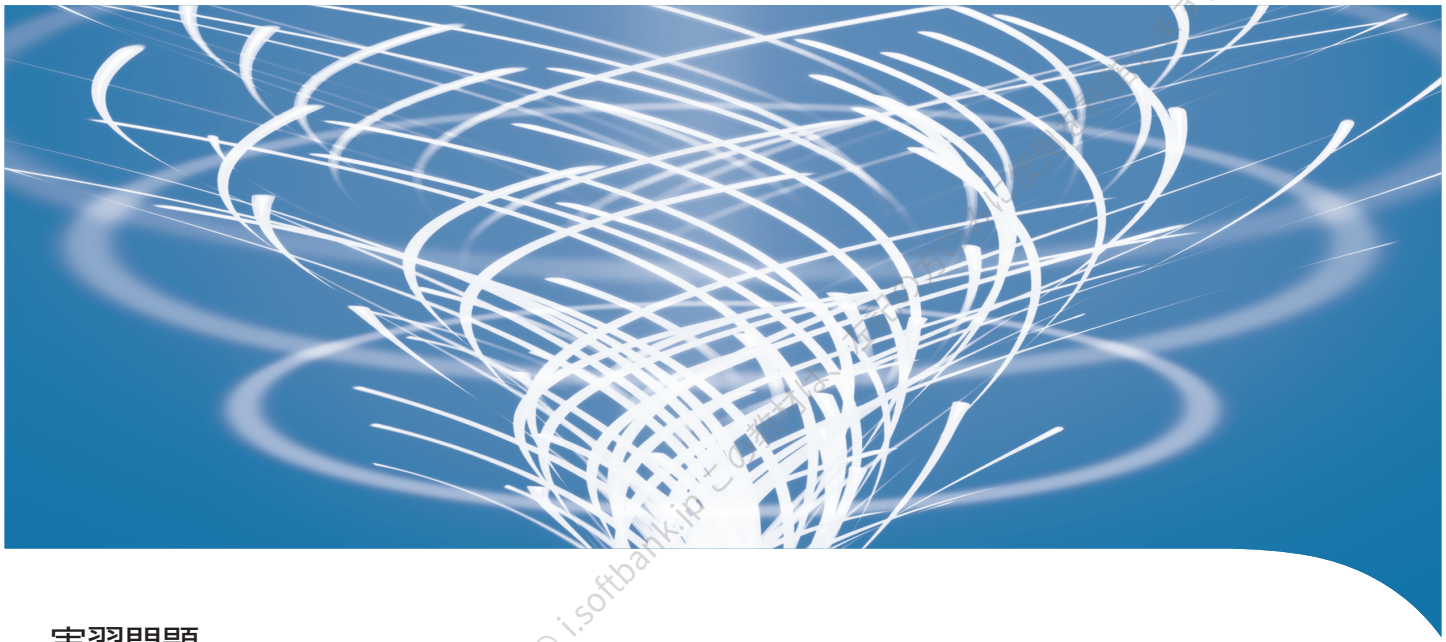
FUJITSU

FUJITSU 人材育成・研修サービス

アプリ開発者のための、Dockerで学ぶコンテナ仮想化入門



実習問題

UBS83L1E-02

shaping tomorrow with you

社会とお客様の豊かな未来のために

はじめに

本書は「アプリ開発者のための、Dockerで学ぶコンテナ仮想化入門」コースのために作成された実習手引きです。

- 本書は、2016 年 5 月の資料に基づいて作成されています。
- 本書は、講習会のレベルアップのために予告なしに変更することがあります。

2016 年 6 月 初版
2016 年 7 月 2 版

〔お願い〕
- 本書を無断で他に転載しないようにお願いします。
- 製品適用に際しては、各ソフトウェアの機能詳細を確認してください。
- 本書についてのご意見、ご要望、ご質問がございましたら、お手数ですが、以下のフォームの「その他のお問い合わせ」よりお送りください。
  https://www.knowledgewing.com/contact/FLM/default/

〒108-0075 東京都港区港南 2-13-34（NSS-Ⅱビル）
株式会社富士通ラーニングメディア
http://jp.fujitsu.com/flm

登録商標/商標

●本書に記載されている会社名、製品名は各社の商標および登録商標です。

目 次

Lab 1-1　Ubuntu イメージからのコンテナの作成 1
Lab 1-2　MySQL コンテナの起動 3

Lab 3-1　Apache コンテナの作成と停止 5
Lab 3-2　Ubuntu コンテナの起動とコンテナへの接続 7
Lab 3-3　Web3 階層システムの作成 8

