Micro Focus Visual COBOL チュートリアル

COBOL 開発：コンテナを利用した DB アプリケーション開発

1. 目的

コンテナ技術により、Linux カーネルのコンテナ機能を使って実行環境を他のプロセスから隔離し、その中でアプリケーションを動作させることができます。また、コンテナプロセスの起動に必要なシステム資源は、仮想マシンの起動と比較すると非常に軽量です。コンテナ技術の利用により、アプリケーションとライブラリを同一のコンテナ内に固められるため、容易にアプリケーションの移動やディプロイが行えます。

Visual COBOL は、コンテナ技術として Docker、もしくは Podman を利用することができます。

コンテナ技術を利用することにより COBOL 開発に以下の利点を提供します。

- 開発・実行環境をイメージで保持するため、CI ツールとの連携による日々の自動テストや回帰テストの実施や、同一環境の複数立ち上げが非常に容易
- バージョン毎にイメージが作成されるため、パッチアップデートを含めたバージョンアップ検証作業において複数環境構築が不要

本チュートリアルでは、コンテナ環境内で ODBC を利用したデータベースアクセスを伴うアプリケーション開発手順を説明します。

2. 前提

- 本チュートリアルで使用したマシン OS ： Red Hat Enterprise Linux 7.6 / Red Hat Enterprise Linux 8.2
- Linux 環境に Red Hat のサブスクリプション情報が登録済みであること
- Visual COBOL 7.0 Development Hub 製品をご購入のお客様
- ODBC を利用したデータベース接続が行える環境を構築できること
- コンテナコマンドの知識があること
- 別チュートリアル「ステップバイステップチュートリアル - コンテナを利用した開発」、および、「COBOL 開発 - コンテナを利用した SOA 開発」を実施済みであること

また、本文書では、製品コンテナイメージをインストールするホスト OS 環境を Red Hat Enterprise Linux 7.x, 8.x 、もしくは、単に 7.x, 8.x と記載していますが、製品コンテナイメージがサポートする環境は Red Hat Enterprise Linux 7.4 以降, もしくは Red Hat Enterprise Linux 8.0 以降となります。また、各 OS 環境とコンテナサポートバージョンについては、Red Hat 社サイトにてご確認ください。

本チュートリアルでは、一部の手順において、下記リンク先のサンプルファイルを使用します。事前にダウンロードをお願いします。

サンプルプログラムのダウンロード

3. チュートリアルの流れ

本章では、製品コンテナイメージを利用した開発に必要な操作や機能を以下のステップを踏みながら確認していきます。

1. 事前準備について
2. コンソールアプリケーションの実行
3. SOA アプリケーションの実行

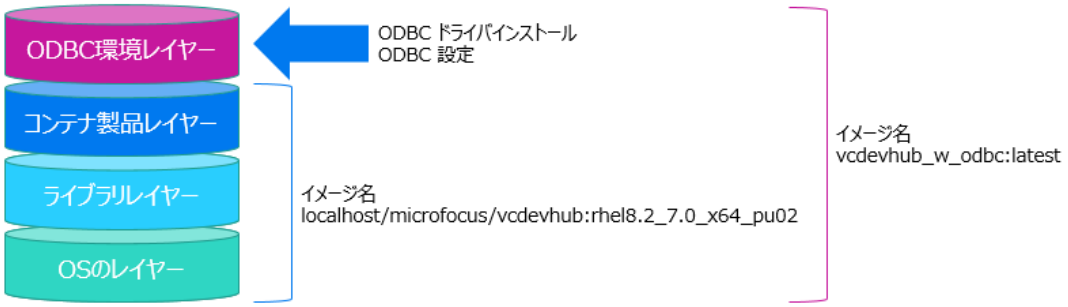
Red Hat Enterprise Linux 7.x, 8.x いずれの環境をご利用のお客様も本チュートリアルは実施できますが、説明時にコンテナコマンドの違いを吸収するため、以下の用語を使用します。

<コンテナコマンド>: 7.x 環境のお客様は docker, 8.x 環境のお客様は podman に読み替えてください。

また、本チュートリアルでは、コンテナイメージは RHEL 8.x 上で Visual COBOL 7.0J の PatchUpdate02 を使用しています。異なるバージョンをご利用のお客様は、適宜、イメージ名を読み替えてください。

