Micro Focus Enterprise Developer チュートリアル

メインフレーム COBOL 開発

コンテナ型仮想化の利用 CICS, JCL, IMS 編

1. 目的

コンテナ型仮想化では、ホストマシンとカーネルを共有しながら、アプリケーションやライブラリなどを含むコンテナをコンテナエンジン上でプロセスとして複数稼働させることができます。この技術により、次のようなメリットを享受することができます。

- ✓ 移植性

  コンテナ内でテストが完了したアプリケーションを他のコンテナへ移植し、同じように動作させることができます。

- ✓ パフォーマンス

  コンテナ内に OS を含まないことによりリソースが軽減され、コンテナを迅速に作成でき、すばやく起動できます。

- ✓ 機敏性

  移植性およびパフォーマンスの利点により開発プロセスが早くなります。また、継続的インテグレーション（CI）を活用することによりアプリケーションの更新や提供を迅速に行えるようになります。

- ✓ 分離性

  コンテナごとに異なるバージョンのソフトウェアをインストールして使用することができます。各コンテナは完全に独立しているため、環境の保全性が確保されます。

- ✓ スケーラビリティ

  ニーズに合わせて新しいコンテナをすばやく作成できます。

Enterprise Developer のコンテナ製品は、製品本体とは異なるコンテナ製品とコンテナ用ライセンスが必要になります。また、コンテナ製品では、製品本体がインストールされたコンテナイメージをご提供しています。

本チュートリアルではこのベースイメージを利用して製品に含まれる CICS コンテナデモンストレーションのコンテナイメージを生成します。このイメージには CICS, JCL, IMS アプリケーションが含まれており、これらの実行をご体験いただくことを目的としています。

2. 実行環境と前提

- ホスト OS ： Red Hat Enterprise Linux Server release 8.4
- Enterprise Developer 8.0J for Linux and UNIX のコンテナ製品とライセンスを別途入手済であること
- ホストマシンに Red Hat のサブスクリプション情報が登録済みであること
- コンテナコマンドの知識があること
- CICS チュートリアルを終了していること
- JCL チュートリアルを終了していること
- IMS チュートリアルを終了していること
- Linux/UNIX チュートリアルを終了していること

3. 対応 OS とビット数

コンテナ製品では Red Hat Enterprise Linux 7.4 以降と Red Hat Enterprise Linux 8.0 以降の OS を対象として、64 ビットのコンテナイメージをご提供しています。また、7.4 以降は Docker を、8.0 以降は podman を利用することができますが、OS におけるコンテナのサポート状況については Red Hat 社のサイトをご確認ください。本チュートリアルでは 8.4 を利用するため podman コマンドを利用しますが、7.X の場合は Docker コマンドに読み替えて処理を進めてください。

4. コンテナイメージの種類

コンテナ製品はベースイメージと対話型ログインに対応する２種類のイメージをご提供します。対話型ログインイメージを利用すると、実行中のコンテナにログインしてスクリプトやコマンドの実行、開発環境の操作などを対話型で操作することができます。

1) ベースイメージ： タグ名の例）rhel8.4_8.0_x64
2) 対話型ログインのベースイメージ： タグ名の例）rhel8.4_8.0_x64_login
3) OS レイヤー： 8.X のレポジトリ名）localhost/ubi8/ubi-minimal

本チュートリアルではデモイメージを生成する Containerfile に記述されているベースイメージを使用します。Containerfile の所在については後述で確認します。

5. パッチアップデートの適用

コンテナ製品のパッチアップデートでは、製品本体がインストールされたコンテナイメージをご提供しているため、パッチアップデートごとに異なるタグ名が付けられ、製品本体の不具合が吸収されたコンテナイメージが提供されます。製品本体のパッチアップデートの適用方法とは異なりますのでご注意ください。

例）Red Hat Enterprise Linux 8.4 環境に パッチアップデートなしのコンテナ製品を展開したイメージタグ名

rhel8.4_8.0_x64

例）Red Hat Enterprise Linux 8.4 環境に コンテナ製品の PatchUpdate05 を展開したイメージタグ名

rhel8.4_8.0_x64_login_pu05

6. コンテナの準備

Red Hat Enterprise Linux 7.X では Docker パッケージがインストールされていることが前提になります。未実施の場合はインストールしてください。

インストールコマンド例）

```
subscription-manager repos --enable=rhel-7-server-extras-rpms
yum install docker -y
```