Lab 4-1　コンテナからのイメージの作成 12
Lab 4-2　Docker Hub へのイメージのアップロード 13
Lab 4-3　ローカルイメージの書き出しとリストア 14
Lab 4-4　コンテナの書き出しとリストア 15
Lab 4-5　Dockerfile を使用したイメージ作成 17

Lab 5-1　Docker Toolbox の Windows へのインストール 19
Lab 5-2　Docker Toolbox の操作 21

Lab 1-1

Ubuntu イメージからのコンテナの作成

- ダウンロード済みの公式 Ubuntu イメージからコンテナを作成します
- コマンドの詳細は第 3 章で説明します

1. 実習環境の確認と接続

(1) SSH で接続する、実習環境用の Linux ホストの IP アドレスを記入してください

IP アドレス

(2) Tera Term を使用し、SSH で Linux ホストに接続してください。ユーザー名は root 、パスワードは P@ssword です。

2. Ubuntu コンテナの作成と起動

(1) Linux ホストのディストリビューションを確認します。CentOS 7 を使用しています。

```
# cat /etc/redhat-release
```

(2) ダウンロード済みの公式 Ubuntu イメージから、ubuntu-svr という名前のコンテナを作成し、起動させます。コンテナ内では/bin/bash を実行させます。コンテナの起動後、プロンプトの表示が変化し、コンテナ内で bash を使用して操作ができるようになります。

```
# docker run -it --name ubuntu-svr ubuntu /bin/bash
```

※操作に失敗したら、「docker rm コンテナ名」を実行してコンテナを削除すると、(2) をやり直すことができます。

(3) コンテナのディストリビューションが Ubuntu であることを確認します。

```
# cat /etc/os-relaese
```

(4) コンテナ内で起動しているプロセスが bash のみであることを確認します。

```
# ps aux
```

(5) コンテナからログアウトします。Linux ホストでの操作に戻ります。

```
# exit
```

(6) コンテナ内で最初に起動したプロセスが終了すると、コンテナ自体が停止します。exit コマンドで bash が停止したことで、ubuntu-svr コンテナも停止したことを確認します。

```
# docker ps -a
```

「STATUS」が「Exited」と表示されていれば、コンテナは停止しています。

※ (2)はコンテナを作成するコマンドですので、1 回しか実行できません。(2)のコマンドを 2 回実行しようとすると、同じ名前のコンテナがすでに存在するため、エラーとなります。

Lab 1-2

MySQL コンテナの起動

- 作成済みの MySQL コンテナを起動させます
- バージョンの異なる MySQL のコンテナを同時に起動できることを確認します

1. バージョン 5.6 の MySQL コンテナの起動

(1) mysql-5.6 という名前のコンテナを起動します。

```
# docker start mysql-5.6
```

(2) コンテナ内で bash を起動させ、接続します。プロンプトの表示が変化し、コンテナ内で操作が可能になります。

```
# docker exec -it mysql-5.6 /bin/bash
```

(3) MySQL のプロセスが起動していることを確認します。

```
# ps aux
```

(4) mysql コマンドでデータベースに接続します。

```
# mysql --user=root --password=password
```

(5) MySQL のバージョンを確認します。

```
mysql> SELECT version();
```

(6) mysql コマンドを終了します。

```
mysql> quit
```

(7) bash を終了し、Linux ホストでの操作に戻ります。MySQL のプロセスは停止させていませんので、コンテナは起動したままです。

```
# exit
```

2. バージョン 5.7 の MySQL コンテナの起動

(1) mysql-5.7 という名前のコンテナを起動します。

```
# docker start mysql-5.7
```

(2) コンテナ内で bash を起動させ、接続します。プロンプトの表示が変化し、コンテナ内で操作が可能になります。

```
# docker exec -it mysql-5.7 /bin/bash
```

(3) MySQL のプロセスが起動していることを確認します。

```
# ps aux
```

(4) mysql コマンドでデータベースに接続します。

```
# mysql --user=root --password=password
```

(5) MySQL のバージョンを確認します。

```
mysql> SELECT version();
```

(6) mysql コマンドを終了します。

```
mysql> quit
```

(7) bash を終了し、Linux ホストでの操作に戻ります。MySQL のプロセスは停止させていませんので、コンテナは起動したままです。

```
# exit
```