3.1. 事前準備について

本チュートリアルでは、データベースと連携したアプリケーションを実行するため、事前にデータベースサーバーの稼働と、データベースに接続するための ODBC ドライバなどの設定がコンテナ環境内で必要です。これらの設定を事前に行ったコンテナイメージを作成しておくことで、都度環境構築作業が不要になります。以降の手順では、このイメージを vcdevhub_w_odbc:latest として作成済みであることを前提に説明させていただきます。



注意）
3.2 では、対話形式で COBOL プログラムのコンパイル・実行を行います。
このため、vcdevhub_w_odbc:latest のコンテナイメージは対話形式での操作ができるよう作成してください。

本チュートリアルでは、MySQL 5.7 をデータベースサーバーとして利用し、unixODBC を利用した開発を行っています。
しかし、ODBC 設定により、異なるデータベースサーバーを利用頂くことも可能です。

3.1.1. チュートリアルで使用するアプリケーションについて

別チュートリアル「COBOL 開発： コンテナを利用した SOA 開発」で使用した書籍情報管理アプリケーションをもとに、書籍データを索引ファイルからデータベースに変更したものとなります。

この書籍情報を格納するためのテーブルとして、以下のテーブルを事前に作成してください。

- スキーマ名： vcdemo
- テーブル名： book_info

カラム名	型	説明
stock_no	CHAR(4) NOT NULL	書籍番号 一意キー
title	VARCHAR(50) NOT NULL	書籍名
type	VARCHAR(20) NOT NULL	ジャンル
author	VARCHAR(50) NOT NULL	著者
retail	INT(5) NOT NULL	販売数
onhand	INT(5) NOT NULL	在庫数
sold	INT(5) NOT NULL	売上数

デモで参照するデータは、以下の SQL を実行して登録できます。

```
insert into book_info values ('1111', 'LOAD OF THE RINGS', 'FANTASY', 'TOKLKIEN', 1500, 4000, 3444);
insert into book_info values ('2222', 'OLIVER TWIST', 'FICTION', 'DICKENS', 1000, 3000, 2333);
```

3.2. コンソールアプリケーションの実行

本節では、データベース接続を行うコンソールアプリケーションの開発を行うことで、製品コンテナイメージを利用した開発も、従来同様の開発で行えることの確認を目的としています。このため、書籍情報テーブルに登録された情報をコンソール出力する簡単な COBOL アプリケーションを実行していきます。

1) ダウンロードしたサンプルファイルを任意のディレクトリに解凍したうえで、解凍されたディレクトリに移動します。

ここでは、サンプルファイル内の以下のリソースを使用します。

リソース名	説明
consoleapp/BookConsole.cbl	データベースにアクセスを行い、書籍情報を取得、一覧表示する COBOL プログラム
consoleapp/BOOK-INFO-HOSTVAR.cpy	上記 COBOL プログラムが参照するコピーブックファイル

2) consoleapp/BookConsole.cbl を開き、6~8 行目に記載されているデータベース接続情報をお客様の環境に合わせ修正してください。

```
5:          01 DB-CONNECTION.
6:             03 ODBC-NAME      PIC X(20) VALUE "DEMODB".
7:             03 DB-USER        PIC X(20) VALUE "user".
8:             03 DB-PASSWORD PIC X(20) VALUE "password".
```

補足）
初期値として、データソース名を “DEMODB”、ユーザーを “user”、パスワードを “password” としています。

3) 以下のコマンドを実行し、コンテナ環境を起動します。

<コンテナコマンド> run --rm -it -v $PWD/consoleapp:/tmp/app:z vcdevhub_w_odbc bash

```
# pwd
/tmp/vc-containertutorial03
[root@localhost vc-containertutorial03] # podman run --rm -it -v
$PWD/consoleapp:/tmp/app:z vcdevhub_w_odbc bash
[root@27fac1faf6b0 /]#
```

4) バインドマウントを行った /tmp/app ディレクトリに移動します。

cd /tmp/app

```
# cd /tmp/app/
[root@27fac1faf6b0 app]# ls
BOOK-INFO-HOSTVAR.cpy   BookConsole.cbl
```

ホスト環境上の COBOL プログラムとコピーブックが確認できます。

5) COBOL 開発に必要な環境変数を設定します。

. /opt/microfocus/VisualCOBOL/bin/cobsetenv

```
# . /opt/microfocus/VisualCOBOL/bin/cobsetenv
COBDIR set to /opt/microfocus/VisualCOBOL
[root@27fac1faf6b0 app]#
```

