必要があります（コラムを参照）。また、本書の執筆時点（2020 年 2 月）では、Docker for Windows はまだ十分に安定していないというのが筆者の印象です。

そこで、本書では VirtualBox を利用して間接的に Docker をインストールする方法を推奨します。

> VirtualBox（バーチャルボックス）は正式名称を Oracle VM VirtualBox といい、仮想環境を作り出すソフトウェアのひとつです。

具体的には、Windows に VirtualBox をインストールし、その上に Linux 系 OS である Ubuntu 18.04 Desktop をインストールし、さらにその上に Docker をインストールします。複雑に思えるかもしれませんが、やってみればそんなに難しくはありません。

■コラム: Docker Desktop for Windows

Docker Desktop for Windows を利用するには、以下の条件をすべて満たす必要があります。特に、Windows 10 Home が対象外である点に注意してください。

- 64 ビット版の Windows 10 Pro または Enterprise
- Windows 10 May 2019 Update (1903) が適用済みであること
- BIOS 設定で BIOS レベルのハードウェア仮想化が有効化されていること
- 4GB 以上の RAM を搭載していること

「BIOS レベルのハードウェア仮想化」の状態を調べる方法は、PC によって異なります。PC の型番・機種名と「BIOS 仮想化」というキーワードを組み合わせてインターネット検索を行うと解説記事が見つかるでしょう。

2.2 VirtualBox と Ubuntu をインストールする上での注意事項

VirtualBox および Ubuntu 18.04 のインストール手順は省略します。「virtualbox ubuntu windows 10」などのキーワードでインターネット検索を行うと多数のブログ記事などにヒットしますので、なるべく日付の新しいものを選んで参考にしてください。なお、インストール完了までにはかなり長い時間（2 時間以上）を要します。

Ubuntu 18.04 をインストールする際に注意すべき事項を 3 点挙げます。

1. 本書の執筆時点（2020 年 2 月）における Ubuntu の最新バージョンは 19.10 ですが、バージョン 18.04 を選択してください。
2. 仮想ディスクに確保するサイズは 16GB 以上にしてください（デフォルトは 10GB）。
3. 可能であれば 4GB 以上のメモリを割り当ててください。

Ubuntu のインストールが完了したら、次章に進んで Docker と Docker Compose をインストールしてください。

第 3 章

Docker を利用した Ubuntu への環境構築手順

この章では、Docker と Docker Compose を用いて Ubuntu 18.04 に Elixir/Phoenix 開発環境を整える手順を解説します。

3.1 Git のインストール

ホスト PC（Ubuntu）に **Git** がインストールされているかどうか調べるため、ターミナルで次のコマンドを実行してください。

```
% git --version
```

ここで、次のように表示されれば Docker がインストール済みです。

```
git version 2.7.4
```

ここで「git: command not found」あるいは「git: コマンドが見つかりません」と表示された場合は未インストールです。ターミナルで以下のコマンドを順に実行してください。

```
% sudo add-apt-repository ppa:git-core/ppa
% sudo apt-get update
```

```
% sudo apt-get install git
```

3.2 Dockerfile 等の取得

ターミナルで次のコマンドを実行してください。

```
% git clone https://github.com/oiax/phx-compose.git
% cd phx-compose
```

以後、この phx-compose ディレクトリを**作業ディレクトリ**と呼びます。

3.3 Docker と Docker Compose の状態を確認

本書では、**Docker**（ドッカー）と **Docker Compose**（ドッカー・コンポーズ）を用いて Linux の仮想環境を作成し、その上に Elixir/Phoenix の開発環境を構築します。

まず、ホスト PC（Ubuntu）に Docker がインストールされているかどうか確認します。ターミナルで、次のコマンドを実行してください。

```
% docker --version
```

ここで、次のように表示されれば Docker がインストール済みです。

```
Docker version 19.03.5, build 633a0ea838
```

ここで「docker: command not found」あるいは「docker: コマンドが見つかりません」と表示された場合は未インストールです。また、Docker のバージョンが 18 未満の場合は、再インストールが必要です。

Docker バージョン 18 以上がインストールされている場合は、さらに Docker Compose の状態を確認します。ターミナルで、次のコマンドを実行してください。

```
% docker-compose --version
```

ここで、次のように表示されれば OK です。

```
docker-compose version 1.24.0, build 0aa59064
```

Docker と Docker Compose の最新バージョンがインストールされている場合、次の節をスキップしてください。

3.4 Docker と Docker Compose のインストール

ターミナルを開き、作業ディレクトリ（phx-compose ディレクトリ）で次のコマンドを実行してください。

```
% bin/install_docker_on_unbuntu.sh
```

そして、バージョン番号を確認します。

```
% docker --version
Docker version 19.03.5, build 633a0ea838
% docker-compose --version
```