(8) Linux ホストでプロセスを確認します。mysqld が 2 つ起動しています。

```
# ps aux | grep mysqld
```

(9) 2 つの MySQL コンテナを停止させます。

```
# docker stop mysql-5.6
# docker stop mysql-5.7
```

(10) Linux ホストでプロセスを確認します。mysqld は動作していません。

```
# ps aux | grep mysqld
```

Lab 3-1

Apache コンテナの作成と停止

- Apache の公式イメージからコンテナを作成します
- 外部からコンテナへネットワーク経由で接続します

1. Apache イメージからのコンテナの作成と停止

(1) Apache の公式イメージからコンテナを作成して起動します。イメージはローカルに保存されていませんので、Docker Hub から自動的にダウンロードされます。
外部からアクセスができるよう、NAPT の設定を行います。

```
# docker run -d --name web01 -p 8080:80 httpd:latest
```

(2) 動作中のコンテナの一覧を表示し、web01 が起動していることを確認します。

```
# docker ps
```

(3) httpd のプロセスが動作していることを、Linux ホスト上で ps コマンドを実行して確認します。

```
# ps aux | grep httpd
```

(4) httpd のプロセスが動作していることを、コンテナ内で ps コマンドを実行して確認します。
Linux ホスト上で ps コマンドを実行した場合とは、表示されるプロセス ID が異なります。

```
# docker exec web01 ps aux | grep httpd
```

(5) Windows から Web ブラウザでアクセスします。以下の URL を指定してください。
http://<Linux ホストの IP アドレス>:8080
「It works!」と表示されれば、正常に動作しています。

2. Apache コンテナの停止と削除

(1) コンテナを停止します。

```
# docker stop web01
```

(2) コンテナの一覧を表示し、web01 が停止したことを確認します。

```
# docker ps

# docker ps -a
```

(3) Linux ホスト上で ps コマンドを実行し、httpd プロセスが動作していないことを確認します。

```
# ps aux | grep httpd
```

(4) web01 を削除します。

```
# docker rm web01
```

Lab 3-2

Ubuntu コンテナの起動とコンテナへの接続

1. 停止中のコンテナの起動と接続

（1）Lab 1-1 で作成した ubuntu-svr コンテナを起動させます。

```
# docker start ubuntu-svr
```

（2）ubuntu-svr で動作している bash を使用して接続します。

```
# docker attach ubuntu-svr
```

コマンドの実行後、プロンプトが表示されない場合は、Enter キーを押してください。コンテナ内で操作ができるようになります。

（3）[Ctrl] + p 、[Ctrl] + q を押すことでコンテナとの接続が切断できます。接続を切断しただけですので、コンテナは実行中であり、docker attach コマンドで再度接続ができます。

Lab 3-3

Web3 階層システムの作成

- Apache、Tomcat、PostgreSQL のコンテナを起動します
- --link オプションでコンテナ間のネットワークを設定します
- コンテナへのアプリケーションの配備や設定変更します

1. PostgreSQL コンテナの起動

（1）PostgreSQL の公式イメージから dbsvr という名前のコンテナを作成して起動させます。

```
# docker run -d --name dbsvr postgres:latest
```

（2）テーブルを作成するためのスクリプトファイルをコンテナのルートディレクトリにコピーします。

```
# docker cp $HOME/freemarknans/sql/createtable.sql dbsvr:/
```

（3）dbsvr コンテナで bash を起動させ、接続します。

```
# docker exec -it dbsvr /bin/bash
```

（4）コンテナ上で createdb コマンドを実行し、testdb という名前のデータベースを作成します。

```
# createdb -U postgres testdb
```

（5）コンテナ上で psql コマンドを実行し、スクリプトを使用して testdb にテーブルを作成します。

```
# psql -U postgres -f /createtable.sql testdb
```

（6）コンテナからログアウトします。

```
# exit
```

2. Tomcat コンテナの起動

（1）Tomcat の公式イメージから appsvr という名前のコンテナを作成して起動させます。
--link オプションを使用し、appsvr コンテナから dbsvr コンテナに対して通信する際に、ホスト名として dbsvr を使用できるようにします。

```
# docker run -d --name appsvr --link dbsvr --expose 8009 tomcat:latest
```

（2）appsvr コンテナに Web アプリケーションを配備します。freemarknans というフォルダに Web アプリケーションのファイルが格納されていますので、ディレクトリをまるごとコピーします。

```
# docker cp $HOME/freemarknans appsvr:/usr/local/tomcat/webapps
```