6) 以下のコマンドを実行し、プログラムのコンパイルを行います。

cob -x BookConsole.cbl

```
# cob -x BookConsole.cbl
[root@27fac1faf6b0 app]# ls
BOOK-INFO-HOSTVAR.cpy   BookConsole   BookConsole.cbl   BookConsole.idy
BookConsole.int   BookConsole.o
```

コンパイルが行われ、実行モジュールが作成されます。

7) プログラムの実行を行います。

./BookConsole

```
# ./BookConsole
*** BOOKINFO ***
  STOCK_NO: 1111
  TITLE:      LOAD OF THE RINGS
  TYPE:       FANTASY
  AUTHOR:    TOKLKIEN
  RETAIL:    01500
  ONHAND:    04000
  SOLD:      03444
*** BOOKINFO ***
  STOCK_NO: 2222
  TITLE:      OLIVER TWIST
  TYPE:       FICTION
  AUTHOR:    DICKENS
  RETAIL:    01000
  ONHAND:    03000
  SOLD:      02333
[root@mypod app]#
```

テーブル内のレコード情報がコンソールに出力されます。

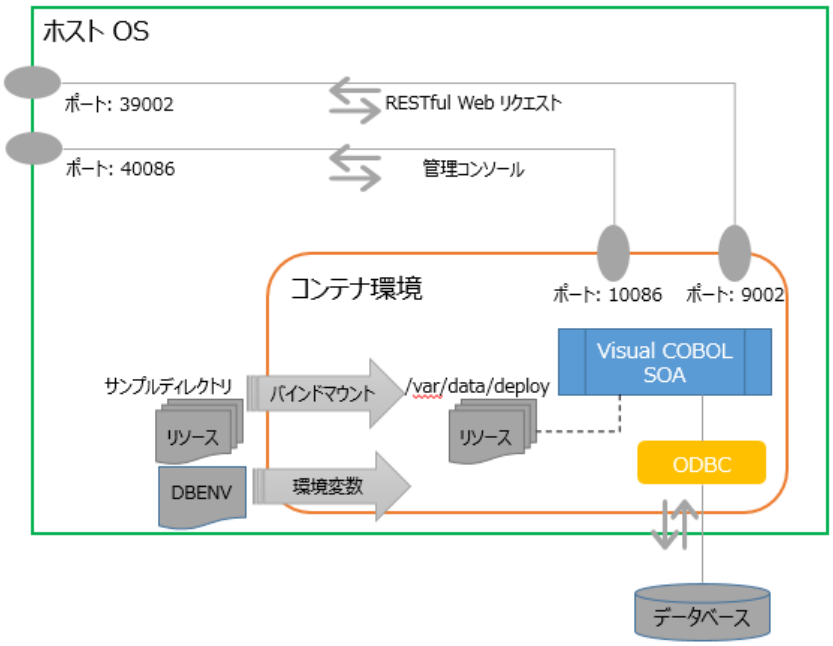
8) コンテナン環境の停止・破棄を行います。

exit

```
# exit
exit
[root@localhost vc-containertutorial03]#
```

3.3. SOA アプリケーションの実行

前節では、コンテナ内でのデータベース接続アプリケーションであっても、従来同様の開発手順で開発が行え、実行ができることを確認しました。本節では、コンテナ内で書籍情報テーブルを参照する SOA アプリケーションを稼働させ、書籍情報の新規追加や、参照・削除機能を API 経由で提供するデモ環境を構築します。アプリケーション構成は、以下のようになります。



コンテナ環境内に SOA アプリケーション環境を構築しているため、任意のタイミングで迅速に環境を起動でき、複数環境の同時稼働といったことも容易に実現できるようになります。

提供される API は以下の通りです。

サービス名	API 例

書籍情報検索	http://localhost:39002/temppath/BookRestDBWS/1.0/SearchBook
	リクエストデータ例)
	{
	"LNK_B_STOCKNO": "1111"
	}

書籍情報追加	http://localhost:39002/temppath/BookRestDBWS/1.0/AddBook
	リクエストデータ例
	{
	"LNK_B_DETAILS":
	{
	"LNK_B_TEXT_DETAILS":
	{
	"LNK_B_TITLE": "Alice's Adventures in Wonderland",
	"LNK_B_TYPE": "Fantasy",
	"LNK_B_AUTHOR": "Lewis Carroll",
	},
	"LNK_B_STOCKNO": 9999,
	"LNK_B_RETAIL": 100,
	"LNK_B_ONHAND": 200,
	"LNK_B_SOLD": 300,
	}
	}