1) コンテナサービスが停止している場合は OS のコマンドを利用して開始後、状態を確認します。

開始コマンド例）

systemctl start podman.service

状態確認コマンド例）

systemctl status podman.service

```
#systemctl status podman.service
* podman.service - Podman API Service
   Loaded: loaded (/usr/lib/systemd/system/podman.service; static; vendor preset: disabled
   Active: active (running) since Wed 2022-07-13 16:15:14 JST; 4s ago
     Docs: man:podman-system-service(1)
 Main PID: 33031 (podman)
```

active (running) から稼働中であることが確認できます。

7. 製品ベースイメージの生成とライセンスの配置

CICS コンテナデモンストレーションのイメージを生成するために、まずは製品のベースイメージを生成します。これに伴いコンテナ用ライセンスの配置が必要になります。

1) 入手済のコンテナ製品を任意のディレクトリへ配置して展開します。

展開コマンド例）tar xvf entdev_dockerfiles_8.0_redhat8_x64.tar

```
#tar xvf entdev_dockerfiles_8.0_redhat8_x64.tar
```

2) 製品のベースイメージを生成するための EntDev ディレクトリと，デモンストレーションイメージを生成するための Examples ディレクトリ、README ファイルが展開されます。

```
#ls
EntDev  Examples  README.html  README.txt  bld.env.rhel  entdev_dockerfiles_8.0_redhat8_x64.tar
```

EntDev ディレクトリへ移動します。

```
#cd EntDev
#ls
Containerfile                                README.html   bld.funcs   cleanup_and_prune.sh  prodver.env
OpenJDK11U-jdk_x64_linux_hotspot_11.0.15_10.tar.gz  README.txt   bld.pfuncs  dotnet_install.sh     setup_entdev_for_docker_8.0_redhat_x64.gz
README-tree.png                              bld.env.rhel  bld.sh      license_install.sh
```

3) 現時点でホストマシンに存在する コンテナイメージを確認します。

コマンド例）podman image ls

```
#podman image ls
REPOSITORY  TAG     IMAGE ID  CREATED  SIZE
#
```

本チュートリアルで使用しているホストマシンにはイメージが存在していません。

4) コンテナ製品のライセンスを EntDev ディレクトリ内へ配置します。

```
#ls
 Containerfile                                README.html   bld.pfuncs            license_install.sh
'Enterprise-Developer-TE-Build-Tools-Docker(PA).mflic'  README.txt   bld.sh                prodver.env
 OpenJDK11U-jdk_x64_linux_hotspot_11.0.15_10.tar.gz  bld.env.rhel  cleanup_and_prune.sh  setup_entdev_for_docker_8.0_redhat_x64.gz
 README-tree.png                              bld.funcs     dotnet_install.sh
```

コンテナ製品では、ベースイメージを生成するシェルスクリプト（本チュートリアルでは bld.sh）と同じディレクトリにコンテナ用のライセンスを配置する必要があります。

このディレクトリにコンテナ用ライセンスが配置されていない場合はエラーとなり、イメージの生成はできません。

エラーメッセージの例）

```
#./bld.sh IacceptEULA
Using environment from bld.env.rhel
./bld.sh: Sorry no .mflic or .xml file found.
 - Expected Enterprise-Developer-TE-Build-Tools-Docker(PA).mflic or another .mflic/.xml file
```

5) 製品のベースイメージを生成し、コンテナイメージを確認します。ここで指定している IacceptEULA オプションは Micro Focus エンドユーザー使用許諾契約（EULA）に同意することを示します。 IacceptEULA を指定しない場合、イメージは生成できません。その他のオプション指定に関しては製品マニュアルをご参照ください。

コマンド例）

./bld.sh IacceptEULA

```
Completed - we have the following microfocus/entdevhub images
 localhost/microfocus/entdevhub              rhel8.4_8.0_x64_login  29b1869dfa9e  3 seconds ago  1.52 GB
 localhost/microfocus/entdevhub              rhel8.4_8.0_x64        488c00443f93  4 minutes ago  1.52 GB

 To use:
   podman run --rm -ti microfocus/entdevhub:rhel8.4_8.0_x64_login
```

生成された 2 つのイメージが追加されていることを確認します。

```
#podman image ls
REPOSITORY                          TAG                   IMAGE ID      CREATED         SIZE
localhost/microfocus/entdevhub      rhel8.4_8.0_x64_login  29b1869dfa9e  7 minutes ago   1.52 GB
localhost/microfocus/entdevhub      rhel8.4_8.0_x64        488c00443f93  12 minutes ago  1.52 GB
```