（3）appsvr コンテナに Tomcat の設定ファイル（server.xml）をコピーします。コピーする server.xml には、Apache と連携するための設定が記述されています。

```
# docker cp $HOME/server.xml appsvr:/usr/local/tomcat/conf
```

（4）appsvr コンテナを再起動します。

```
# docker restart appsvr
```

3. Apache コンテナの起動

（1）Apache の公式イメージから websvr という名前のコンテナを作成して起動させます。
--link オプションを使用し、websvr コンテナから appsvr コンテナに対して通信する際に、ホスト名として appsvr を使用できるようにします。また、外部からアクセスするために NAPT を設定します。

```
# docker run -d --name websvr --link appsvr -p 8080:80 httpd:latest
```

（2）Tomcat との連携の設定を記述した設定ファイル（httpd.conf）を websvr コンテナにコピーします。

```
# docker cp $HOME/httpd.conf websvr:/usr/local/apache2/conf
```

（3）websvr コンテナを再起動します。

```
# docker restart websvr
```

4. Web アプリケーションのトップページの表示

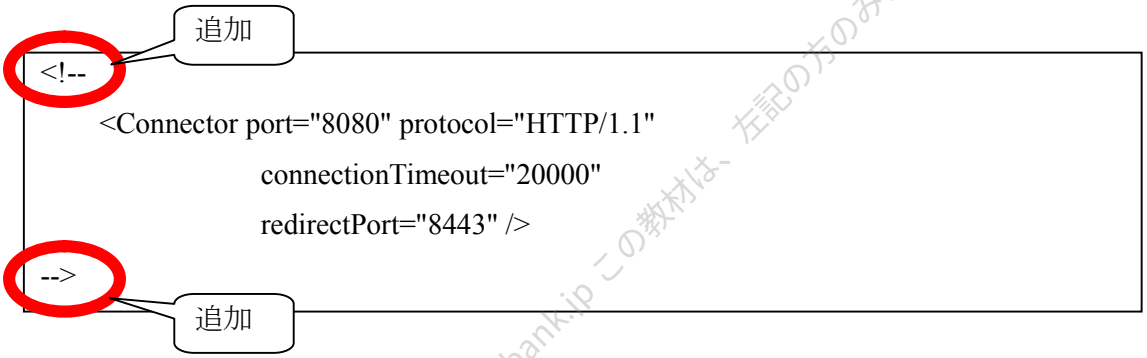
（1）以下の URL に Windows の Web ブラウザで接続します。

http://<Linux ホストの IP アドレス>:8080/freemarknans/jsp/index.html

（2）トップページに表示される画像が、商品リストへのリンクとなっていますので、クリックし、Web アプリケーションが動作することを確認してください。

＜参考＞　Tomcat の /usr/local/tomcat/conf/server.xml の内容

Tomcat は標準で 8080 番ポートを使用しますが、Apache と連携する場合は 8009 番ポートを使用します。server.xml の以下の設定をコメントアウトし、8080 番ポートを使用しない設定にします。



＜参考＞　Apache の /usr/local/apache2/conf/httpd.conf の内容

Tomcat と連携するために、モジュールの有効化や、Tomcat に配備した Web アプリケーションのロケーションの追加、docker run コマンドの --link オプションにより使用できるようになった appsvr の IP アドレスを表す環境変数の指定などを記述します。

（a）以下の 2 行の先頭の # を削除してコメントを解除し、モジュールを有効化します。

```
#LoadModule proxy_module modules/mod_proxy.so
```

```
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
```

(b) 以下の設定を追加し、httpd.conf ファイル内で環境変数を使用可能にします。環境変数名に含まれるコンテナ名は大文字で記述します。

```
PassEnv APPSVR_PORT_8009_TCP_ADDR
```

(c) 以下の設定を追加し、配備した Web アプリケーションのロケーションを設定します。

```
<Location /freemarknans/>
ProxyPass ajp://${APPSVR_PORT_8009_TCP_ADDR}/freemarknans/
</Location>
```

＜参考＞　Web アプリケーション内での接続先のデータベースの指定

Tomcat に配備した Web アプリケーションでは、
freemarknans/WEB-INF/classes/market/dao/DB.properties ファイル内で接続先のデータベースが指定されています。

```
DRIVERNAME=org.postgresql.Driver
URL=jdbc:postgresql://dbsvr:5432/testdb
USER=postgres
```

docker run の --link オプションにより、appsvr コンテナでは dbsvr をデータベースのコンテナのホスト名として使用することができます。

