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

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 ： Amazon Linux release 2
- Enterprise Developer 10J 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
- 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

開発環境製品の Enterprise Developer と 実行環境製品の Enterprise Server は、下記の OS を対象とした 64 ビットのベースイメージをご提供しており、Docker や OCI 準拠コンテナツールの利用をサポートしています。

・Amazon Linux 2, Amazon Linux 2023
・Red Hat Enterprise Linux 8.2 以降
・SUSE Linux Enterprise Server 15 以降
・Rocky Linux 9

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)amzn2_10.0_x64
2） 対話型ログインのベースイメージ: タグ名の例)amzn2_10.0_x64 _login

本チュートリアルではデモイメージを生成する Dockerfile に記述されているベースイメージを使用します。Dockerfile 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)Amazon Linux2 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
amzn2_10.0_x64
例)Amazon Linux2 環境に コンテナ製品の PatchUpdate01 を展開したイメージタグ名
amzn2_10.0_x64_pu01

6. コンテナツールの準備

Amazon Linux2 に Docker をインストールします。
コマンド例)yum -y install docker

```
#yum -y install docker
```

バージョン確認)

```
#docker --version
Docker version 25.0.5, build 5dc9bcc
```

Docker デーモンを起動します。

コマンド例)systemctl start docker

```
#systemctl start docker
```

ブート時に自動起動するよう設定します。これは任意です。

コマンド例)systemctl enable docker

```
#systemctl enable docker
```

ステータスが稼働中であることを確認します。

コマンド例)systemctl status docker

```
#systemctl status docker
● docker.service - Docker Applicatio
   Loaded: loaded (/usr/lib/systemd/
   Active: active (running) since 火
     Docs: https://docs.docker.com
 Main PID: 17651 (dockerd)
```

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_10.0_amazon_x64.tar

```
#tar xvf entdev_dockerfiles_10.0_amazon_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
```

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)docker image ls

```
#docker 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
Dockerfile                                  bld.env.amzn  dotnet_install.sh
Enterprise-Developer-Linux-Docker-Named-User.xml  bld.funcs     license_install.sh
README-tree.md                              bld.pfuncs    post_setup.sh
README-tree.png                             bld.sh        prodver.env
README.html                                 bld.sh.env    setup_entdev_for_docker_10.0_amazon_x64.gz
README.txt                                  bld.variants
```

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.amzn
Using 'docker buildx'
./bld.sh: Sorry no license .xml file found.
 - Expected Enterprise-Developer-Linux-Docker-Named-User.xml or another .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 オプションは製品の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

REPOSITORY:TAG                              IMAGE ID       SIZE      CREATED
microfocus/entdevhub:amzn2_10.0_x64_login   0cb03ab0f652   1.75GB    Less than a second ago
microfocus/entdevhub:amzn2_10.0_x64         2b4c72ce29bd   1.73GB    24 seconds ago

 To use:
   docker run  --rm -ti microfocus/entdevhub:amzn2_10.0_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。

```
#docker image ls
REPOSITORY            TAG                      IMAGE ID       CREATED              SIZE
microfocus/entdevhub  amzn2_10.0_x64_login     0cb03ab0f652   45 seconds ago       1.75GB
microfocus/entdevhub  amzn2_10.0_x64           2b4c72ce29bd   About a minute ago   1.73GB
```

追加された 2 つのベースイメージタグ名)

amzn2_10.0_x64_login

amzn2_10.0_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展開したディレクトリに移動して 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。

```
#ls
Build_HelloWorld      Build_NET8_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が含まれる CICS ディレクトリへ移動します。

```
#cd CICS
#ls
Dockerfile          README.html  bld.env.amzn  bld.pfuncs  bld.sh.env      common_setup  prodver.env   src
MSS64_combined.xml  README.txt   bld.funcs     bld.sh      bld_package.sh  mssdata       sample_setup
```

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.amzn, bld.sh.env, prodver.env 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ファイルです。

④ bld.funcs, bld.pfuncs 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 Dockerfile ファイル
Docker イメージを作成するための手順を記したファイルです。

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.html, READNE.txt 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。

⑪ common_setup
デフォルトセキュリティに関する設定ファイルです。

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)docker ps -a

```
#docker 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

```
Image microfocus/ed-mss:amzn2_10.0_x64 created

Default User/Password   :
   User                 : SYSAD
   Password             : qBOjZSGj

   To use:
        docker run --name acct -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d microfocus/ed-mss:amzn2_10.
0_x64
        open http://localhost:7443
        open http://localhost:30086
        docker logs -f acct
        docker kill acct
        docker rm acct
```