追加された 2 つのベースイメージタグ名）

rhel8.4_8.0_x64_login

rhel8.4_8.0_x64

8. デモンストレーションイメージの生成

製品のベースイメージを利用して CICS コンテナデモンストレーションイメージを生成します。

1) 複数のコンテナデモンストレーション用ディレクトリが含まれる Examples ディレクトリへ移動します。

```
#cd Examples
#ls
Build_HelloWorld      Build_NET6_HelloWorld  Build_demo_ant   Build_demo_mfunit   CICS
Build_JVM_HelloWorld  Build_PLI_HelloWorld   Build_demo_json  Build_demo_mthread
```

2) CICS コンテナデモンストレーションは CICS ディレクトリに含まれているため、このディレクトリへ移動します。

```
#cd CICS
#ls
Containerfile  MSS64_combined.xml  README.html  README.txt  bld.env.rhel  bld.funcs  bld.sh  bld_package.sh  mssdata  sample_setup  src
```

3) CICS ディレクトリに含まれるファイルを確認します。このディレクトリに含まれている bld.sh を使用してデモンストレーションイメージを生成します。

① mssdata ディレクトリ

CICS アプリケーションで使用する画面定義やデータ ファイルが含まれています。

② src ディレクトリ

CICS アプリケーションで使用するソースファイルが含まれています。

③ bld.env.rhel ファイル

コンテナイメージの構築に使用されるさまざまな環境変数を定義する構成ファイルです。

④ bld.funcs ファイル

bld.sh スクリプトで使用される関数の定義ファイルです。

⑤ bld.sh ファイル

イメージの生成プロセスを自動化するバッチファイルです。

⑥ bld_package.sh ファイル

リビルドしてパッケージ化するシェルスクリプトです。

⑦ Containerfile ファイル

イメージの生成に使用される Containerfile です。

⑧ MSS64_combined.xml ファイル

CICS アプリケーションを稼働させるための Enterprise Server インスタンス定義を含むファイルです。

⑨ README.* ファイル

イメージの生成方法に関する説明を含む HTML およびプレーン テキストのドキュメントのファイルです。

⑩ sample_setup ファイル

Micro Focus Directory Server (以降 MFDS と称す) を起動し、Enterprise Server インスタンスの定義やアプリケーションを設定して、コンテナが終了するまでのコンソールログファイルを表示するスクリプトです。

4) 現在、実行中および停止中のコンテナを確認します。

コマンド例）podman ps –a

```
#podman ps -a
CONTAINER ID  IMAGE   COMMAND  CREATED  STATUS  PORTS   NAMES
#
```

このホストマシンでは、実行中および停止中のコンテナは 1 つもないことが確認できます。

5) bld.sh を実行して CICS コンテナデモのイメージを生成します。コマンドのオプション指定に関しては製品マニュアルをご参照ください。

コマンド例）./bld.sh

```
#./bld.sh
Using environment from bld.env.rhel
Build an MSS image
------------------

Building image : microfocus/ed-mss:rhel8.4_8.0_x64
 STEP 1: FROM microfocus/entdevhub:rhel8.4_8.0_x64 AS builder
```

```
Image microfocus/ed-mss:rhel8.4_8.0_x64 created

To use:
        podman run --rm -p 30086:10086 -p 30040-30050:30040-30050 -d microfocus/ed-mss:rhel8.4_8.0_x64
```

6) イメージを確認すると、レポジトリ名：localhost/microfocus/ed-mss が生成されています。

```
#podman image ls
REPOSITORY                          TAG                 IMAGE ID      CREATED         SIZE
localhost/microfocus/ed-mss         rhel8.4_8.0_x64     6d19845e8179  2 minutes ago   1.52 GB
localhost/microfocus/entdevhub      rhel8.4_8.0_x64_login 29b1869dfa9e  28 minutes ago  1.52 GB
localhost/microfocus/entdevhub      rhel8.4_8.0_x64     488c00443f93  32 minutes ago  1.52 GB
```

7) 生成時に表示されたメッセージを参考に podman コマンドを使用してコンテナを実行します。実行の際に name オプションを利用して、判別が容易な cicsdemo というコンテナ名（任意）を指定します。

コマンド例）

podman run -d --name cicsdemo -p 30086:10086 -p 30040-30050:30040-30050 -d microfocus/ed-mss:rhel8.4_8.0_x64

```
#podman run -d --name cicsdemo -p 30086:10086 -p 30040-30050:30040-30050 -d microfocus/ed-mss:rhel8.4_8.0_x64
3b43668de7f8f8367192030ca07d2e98cf06bca659c5c81012c0f148b700c0ac
```