Lab 4-1

コンテナからのイメージの作成

- コンテナをイメージとして保存します
- 保存したイメージからコンテナを作成し、起動します

1. websvr コンテナの commit

（1）現在起動中の websvr コンテナを停止します。

```
# docker stop websvr
```

（2）ローカルに保存されているイメージの一覧を表示します。

```
# docker images
```

（3）websvr コンテナを web-server:1.0 という名前でイメージに保存します。

```
# docker commit websvr web-server:1.0
```

（4）ローカルに保存されているイメージの一覧を表示し、イメージが追加されたことを確認します。

```
# docker images
```

2. web-server:1.0 からのコンテナの作成

（1）イメージ web-server:1.0 から websvr2 コンテナを作成します。NAPT の設定は 8080 番ポートではなく、8000 番ポートを使用します。

```
# docker run -d --name websvr2 --link appsvr -p 8000:80 web-server:1.0
```

（2）以下の URL に接続し、websvr コンテナが停止しているため Web ページが表示できないことを確認します。

http://<Linux ホストの IP アドレス>:8080/freemarknans/jsp/index.html

（3）以下の URL に接続し、websvr2 コンテナにより Web ページが表示されることを確認します。

http://<Linux ホストの IP アドレス>:8000/freemarknans/jsp/index.html

websvr コンテナから作成したイメージを使用しているため、httpd.conf ファイルに編集を加えずに Tomcat と連携できます。

Lab 4-2

Docker Hub へのイメージのアップロード

・ Docker Hub にログインし、作成したイメージをアップロードします

(1) Docker Hub にイメージをアップロードするには、イメージ名を「ユーザー名/リポジトリ名:タグ」という形式にします。Lab 4-1 で作成したイメージに新たに名前を付与します。

```
# docker tag web-server:1.0 flmuser/training:<座席番号>
```

(2) イメージの一覧を表示し、web-server イメージと flmuser/training:<座席番号>イメージのイメージ ID が同じであることを確認します。

```
# docker images
```

(3) Docker Hub にログインします。

```
# docker login
```

ユーザー名： flmuser

パスワード： 講師からお伝えします

(4) Docker Hub にイメージをアップロードします。

```
# docker push flmuser/training:<座席番号>
```

(5) Docker Hub でイメージを検索します。

```
# docker search flmuser
```

Lab 4-3

ローカルイメージの書き出しとリストア

- ローカルイメージを tar 形式で書き出します
- イメージを削除し、書き出したファイルからリストアします

1. ローカルイメージのファイルへの書き出し

(1) web-server イメージを tar 形式で書き出し、gzip コマンドで圧縮します。

```
# docker save web-server:1.0 | gzip > export.tar.gz
```

(2) アーカイブファイルの内容を確認します。

```
# tar tvfz export.tar.gz
```

(3) ローカルイメージを削除する前に、そのイメージから作成されたコンテナを削除します。
websvr2 コンテナを停止し、削除します。

```
# docker stop websvr2
# docker rm websvr2
```

(4) イメージから web-server:1.0 という名前を削除します。

```
# docker rmi web-server:1.0
```

(5) イメージから flmuser/training:<座席番号> という名前を削除し、イメージを削除します。

```
# docker rmi flmuser/training:<座席番号>
```

(6) docker save コマンドで書き出したイメージをリストアします。

```
# docker load -i export.tar.gz
```

(7) イメージの一覧を表示し、リストアされたことを確認します。

```
# docker images
```

Lab 4-4

コンテナの書き出しとリストア

- コンテナを tar 形式で書き出します
- 書き出されたファイルからイメージとしてリストアします

1. コンテナの tar 形式での書き出し

(1) websvr コンテナを tar 形式で書き出し、gzip コマンドで圧縮します。

```
# docker export websvr | gzip > httpd.tar.gz
```

(2) アーカイブファイルの内容を確認します。

```
# tar tvfz httpd.tar.gz
```

docker save コマンドで書き出されたファイルとは異なり、コンテナのファイルシステムがそのままアーカイブファイルに含まれています。

(3) アーカイブファイルから、イメージ web-server:2.0 を作成します。

```
# docker import httpd.tar.gz web-server:2.0
```