書籍情報削除	http://localhost:39002/temppath/BookRestDBWS/1.0/DeleteBook
	リクエストデータ例)
	{
	"LNK_B_STOCKNO": "9999"
	}

1) ダウンロードしたサンプルファイルを任意のディレクトリに解凍したうえで、解凍されたディレクトリに移動します。

ここでは、サンプルファイル内の以下のリソースを使用します。

リソース名	説明
Containerfile / Dockerfile	コンテナイメージとして、SOA アプリケーション環境を構築するための定義ファイル 定義内容は両ファイルで同様だが、コンテナコマンドによって参照されるファイルが異なる。
DBENV	SOA アプリケーション内で参照する ODBC 設定情報 コンテナ内では環境変数として定義され、SOA アプリケーションは本情報を参照してデータベースに接続する。
req/*	RESTful ウェブサービス実行時のリクエストデータ例
setup/DEMOSV.xml	サンプルアプリケーションを稼働させるためのアプリケーションサーバーインスタンスの定義ファイル setup.sh 内でインポート処理が行われる。
setup/.mfdeploy	サンプルアプリケーションをディプロイするための設定ファイル
setup/deploy/*	サンプルアプリケーションのアーカイブファイル setup.sh 内でアプリケーションサーバーに登録される。
setup/script/setup.sh	コンテナイメージを作成する際に、コンテナ内部で実行されるセットアップ内容を記載したスクリプト

2) 以下のコマンドを実行し、SOA アプリケーション環境のコンテナイメージを作成します。

<コンテナコマンド> build -t vcdemo03 .

```
# podman build -t vcdemo03 .
STEP 1: FROM vcdevhub_w_odbc
STEP 2: RUN   mkdir -p /inst && mkdir -p /var/data/deploy
--> 56985cbba13
STEP 3: ADD   setup/script /inst/script
--> cf3d35052cd
STEP 4: COPY setup/DEMOSV.xml /inst
--> 6b85c26bf16
STEP 5: COPY setup/.mfdeploy /var/data
--> 980db763f9c
STEP 6: COPY setup/deploy /var/data/deploy
--> fc9e584dd14
STEP 7: RUN   sh /inst/script/setup.sh
COBDIR set to /opt/microfocus/VisualCOBOL
Processing -g option...
(c) Copyright 1991 - 2020 Micro Focus or one of its affiliates.
Micro Focus Directory Server daemon: Version 1.26.55
```

```
Using:
        Repository Type = XML

        Import Path = /inst/DEMOSV.xml
        Options = O
        User ID = [not specified]
        Password = [not specified]



Response = 0 1.
Import requested processed. Check journal file and export history for full result.

1 server(s) imported.
Processing -s option...
(c) Copyright 1991 - 2020 Micro Focus or one of its affiliates.
Micro Focus Directory Server daemon: Version 1.26.55
PR7007I Changing effective process uid to "esadm"PR0007I Process is using UID 1010STEP
8: COMMIT vcdemo03
--> 853ae6524d6
853ae6524d61bf6a6ba04f60f3f6f0349825aea76186e99e336402cd731f86d2
[root@localhost vc-containertutorial03]#
```

vcdemo03 コンテナイメージが作成されていることを確認します。

<コンテナコマンド> images |grep vcdemo03

```
# podman images |grep vcdemo03
localhost/vcdemo03                          latest                      853ae6524d61   2
minutes ago   1.25 GB
```

3) 必要に応じて、ODBC 設定情報ファイル DBENV の中身を更新してください。

キー名	値
ODBC_NAME	データベースサーバーへの接続情報を定義したデータソース名
DB_USER	データベースの認証ユーザー
DB_PASSWORD	データベースの認証パスワード

補足）
この定義は、次のコンテナ環境起動時に参照され、キー名称・値のペアで環境変数として定義されます。
SOA アプリケーションは、環境変数を参照してデータベース接続情報を取得しています。

4) 以下のコマンドを実行して、コンテナ環境を起動します。

以下のコマンドは 1 行で実行してください。

<コンテナコマンド> run --rm -itd -p 40086:10086 -p 39002:9002 --name vcdev --env-file DBENV vcdemo03

```
# podman run --rm -itd -p 40086:10086 -p 39002:9002 --name vcdev --env-file DBENV vcdemo03
d939d490364e955569c2a2d663bc5fb29871b713ba6b5ad1253e16db3e7fbea7
```