コンテナの終了と共にコンテナを削除するよう、開始時に指定できる --rm オプションがありますが、本チュートリアルでは一連の操作を実施する目的でこのオプションは指定していません。

8) 実行または停止中のコンテナを確認すると cicsdemo コンテナが実行中であることを確認できます。

コマンド例）podman ps -a

```
#podman ps -a
CONTAINER ID  IMAGE                                    COMMAND              CREATED            STATUS                PO
RTS                                                    NAMES
3b43668de7f8  localhost/microfocus/ed-mss:rhel8.4_8.0_x64  /bin/sh -c /home/...  About a minute ago  Up About a minute ago  0.
0.0.0:30086->10086/tcp, 0.0.0.0:30040-30050->30040-30050/tcp  cicsdemo
```

9. コンテナ内の確認

実行したコンテナ内のアプリケーションを確認します。

1) コンテナ名を指定してコンテナ内へ入り、対話的に内容を確認します。

コマンド例）

podman exec -it cicsdemo bash

```
#podman exec -it cicsdemo bash
[esadm@3b43668de7f8 ~]$
```

コンテナ内に esadm ユーザーで入りました。

2) MSS ディレクトへ移動して内容を確認します。

```
[esadm@3b43668de7f8 ~]$ cd MSS
[esadm@3b43668de7f8 MSS]$ ls
ESJCL.jcl  IMSBATCH.JCL  cbl  cpy  loadlib64
```

JCL, IMS で使用する JCL が 2 本と、ソース類を格納しているディレクトリ、実行可能ファイルを格納しているディレクトリが存在しています。

3) MSS ディレクトリ配下の loadlib64 ディレクトリに実行可能ファイルがあるか確認します。

```
[esadm@3b43668de7f8 MSS]$ cd loadlib64
[esadm@3b43668de7f8 loadlib64]$ ls
ACCT00.so  ACCT02.so  ACCT04.so    DEMO001B.so  DEMO006L.so  JCLCREAT.so  RGHTJUST.so  TEST002T.so
ACCT01.so  ACCT03.so  CDLIDEMO.so  DEMO001T.so  EXECDEMO.so  JCLREAD.so   TEST001T.so
```

CICS, JCL, IMS で使用するソースがコンパイルされ、実行可能ファイルが存在しています。

4) /home/esadm/mssdata/mod ディレクトリに MOD ファイルがあるか確認します。

```
[esadm@3b43668de7f8 loadlib64]$ ls /home/esadm/mssdata/mod
ACCTSET.MOD
```

画面定義である BMS ファイルがコンパイルされ、MOD ファイルが存在しています。

5) /home/esadm/mssdata ディレクトリには JES で使用するカタログファイル類が存在してます。

```
[esadm@3b43668de7f8 loadlib64]$ ls /home/esadm/mssdata
SPLDSN.dat  SPLJNO.dat  SPLJOB.dat  SPLMSG.dat  bms  catalog.dat  data  dbd  imsloadlib  mfs  mod  psb  system
```

6) 実行させる Enterprise Server インスタンスに関連する console.log などは /var/mfcobol/es/MSS64/ にあり、内容を表示すると、正常に MSS64 インスタンスが開始されたことが確認できます。

```
[esadm@3b43668de7f8 loadlib64]$ cat /var/mfcobol/es/MSS64/console.log
220713 07360484                  CASCD0100I ES Threaded Daemon Initialized (Ver CAS 8.0.00) process-id = 22 (07:36:04.52) 07:36:04
220713 07360484                  CASCD0099I ES Build Tag: ED8.0/20220509a_ED80 07:36:04
220713 07360484                  CASCD0028I Console Daemon running with effective user ID = 01010 07:36:04
220713 07360549                  CASCD0120I Server manager created for ES MSS64, process-id = 28 07:36:05
220713 07360559        28 MSS64  CASSI0000I Server manager initialization started 07:36:05
```

7) CICS で使用するリソース定義ファイルは /home/esadm/mssdata/system にあります。

```
[esadm@3b43668de7f8 loadlib64]$ ls /home/esadm/mssdata/system
dfhdrdat
```

8) 他のディレクトリ内も確認後、コンテナから抜けてホスト OS へ戻ります。

コマンド例）exit

```
[esadm@3b43668de7f8 loadlib64]$ exit
exit
#
```

10. Enterprise Server インスタンスの確認

Web ブラウザからコンテナ内で起動している Enterprise Server Common Web Administration (以降 ESCWA) を表示します。デモンストレーション環境には SHIFT_JIS ロケールが設定されていないため、実行するアプリケーションは英語表記となることをご了承ください。

