付録 A

各種ソフトウェアのインストール

Elixir/Phoenix による Web アプリケーション開発の環境を整えましょう。まず、Docker と Docker Compose をインストールします。OS により手順が異なる場合は、各節の前半で macOS の場合について、後半で Ubuntu の場合について解説します。その後、Docker Compose を用いて Elxir/Phoenix コンテナを作成します。こちらの手順は両 OS で共通です。

A.1 Docker のインストール

準備作業

あなたの PC 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 Git がインストール済みです。

```
git version 2.25.1
```

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。

macOS の場合は、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、通常の手順でインストールしてください。

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

Ubuntu の場合は、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
```

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 -b ex02 https://github.com/oiax/phx-compose.git ex-v02
% cd ex-v02
```

この ex-v02 ディレクトリが本書における作業ディレクトリとなります。

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、あなたの PC 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 19.03.13, build 4484c46d9d
```

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 1.26.2, build eefe0d31
```

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 の新しいバージョンがインストールされている場合、次の節に進んでください。

macOS へのインストール

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 19.03.13, build 4484c46d9d
```

■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。

Ubuntu へのインストール

Ubuntu へのインストール手順はやや複雑であるため、筆者がインストールスクリプトを用意しました。この節の前半で *git* コマンドにより `Dockerfile` などを含むリポジトリを取得しました。その際にできた `ex-v02` 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 19.03.13, build 4484c46d9d
% docker-compose --version
docker-compose version 1.26.2, build eefe0d31
```

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。なお、Docker for Mac で作った Docker コンテナでは max_user_watches の値が「524288」にセットされています。上記のコマンド群を実行する必要があるのは、ホスト PC が Linux 系 OS である場合だけです。

A.2 Elixir/Phoenix コンテナの作成と利用

Docker で構築された個々の仮想環境を**コンテナ**と呼びます。この節では 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。

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（ex-v02 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。

以後、Elixir/Phoenix コンテナのことを web コンテナと呼びます。

web コンテナの起動

次のスクリプトを実行すると web コンテナ、およびデータベースサーバである postgres コンテナ、mysql コンテナが起動します。

```
% bin/start.sh
```

> このスクリプトは docker-compose up -d コマンドを実行します。

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_postgres_1  ... done
Starting phx-compose_mysql_1  ... done
Starting phx-compose_web_1  ... done
```

web コンテナ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、正常に起動されていることがわかります。

```
         Name                       Command               State           Ports
-------------------------------------------------------------------------------------
phx-compose_mysql_1      docker-entrypoint.sh --def ...   Up      3306/tcp, 33060/tcp
phx-compose_postgres_1   docker-entrypoint.sh postgres    Up      5432/tcp
phx-compose_web_1        bash                             Up      0.0.0.0:4000->4000/tcp
```

次のように出力された場合は、停止中です。

```
      Name          Command   State          Ports
-------------------------------------------------------------
```

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。

```
% bin/login.sh
```

> このスクリプトは docker-compose exec web bash コマンドを実行します。

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。

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 /apps 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 ex-v02/apps 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 ex-v01/apps 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。

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:    0.20.5
Elixir: 1.10.4
OTP:    22.3.4.10

Built with: Elixir 1.10.0 and OTP 21.3
```

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 v1.5.5
```

Node.js のバージョン番号の確認

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。

```
v12.18.4
```

npm のバージョン番号の確認

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。

```
6.14.6
```

pg_dump のバージョン番号の確認

web コンテナで次のコマンドを実行してください。

```
$ pg_dump --version
```

次のような結果が出力されれば OK です。

```
pg_dump (PostgreSQL) 12.4 (Debian 12.4-1.pgdg100+1)
```

mysqldump のバージョン番号の確認

web コンテナで次のコマンドを実行してください。

```
$ mysqldump --version
```

次のような結果が出力されれば OK です。

```
mysqldump  Ver 10.17 Distrib 10.3.23-MariaDB, for debian-linux-gnu (x86_64)
```

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
```

あるいは Ctrl-D でもログアウトできます。

web コンテナの停止

web コンテナからログアウトした後、ホスト PC 上で次のコマンドを実行すると、web コンテナが停止します。

```
% bin/stop.sh
```

このスクリプトは docker-compose down コマンドを実行します。