5) 以下のコマンドを実行して、コンテナ内でアプリケーションサーバーを立ち上げ、SOA アプリケーションを起動します。

<コンテナコマンド> exec -d vcdev sh -c ". ¥$MFPRODBASE/bin/cobsetenv && mfds"

<コンテナコマンド> exec vcdev sh -c ". ¥$MFPRODBASE/bin/cobsetenv && casstart /rDEMOSV"

```
# podman   exec -d vcdev sh -c ". ¥$MFPRODBASE/bin/cobsetenv && mfds"
778a8e3b535a1e148e67fea59e02e7d73d55345db568c89f686d5feb7958a783
[root@localhost vc-containertutorial03]# podman exec vcdev sh -c ". ¥$MFPRODBASE/bin/cobsetenv && casstart /rDEMOSV"
COBDIR set to /opt/microfocus/VisualCOBOL
..
CASCD0167I ES Daemon successfully auto-started 06:21:22
CASCD1005I /var/mfcobol/es/DEMOSV/console.log 06:21:22
CASCD0050I ES "DEMOSV" initiation is starting 06:21:22
[root@localhost vc-containertutorial03]#
```

6) リクエストデータ例で紹介したリクエストを実行し、正常に処理が行われていることを確認します。

書籍情報検索）

以下のコマンドは 1 行で実行してください。

curl -X POST http://localhost:39002/temppath/BookRestDBWS/1.0/SearchBook -d @req/SearchBook

```
# curl -X POST http://localhost:39002/temppath/BookRestDBWS/1.0/SearchBook -d @req/SearchBook
{
  "LNK_B_DETAILS" :
  {
    "LNK_B_TEXT_DETAILS" :
    {
      "LNK_B_TITLE" : "LOAD OF THE RINGS",
      "LNK_B_TYPE" : "FANTASY",
      "LNK_B_AUTHOR" : "TOKLKIEN"
    },
    "LNK_B_STOCKNO" : "1111",
    "LNK_B_RETAIL" : 1500,
```

```
    "LNK_B_ONHAND" : 4000,
    "LNK_B_SOLD" : 3444
  },
  "LNK_STATUS" :
  {
    "LNK_SQLCODE" : 0,
    "LNK_PHASE" : "DO-READ"
  }
}
```

書籍情報追加）

curl -X POST http://localhost:39002/temppath/BookRestDBWS/1.0/AddBook -d @req/AddBook

```
# curl -X POST http://localhost:39002/temppath/BookRestDBWS/1.0/AddBook-d
@req/AddBook
{
  "LNK_STATUS" :
  {
    "LNK_SQLCODE" : 0,
    "LNK_PHASE" : "DO-ADD"
  }
}
```

書籍情報削除）

以下のコマンドは１行で実行してください。

curl -X POST http://localhost:39002/temppath/BookRestDBWS/1.0/DeleteBook -d @req/DeleteBook

```
# curl -X POST http://localhost:39002/temppath/BookRestDBWS/1.0/DeleteBok -d
@req/DeleteBook
{
  "LNK_STATUS" :
  {
    "LNK_SQLCODE" : 0,
    "LNK_PHASE" : "DO-DELETE"
  }
}
```

7) コンテナ環境を停止・破棄します。

<コンテナコマンド> stop vcdev

```
# podman stop vcdev
d939d490364e955569c2a2d663bc5fb29871b713ba6b5ad1253e16db3e7fbea7
```

4. 補足

4.1. setup/script/setup.sh

```
 1: #!/bin/sh
 2: . $MFPRODBASE/bin/cobsetenv
 3: mfds &
 4: sleep 5
 5: mfds /g 5 /inst/DEMOSV.xml O
 6: cd /var/data/deploy && mfdepinst BookRestDBWS.car
 7: mfds /s 1
 8: while :
 9: do
10:     PROCINF=`ps -efa |grep mfds |grep -v grep`
11:     if [ "$PROCINF" == "" ]; then
12:         break
13:     fi
14:     sleep 1
15: done
16: exit 0
17:
```

行数	説明
2	Visual COBOL 製品を使用するために必要な環境変数を設定しています。
3~6	サンプルで使用するアプリケーションサーバーインスタンス DEMOSV の定義をインポートしたうえで、アプリケーションファイル BookRestDBWS.car のディプロイを行います。その後、アプリケーションサーバーを停止します。
7~15	アプリケーションサーバーの完全停止を確認しています。

WHAT'S NEXT

- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

免責事項