1) ホストマシンの IP アドレスとコンテナイメージ生成時に指定している 30086 番ポートを指定して ESCWA にアクセスします。アクセスできない場合はホストマシンのファイアウォール設定をご確認ください。

例）http://192.168.33.44:30086

2) ESCWA は Micro Focus Directory Server（以降 MFDS）のポートへ接続して、登録されている Enterprise Server インスタンスを管理する画面です。初期表示時、コンテナ内の MFDS ポートが 86 に指定してある場合は 30090 へ変更して登録されているインスタンスが表示されることを確認してください。

Containerfile 内の記述）

```
ENV CCITCP2_PORT=30090
```

実行対象となる MSS64 インスタンスが開始状態であることが確認できます。

3) CICS 環境の設定値を確認します。

前述で確認したディレクトリに存在する実行可能ファイルや MOD ファイル、リソース定義ファイルを利用していることがわかります。

＄BASE は CICS ディレクトリの Containerfile に記述された環境変数を指しています。

Containerfile 内の記述）

```
ENV BASE=/home/esadm
```

4) JES 環境の設定値を確認します。

前述で確認したディレクトリに存在する実行可能ファイルやカタログファイルを利用していることがわかります。

5) IMS 環境の設定値を確認します。


```
esadm@3b43668de7f8 mssdata]$ cd imsloadlib
esadm@3b43668de7f8 imsloadlib]$ ls
DC7FPRINT2.DOF  427FTEST03.DOF  427FtEST03.DIF  DBDGEN2.DAT   DEMO91.MFSX   INTEST00.MID  OTDEMO92.MOD  TEST00.MFSX   TEST01DD.DAT
427FDEMO90.DOF  427FdEMO90.DIF  477FDEMO90.DOF  DBDGEN2F.DAT  DEMO92.MFSX   INTEST01.MID  OTTEST00.MOD  TEST001T.ACB  TEST02.MFSX
427FDEMO91.DOF  427FdEMO91.DIF  477FDEMO91.DOF  DEMO001T.ACB  DL1DEMO.MOD   INTEST02.MID  OTTEST01.MOD  TEST002T.ACB  TEST03.MFSX
427FDEMO92.DOF  427FdEMO92.DIF  477FDEMO92.DOF  DEMO03DD.ACB  IMSGEN2.DAT   INTEST03.MID  OTTEST02.MOD  TEST01.MFSX   TRANCODE.TXT
427FTEST00.DOF  427FtEST00.DIF  477FdEMO90.DIF  DEMO03DD.DAT  INDEMO90.MID  OPRINT92.MOD  OTTEST03.MOD  TEST012D.DAT
427FTEST01.DOF  427FtEST01.DIF  477FdEMO91.DIF  DEMO03DD.DBU  INDEMO91.MID  OTDEMO90.MOD  PRINT.MFSX    TEST013D.DAT
427FTEST02.DOF  427FtEST02.DIF  477FdEMO92.DIF  DEMO90.MFSX   INDEMO92.MID  OTDEMO91.MOD  PSBGEN3.DAT   TEST01DD.ACB
```

6) TN3270 エミュレータから接続する TN3270 リスナーと JCL のサブミットを受け付ける Web Services and J2EE リスナーのポート番号を確認します。

TN3270 リスナーポート：30040

Web Services and J2EE リスナーポート：30041

11. ホストマシン、コンテナ間のリソース共有

コンテナ技術を利用したホストマシンやコンテナ間のリソース共有の方法は複数ありますが、後述する CICS の実行ではマウントを使用せずコンテナを利用し、JCL, IMS の実行ではホストマシンのディレクトリを指定するバインドマウントを利用してリソースを共有します。コンテナコマンドに関してはコンテナ技術のマニュアルをご参照ください。

1) コンテナ名を指定してコンテナ内へ入ります。

コマンド例）

podman exec -it cicsdemo bash

2) コンテナ内にホストマシンからバインドマウントするディレクトリを作成します。

コマンド例）mkdir jclshare

作成したパス）/home/esadm/jclshare

```
esadm@3b43668de7f8 ~]$ pwd
/home/esadm
esadm@3b43668de7f8 ~]$ mkdir jclshare
esadm@3b43668de7f8 ~]$ ls
MSS  MSS.tar.gz  MSS64_combined.xml  MSS64_grps.log  MSS64_sit.log  MSS64_stul.log  bin  jclshare  mssdata
```