後述するデフォルトセキュリティに対するデフォルトユーザーと初期パスワードが表示されますので、メモ帳などに記録してください。

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

```
#docker image ls
REPOSITORY              TAG                     IMAGE ID       CREATED         SIZE
microfocus/ed-mss       amzn2_10.0_x64          97a2f42f4bbd   2 hours ago     1.74GB
microfocus/entdevhub    amzn2_10.0_x64_login    0cb03ab0f652   22 hours ago    1.75GB
microfocus/entdevhub    amzn2_10.0_x64          2b4c72ce29bd   22 hours ago    1.73GB
```

7）生成時に表示されたメッセージを参考に docker 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 acct というコンテナ名(任意)を指定します。

コマンド例)

docker run --name acct -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d microfocus/ed-mss:amzn2_10.0_x64

```
#docker run --name acct -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d microfocus/ed-mss:amzn2_10.0_x64
c0c9aca6ff02eeda8ca09d3b29527f2bfaeb7006de9ed3115ef1dcb59a63f769
```

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起動した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)docker ps -a

```
#docker ps -a
CONTAINER ID   IMAGE

               NAMES
c0c9aca6ff02   microfocus/ed-mss:amzn2_10.0_x64
```

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)

docker exec -it acct bash

```
#docker exec -it acct bash
[esadm@c0c9aca6ff02 ~]$
```

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@c0c9aca6ff02 ~]$ cd MSS
[esadm@c0c9aca6ff02 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@c0c9aca6ff02 MSS]$ cd loadlib64
[esadm@c0c9aca6ff02 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@c0c9aca6ff02 MSS]$ cd /home/esadm/mssdata/mod
[esadm@c0c9aca6ff02 mod]$ ls
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配置されています。

```
[esadm@c0c9aca6ff02 mssdata]$ ls
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@c0c9aca6ff02 MSS64]$ ls
IMSDB.TLOG  IMSMESGQ.dat  MFCASCD.lck  TXTDNR  TXTSNR  cascd.lck  console.log  gcf.bin  log.html  stdout  tscLog00000001-00000002.log
IMSDBC.DAT  LOGTSTDQ.dat  MSS64.env    TXTDRC  TXTSRC  cascsf.dat  cwi-user-certs  log-1.html  rcf.bin  tscLog00000001-00000001.log
[esadm@c0c9aca6ff02 MSS64]$ cat console.log
240911 05163745                CASCD0100I ES Threaded Daemon Initialized (Ver CAS 9.0.00) process-id = 137 (05:16:37.39) 05:16:37
240911 05163745                CASCD0099I ES Build Tag: ED10.0/20240607 05:16:37
240911 05163745                CASCD0028I Console Daemon running with effective user ID = 01010 05:16:37
240911 05163839                CASCD0120I Server manager created for ES MSS64, process-id = 141 05:16:38
```

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@c0c9aca6ff02 MSS64]$ 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@c0c9aca6ff02 MSS64]$ 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。

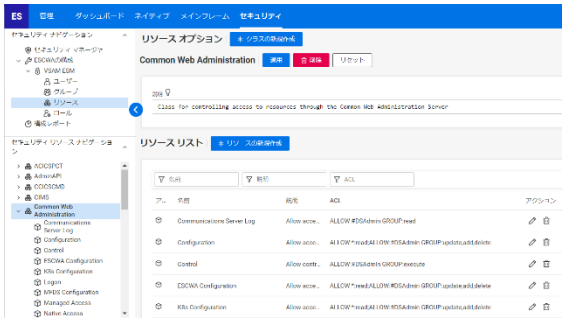
1） ホストマシンのホスト名または IP アドレスとコンテナイメージ生成時に指定している 30086 番ポートを指定して ESCWA にアクセスします。アクセスできない場合はホストマシンのファイアウォール設定をご確認ください。
例)http://XXX.XXX.XXX.XX:30086

デフォルトでは、製品が提供する VSAM 外部セキュリティマネージャー(ESM)モジュールによるセキュリティが有効になっており、すべての処理において実行ユーザーの認証が行われます。

次のコマンドを実行して、デフォルトユーザーと初期パスワードを取得します。
mfsecretsadmin read microfocus/temp/admin

コマンドによりこの設定を無効にすることもできます。

ユーザー権限によっては実行を拒否されることがあり、この場合は ESCWA にログオン後、ESCWA に設定されているセキュリティ設定を確認、または関連するリソースをメンテナンスしてください。