(4) イメージ web-server:2.0 から websvr3 コンテナを作成します。しかし、作成に失敗します。

```
# docker run -d --name websvr3 web-server:2.0
```

イメージの作成に、コンテナのファイルシステムをアーカイブファイルに書き出したものを使用したため、コンテナ起動時に実行するプロセスの指定など、イメージの構成情報が失われてしまっています。コンテナを作成するには、起動時に実行するコマンドの指定を行う必要があります。

(5) コンテナ起動時に実行するコマンドを、コンテナ作成の際に指定します。

```
# docker run -d --name websvr3 web-server:2.0 ¥
  /usr/local/apache2/bin/httpd -DFOREGROUND
```

(6) コンテナ起動時に実行するコマンドを、アーカイブからイメージを作成する際に指定し、イメージ web-server:3.0 を作成します。

```
# docker import -c "CMD /usr/local/apache2/bin/httpd -DFOREGROUND" ¥
  httpd.tar.gz web-server:3.0
```

-c オプションで Dockerfile で使用する命令を指定できます。

(7) イメージ web-server:3.0 からコンテナが作成可能であることを確認します。

```
# docker run -d --name websvr4 web-server:3.0
```

Lab 4-5

Dockerfile を使用したイメージ作成

- APP サーバ用の Dockerfile を作成します
- Dockerfile からイメージを作成します

1. Dockerfile の作成

(1) 作業用ディレクトリを作成します。

```
# mkdir $HOME/build-dir
```

(2) APP サーバ内に配備する Web アプリである freemarknans と設定ファイル server.xml を作業用ディレクトリにコピーします。

```
# cp -r $HOME/freemarknans $HOME/server.xml $HOME/build-dir
```

(3) build-dir ディレクトリ内に、以下のような条件を満たす Dockerfile を作成します。

- 公式の Tomcat のイメージ(イメージ名 :tomcat)をベースとする
- server.xml ファイルを /usr/local/tomcat/conf/ ディレクトリ内にコピーする
- freemarknans ディレクトリを /usr/local/tomcat/webapps/ ディレクトリ内にコピーする
- コンテナ起動時に、/usr/local/tomcat/bin/catalina.sh run を実行する

2. イメージの作成とコンテナの作成

(1) Dockerfile を使用し、新たなイメージ myapp を作成します。

```
# docker build -t myapp $HOME/build-dir
```

(2) イメージの一覧を表示し、イメージが追加されたことを確認します。

```
# docker images
```

(3) 一旦、すべてのコンテナを停止させます。

```
# docker stop `docker ps -q`
```

（4）dbsvr コンテナを起動させます。

```
# docker start dbsvr
```

（5）イメージ myapp から appsvr2 コンテナを作成し、起動します。

```
# docker run -d --name appsvr2 --link dbsvr --expose 8009 myapp
```

（6）--link オプションで appsvr2 と連携する websvr5 コンテナを作成します。appsvr2 には appsvr という別名をつけます。

```
# docker run -d --name websvr5 --link appsvr2:appsvr -p 8080:80 web-server:1.0
```

（7）以下の URL に Windows の Web ブラウザで接続します。

http://<Linux ホストの IP アドレス>:8080/freemarknans/jsp/index.html

＜参考＞ Dockerfile の例

```
FROM tomcat

COPY freemarknans /usr/local/tomcat/webapps/freemarknans/
COPY server.xml /usr/local/tomcat/conf/

ENTRYPOINT /usr/local/tomcat/bin/catalina.sh run
```