3) コンテナから抜けてホスト OS へ戻ります。

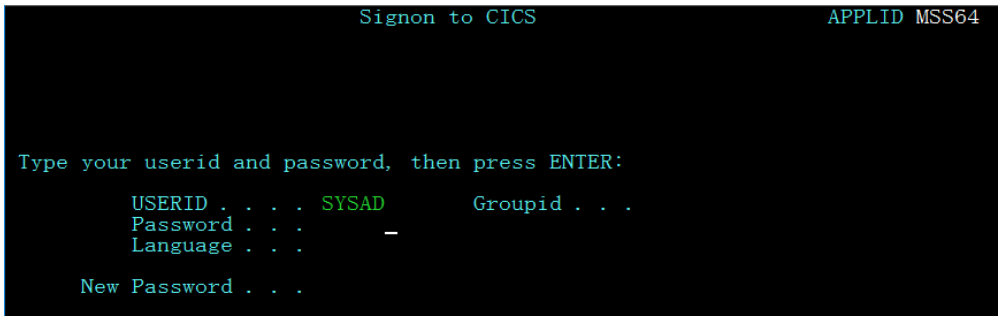
コマンド例）exit

12. CICS の実行

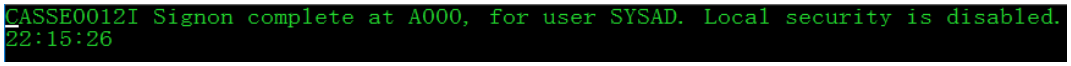
TN3270 エミュレータを使用して、前述で確認したリスナー番号へ接続します。

例）11.22.33.44:30040

1) CICS ログイン画面が表示されますので、USERID と Password に SYSAD を入力して Enter を押下します。

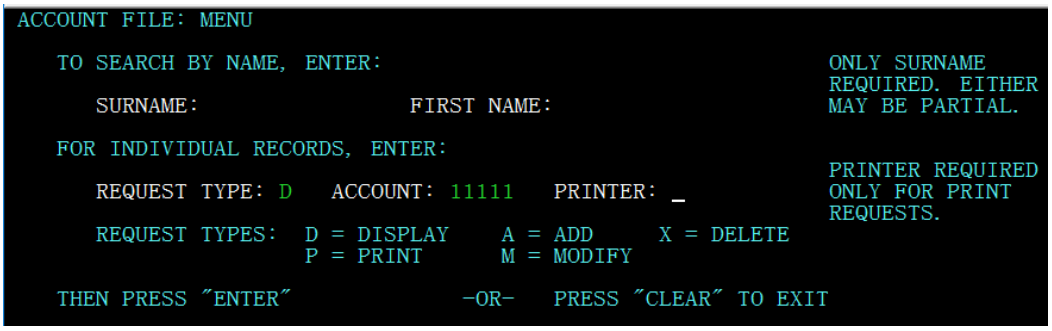


2) CICS へログインが成功したメッセージが表示されます。

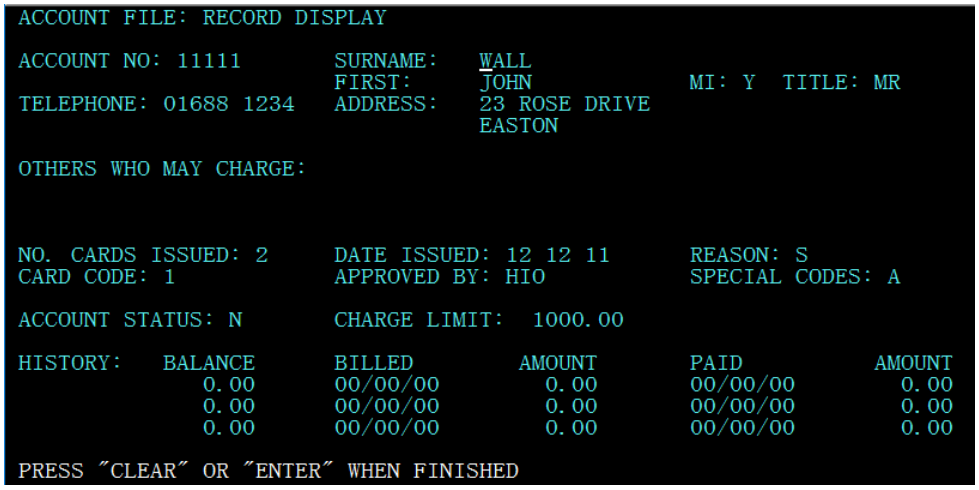


3) 画面をクリアして CICS トランザクションである ACCT を入力して Enter を押下します。

4) メニュー画面が表示されたら、REQUEST TYPE に D を、ACCOUNT に 11111 を入力して Enter を押下します。



5) データファイルを読み込み、11111 に該当するデータ内容が表示されます。



6） コンテナ内のポートへのアクセス確認が完了しましたので、エミュレータからの接続を切断します。

13.JCL の実行

コンテナ内から JCL をサブミットすることはできますが、ここではホストマシンのディレクトリとコンテナ内のディレクトリを共有するバインドマウントを利用してホストマシンから JCL を実行してみます。