続いて、次のコマンドを実行してください。

```
% cat /proc/sys/fs/inotify/max_user_watches
```

すると、ターミナルに「8192」のような数字が出力されます。

この値が十分に大きくないと Phoenix の開発に支障が出ます。通常は、システムの上限値である「524288」をセットします。以下のコマンド群を順に実行してください。

```
% echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf
% sudo sysctl -p
```

Docker コンテナの中で max_user_watches の値を変更することはできません。上記のコマンド群は必ずホスト PC 側で実行してください。

3.5 Elixir/Phoenix コンテナのビルド

ターミナルを開き、作業ディレクトリ（phx-compose ディレクトリ）において次のコマンドを実行してください。

```
% bin/setup.sh
```

本書で使用する Elixir/Phoenix コンテナのビルドが始まります。初回はかなり時間がかかりますので、気長にお待ちください。

第 4 章

web コンテナ

Docker で構築された個々の仮想環境を**コンテナ**と呼びます。この章では Elixir/Phoenix の学習を進めるためのコンテナを作成し、利用する方法を説明します。

4.1 コンテナ群の起動

ホスト PC のターミナル上で次のスクリプトを実行すると db コンテナと web コンテナが起動します。

```
% docker-compose up -d
```

ターミナルに次のように出力されれば、無事に起動されています。

```
Creating network "phx-compose_default" with the default driver
Creating phx-compose_db_1 ... done
Creating phx-compose_web_1 ... done
```

コンテナ群の状態を調べるには、次のコマンドを実行してください。

```
% docker-compose ps
```

次のように出力されれば、2 つのコンテナが正常に起動されていることがわかります。

```
      Name                     Command               State          Ports
-------------------------------------------------------------------------------------
phx-compose_db_1    docker-entrypoint.sh postgres   Up      5432/tcp
phx-compose_web_1   bash                            Up      0.0.0.0:4000->4000/tcp
```

4.2 web コンテナへのログイン

次のスクリプトを実行すると web コンテナ上で *bash* が起動します。

```
% docker-compose exec web bash
```

以後、上記の操作を「web コンテナへログインする」と表現します。

4.3 ホスト PC と web コンテナの間で共有されるディレクトリ

web コンテナにログインしてから、次のコマンドを実行してください。

```
$ pwd
```

ターミナルに /projects と出力されます。

> ホスト PC で作業しているときコマンドプロンプトは % で表されます。web コンテナで作業しているときコマンドプロンプトは $ で表されます。

このディレクトリとホスト PC 上の projects ディレクトリは共有されています。試しに、web コンテナ上でファイルを作ってみましょう。

```
$ echo HELLO > test.txt
```

ホスト PC 上で projects ディレクトリを見ると、「HELLO」という中身を持つファイル test.txt ができています。適当なテキストエディタでこのファイルを開き、「HELLO」の後に感嘆符（!）を加えて保存してください。そして、web

コンテナ上でファイルの中身を確認します。

```
$ cat test.txt
```

ターミナルに HELLO! と出力されます。

さらに、ホスト PC 上でこのファイルを削除してください。すると web コンテナ上でも対応するファイルが削除されます。

4.4 各種ソフトウェアのバージョン番号の確認

Elixir のバージョン番号の確認

web コンテナのターミナルで次のコマンドを実行してください。

```
$ mix hex.info
```

次のような結果が出力されます。

```
Hex:    0.20.1
Elixir: 1.10.0
OTP:    22.2.4

Built with: Elixir 1.8.2 and OTP 20.3
```

3 行目に表示されている数字が Erlang/OTP のバージョン番号です。Elixir のバージョン番号は 2 行目に出力されています。

Phoenix のバージョン番号の確認

web コンテナのターミナルで次のコマンドを実行してください。

```
$ mix phx.new -v
```

次のような結果が出力されれば OK です。

```
Phoenix v1.4.12
```

■コラム: Phoenix のバージョンアップ

最新の Phoenix のバージョン番号を調べるには、ブラウザで次の URL を開いてください。

```
https://hex.pm/packages/phoenix
```