Lab 5-1

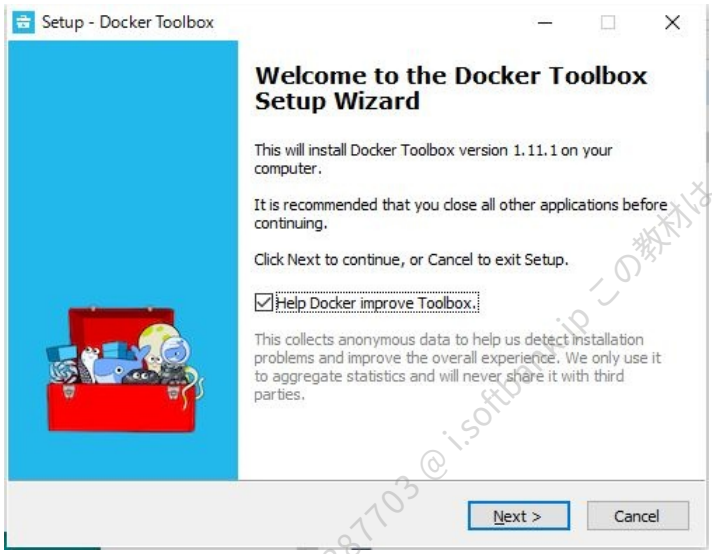
Docker Toolbox の Window へのインストール

・ Docker Toolbox を Windows にインストールし、Docker コンテナの実行環境を用意します

1. Docker Toolbox のインストール

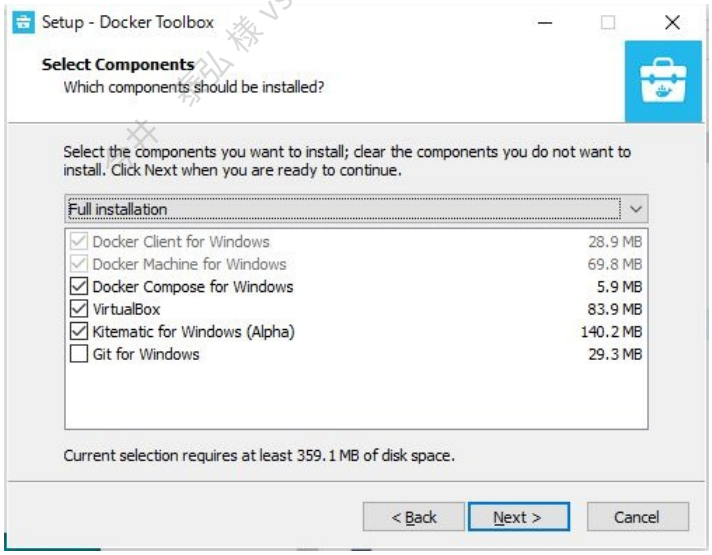
(1) ダウンロード済みの Docker Toolbox のインストーラーを起動し、以下の手順に従いインストールします。