1）ホストマシンにマウント対象のディレクトリを作成します。

コマンド例）mkdir shareJCL

パスの例）/home/tarot/mf/shareJCL

2）ホストマシンに展開した /home/tarot/mf/container/Examples/CICS/src/MSS から 2 つの JCL を作成したディレクトリへコピーします。

コマンド例）

cp *.jcl *.JCL /home/tarot/mf/shareJCL

3）コンテナを一旦停止します。

コマンド例）podman stop cicsdemo

4）コンテナを一旦除去します。

コマンド例）podman rm cicsdemo

5）ホストマシンの /home/tarot/mf/shareJCL をコンテナ内の /home/esadm/jclshare へマウントし、コンテナ名を jcldemo へ変更してコンテナを再度起動します。

コンテナ起動コマンド例） 1 行で入力してください。

podman run -d --name jcldemo -v/home/tarot/mf/shareJCL:/home/esadm/jclshare:z -p 30086:10086 -p 30040-30050:30040-30050 -d microfocus/ed-mss:rhel8.4_8.0_x64

6）コンテナへ入り、JCL が共有できていることを確認します。

コマンド例）podman exec -it jcldemo bash

確認コマンド）ls /home/esadm/jclshare

```
[esadm@f394f777a42c ~]$ ls /home/esadm/jclshare
ESJCL.jcl  IMSBATCH.JCL
```

7） コンテナから抜けてホスト OS へ戻ります。

コマンド例）exit

8) ホストマシンから、共有している JCL を確認した Web Services and J2EE リスナーポートに向けて実行します。

cassub -j/home/tarot/mf/shareJCL/ESJCL.jcl -stcp:<ホストマシンの IP アドレス>:30041

```
#cassub -j/home/tarot/mf/shareJCL/ESJCL.jcl -stcp:[redacted]1:30041
JCLCM0187I J0001000 JCLTEST  JOB  SUBMITTED (JOBNAME=JCLTEST,JOBNUM=0001000) 11:02:53
JCLCM0180I J0001000 JCLTEST  Job ready for execution. 11:02:53
Processed "/home/tarot/mf/shareJCL/ESJCL.jcl"
```

9) ESCWA の JES スプール機能から実行結果を確認します。

		名前	ジョブID	クラス	ユーザー	条件コード
☐						
☐		JCLTEST	J0001000	B	JESUSER	0000

ホストマシンからコンテナ内の Enterprise Server インスタンスに向けて実行した JCL が正常に実行されました。

各スプール内容も参照してみてください。

14. IMS の実行

このコンテナイメージには IMS 実行環境が構築されています。MFS 画面を利用したオンライン処理と、JCL を利用したバッチ処理を実行します。

1) CICS と同様に TN3270 リスナーポートへ接続すると、ログイン画面が表示されますので、USERID と Password に SYSAD を入力して Enter を押下します。

```
                          Signon to CICS                        APPLID MSS64




Type your userid and password, then press ENTER:

         USERID . . . . SYSAD       Groupid . . .
         Password . . .       _
         Language . . .

     New Password . . .
```

2) CICS へログインが成功したメッセージが表示されます。

```
CASSE0012I Signon complete at A000, for user SYSAD. Local security is disabled.
22:15:26
```

3) 画面をクリアして IMS に切り替えます。

コマンド) /IMS

```
/IMS_
```

4) IMS チュートリアルでも使用した MFDEMO トランザクションを実行すると、英語版の画面が表示されます。

コマンド）MFDEMO （最後に空白が 1 文字あります）

```
MFDEMO
OTDEMO91  Micro Focus International Ltd.  TABLE FILE MAINTENANCE

SELECT ONE OF THE FOLLOWING FUNCTION CODES:

<A>DD      - TABLE FILE                                    LTERM: SYSAD
<C>HANGE   - TABLE FILE                                  USER ID: SYSAD
<D>ELETE   - TABLE FILE                                GROUP ID: SYSAD
<I>NQUIRE  - TABLE FILE


<E>ND      - TRANSACTION CODE

FUNCTION CODE _
```