図 4.1 のようにバージョン番号のリストが表示されます。

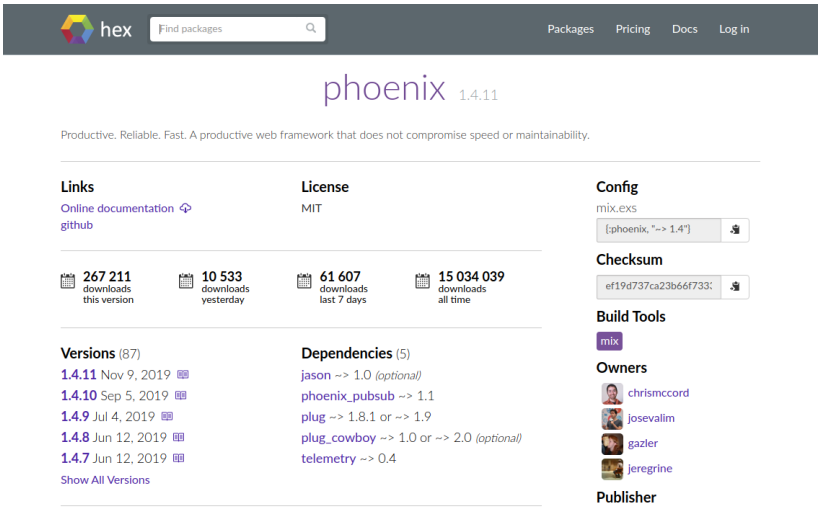

図 4.1　Hex.pm の phoenix ページ

ここに 1.4.13 というバージョン番号が現れている場合、`Dockerfile` の最終行にある `1.4.12` を `1.4.13` に書き換えてから、ホスト側のターミナルで次のコマンドを実行すれば、Phoenix のバージョンが上がります。

ただし、本書の記述は Phoenix 1.4 に基づいていますので、本書での学習中はバージョンを 1.5 以上に上げないでください。

Node.js のバージョン番号の確認 3

`web` コンテナで次のコマンドを実行してください。

```
$ node --version
```

次のような結果が出力されれば OK です。

```
v10.18.1
```

npm のバージョン番号の確認 4

web コンテナで次のコマンドを実行してください。

```
$ npm --version
```

次のような結果が出力されれば OK です。

```
6.13.4
```

4.5 web コンテナからのログアウト

web コンテナで次のコマンドを実行すると、web コンテナからログアウトします。

```
$ exit
```

4.6 コンテナ群の停止

ホスト PC のターミナル上で次のコマンドを実行すると db コンテナと web コンテナが停止します。

```
% docker-compose stop
```

ターミナルに次のように出力されれば、無事に停止しています。

```
Stopping phx-compose_web_1 ... done
Stopping phx-compose_db_1 ... done
```

コンテナ群の状態を調べるため、次のコマンドを実行してください。

```
% docker-compose ps
```

次のように出力されれば、db コンテナと web コンテナは停止中です。

```
      Name                     Command               State    Ports
-------------------------------------------------------------------
phx-compose_db_1    docker-entrypoint.sh postgres   Exit 0
phx-compose_web_1   bash                            Exit 0
```

第 5 章

db コンテナの利用

この章では PostgreSQL サーバーが稼働するコンテナにログインしてデータベースを操作する方法を説明します。

5.1 db コンテナの起動

ホスト PC のターミナル上で次のスクリプトを実行すると db コンテナが起動します。

```
% docker-compose up -d db
```

ターミナルに次のように出力されれば、無事に起動されています。

```
Starting phx-compose_db_1 ... done
```

コンテナ群の状態を調べるため、次のコマンドを実行してください。

```
% docker-compose ps
```

次のように出力されれば、db コンテナだけが正常に起動されていることがわかります。

```
      Name                     Command               State     Ports
-------------------------------------------------------------------------
phx-compose_db_1    docker-entrypoint.sh postgres   Up       5432/tcp
```

```
phx-compose_web_1   bash                          Exit 0
```

5.2 db コンテナへのログイン

次のスクリプトを実行すると db コンテナ上で *bash* が起動します。

```
% docker-compose exec db bash
```

以後、上記の操作を「db コンテナへログインする」と表現します。

5.3 psql コンソール

db コンテナにログインしてから、次のコマンドを実行してください。

```
$ psql -U postgres
```

すると、ターミナルに次のように表示され、対話的に PostgreSQL サーバーを操作する「psql コンソール」が開始します。

```
psql (12.1)
Type "help" for help.

postgres=#
```

psql コンソールに対して \l コマンドを入力すると、データベースのリストが表示されます（図 5.1）。

```
                                 List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------+----------+----------+------------+------------+-----------------------
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
(3 rows)
```

図 5.1　データベースのリスト

\q コマンドを入力すると、psql コンソールが終了します。

5.4 db コンテナからのログアウト

db コンテナで次のコマンドを実行すると、db コンテナからログアウトします。

```
$ exit
```

5.5 db コンテナの停止

ホスト PC のターミナル上で次のコマンドを実行すると db コンテナが停止します。

```
% docker-compose stop
```

ターミナルに次のように出力されれば、無事に停止しています。

```
Stopping phx-compose_db_1 ... done
```

コンテナ群の状態を調べるため、次のコマンドを実行してください。

```
% docker-compose ps
```

次のように出力されれば、すべてのコンテナが停止中です。

```
      Name                     Command              State    Ports
-------------------------------------------------------------------
phx-compose_db_1    docker-entrypoint.sh postgres   Exit 0
phx-compose_web_1   bash                            Exit 0
```

Docker を用いた **Elixir/Phoenix** 開発環境の構築手順

2020 年 2 月 2 日　初版第 1 刷　発行
著　者　黒田努
発行所　株式会社コアジェニック

(C) 2020 Tsutomu Kuroda