ようこそ



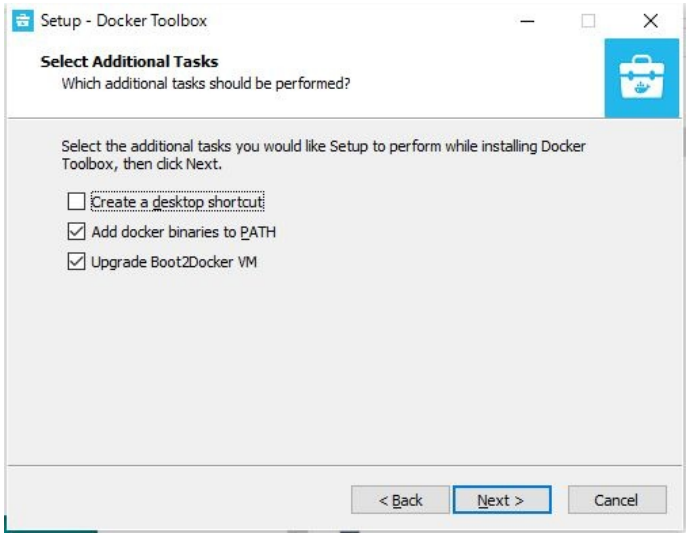
[Next] をクリック。

コンポーネントの選択



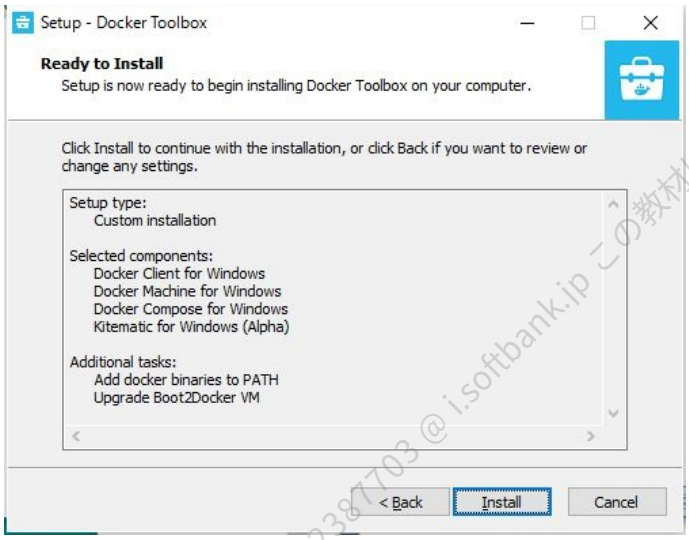
デフォルトのまま [Next] をクリック。

追加のタスク



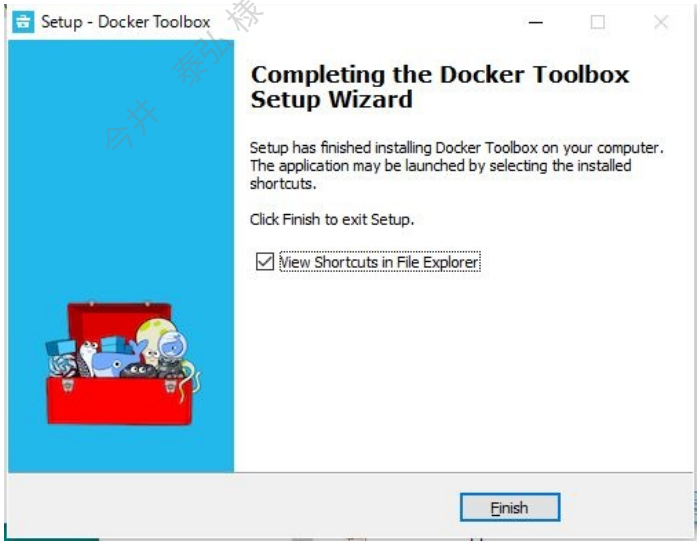
一番上の[Create a desktop shortcut]は必要に応じて選択。
残りの2つは選択したままで[Next]。

インストールの準備完了



[Install] をクリック。

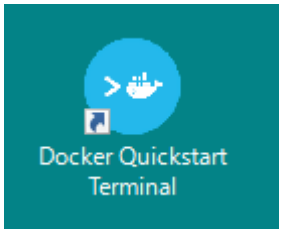
インストール完了



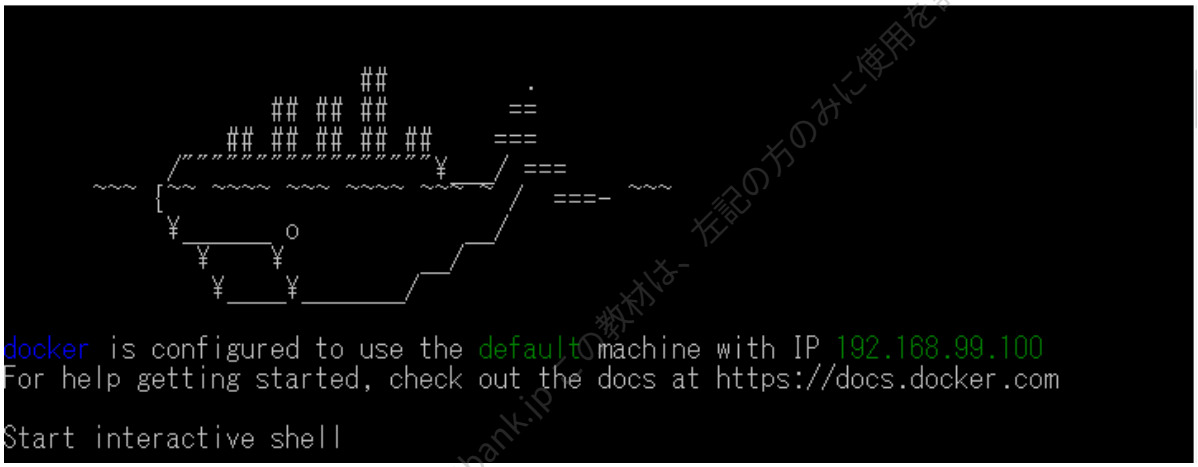
[Finish]をクリック。

2. Docker Quickstart Terminal の起動

スタートメニューやデスクトップのアイコンから Docker Quickstart Terminal を起動します。



初回起動時に、Virtualbox 上に Linux の仮想マシンが作成され、Docker を実行することができるようになります。Docker Quickstart Terminal から Docker コマンドの実行が可能です。



Lab 5-2

Docker Toolbox の操作

ここまでの実習内容をもとに、Docker Toolbox を操作してください。Docker Toolbox は開発者のローカルの開発環境として使用されることが想定されます。実習環境の Linux ホストと Docker Toolbox との間で、tar 形式で書き出したイメージやコンテナのファイルシステムをやりとりするなど、自由に操作を行ってください。