コンテナ内の IMS オンライン処理の正常実行が確認できました。エミュレータを切断します。

5) 次に、前項で設定したホストマシンとコンテナで共有している IMSBATCH.JCL の内容を確認してみます。

```
#cat IMSBATCH.JCL
//IMSBATCH JOB 'MICRO FOCUS',CLASS=A,MSGCLASS=A
//* BMP PROGRAM EXECUTION
//* Copyright (c) Micro Focus 2017. All rights reserved. The
//* software and information contained herein are proprietary to,
//* and comprise valuable trade secrets of, Micro Focus , which
//* intends to preserve as trade secrets such software and
//* information. This software is an unpublished copyright of
//* Micro Focus and may not be used, copied, transmitted, or
//* stored in any manner.  This software and information or any
//* other copies thereof may not be provided or otherwise made
//* available to any other person.
//*
//S01        EXEC PGM=DFSRRC00,REGION=4M,
//           PARM='BMP,DEMO001B,DEMO001T,,,,,,,,CDLI,,N,N'
//REPORT1    DD  SYSOUT=*
//BTSLST     DD  SYSOUT=*
//IMSERR     DD  SYSOUT=*
//IEFRDER    DD  DUMMY
//PRINTDD    DD  SYSOUT=*
//SYSOUT     DD  SYSOUT=*
//SYSPRINT   DD  SYSOUT=*
//*
```

6) IMSBATCH.JCL をコンテナ内で設定している Web Services and J2EE リスナーポートに向けて実行します。

cassub -j/home/tarot/mf/shareJCL/IMSBATCH.JCL -stcp:<ホストマシンの IP アドレス>:30041

```
#cassub -j/home/tarot/mf/shareJCL/IMSBATCH.JCL -stcp:192[redacted]:30041
JCLCM0187I J0001002 IMSBATCH JOB  SUBMITTED (JOBNAME=IMSBATCH,JOBNUM=0001002)
JCLCM0180I J0001002 IMSBATCH Job ready for execution. 17:13:54
Processed "/home/tarot/mf/shareJCL/IMSBATCH.JCL"
```

7) ESCWA の JES スプール機能から実行結果を確認します。

正常に実行されました。各スプール内容も参照してみてください。

15. コンテナの停止と削除

使用したコンテナの停止と削除を実施します。

1）稼働しているコンテナ名を指定して停止します。

コマンド例）podman stop jcldemo

2）停止したコンテナを除去します。

コマンド例）podman rm jcldemo

3）現在、実行中および停止中のコンテナを確認します。

コマンド例）podman ps –a

```
#podman ps -a
CONTAINER ID  IMAGE   COMMAND  CREATED  STATUS  PORTS   NAMES
#
```

生成したコンテナが除去され、ホストマシンに存在するコンテナはゼロになりました。

16. まとめ

ホストマシンのプロセスとして稼働するコンテナは、環境作成や稼働において非常に軽やかに実行できることをご体験いただけたと思います。本チュートリアルではデモンストレーションイメージを生成しましたが、コンテナ製品のベースイメージからご自身のコンテナイメージを生成し、ホストマシンのアプリケーションをコンテナ内にコピー、またはホストマシンのディレクトリをコンテナ内にマウントして使用することや、ロケールに SHIFT_JIS[1] を追加して日本語環境を構築することもできます。

また、Enterprise Developer の異なるバージョンでアプリケーションをテストしたい場合にも、各バージョンのイメージからご自身のコンテナイメージを生成し、現行稼働しているアプリケーションを連携させることで容易に環境を構築して実施することが可能になり、開発工数の削減も期待することができます。

Micro Focus Enterprise Developer は、長年蓄積されたビジネスロジックを持つ COBOL, PL/I アプリケーションを現代的な技術に適用させるシステム環境をご提供しています。例えば、COBOL アプリケーションをコンテナ内で稼働させ、他システムとの連携を行うなど、リホストに留まることなく将来を見据えた段階的なシステムの刷新計画においてお役立ていただければと存じます。

17. 免責事項

本チュートリアルの例題ソースコードは機能説明を目的としたサンプルであり、無謬性を保証するものではありません。例題ソースコードは弊社に断りなくご利用いただけますが、本チュートリアルに関わる全てを対象として、二次的著作物に引用する場合は著作権法の精神に基づき適切な扱いを行ってください。

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。

[1] OS のサポート情報をご確認ください