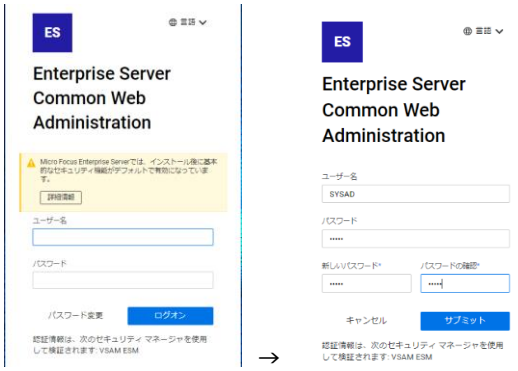
詳細は製品アニュアルをご参照ください。

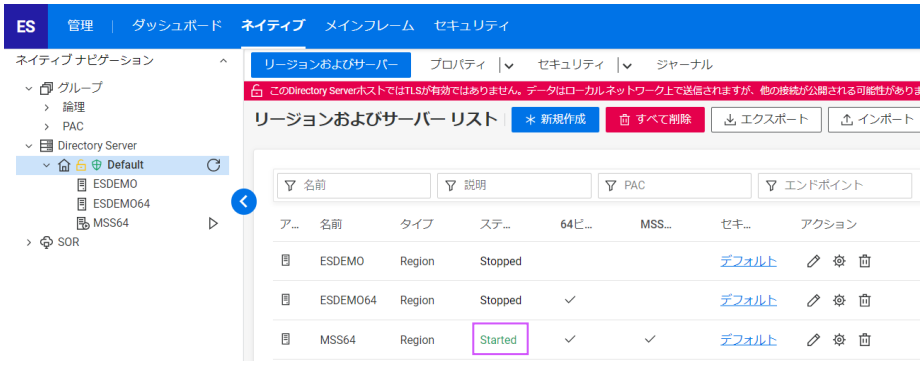
ブラウザが立ち上がり ESCWA が表示され、ユーザー認証を求められます。
セキュリティ確保のために初期パスワードを変更してログオンします。

初期パスワードに半角英小文字が含まれている場合には、CICS サインオン時に TN3270 エミュレータから入力する半角英大文字と、半角英小文字を含む初期パスワードが不一致となり、サインオンできません。
これを回避するために、新パスワードの英字はすべて大文字で入力してください。
良い新パスワードの例）SYSAD123
悪い新パスワードの例）sysad123

[パスワード変更] をクリックし、前項でメモしたデフォルトユーザーと初期パスワード、新しいパスワードを入力して [サブミット] をクリックします。

変更後のパスワードはご自身の責任で管理してください。
パスワード変更後、画面がタイムアウトした場合は新しいパスワードを使用してログオンしてください。





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。

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。

Dockerfile 内の記述）

```
ENV CCITCP2_PORT=30090
```

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 Dockerfile に記述された環境変数を指しています。

Dockerfile 内の記述）

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

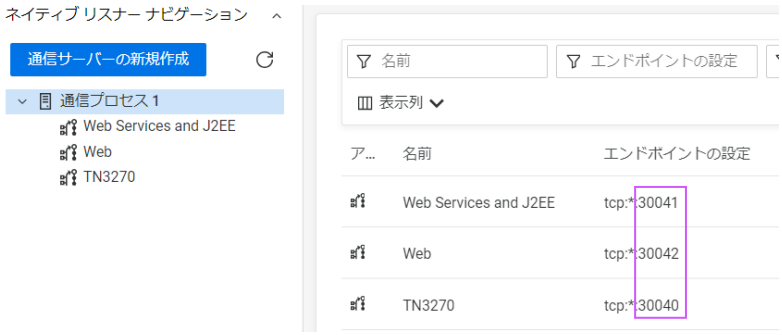
4）JES 環境の設定値を確認します。

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@c0c9aca6ff02 ~]$ ls mssdata/imsloadlib
0C7FPRINT2.DOF  427FTEST02.DOF  427FtEST01.DIF  477FdEM090.DIF  DEM003DD.ACB  DLIDEMO.MOD    INTEST01.MID  OTDEM092.MOD  PSBGEN3.DAT   TEST013D.DAT
427FDEM090.DOF  427FTEST03.DOF  427FtEST02.DIF  477FdEM091.DIF  DEM003DD.DAT  IMSGEN2.DAT    INTEST02.MID  OTTEST00.MOD  TEST00.MFSX   TEST01DD.ACB
427FDEM091.DOF  427FdEM090.DIF  427FtEST03.DIF  477FdEM092.DIF  DEM003DD.DBU  INDEM090.MID   INTEST03.MID  OTTEST01.MOD  TEST001T.ACB  TEST01DD.DAT
427FDEM092.DOF  427FdEM091.DIF  477FDEM090.DOF  DBDGEN2.DAT     DEM090.MFSX   INDEM091.MID   OPRINT92.MOD  OTTEST02.MOD  TEST002T.ACB  TEST02.MFSX
427FTEST00.DOF  427FdEM092.DIF  477FDEM091.DOF  DBDGEN2F.DAT    DEM091.MFSX   INDEM092.MID   OTDEM090.MOD  OTTEST03.MOD  TEST01.MFSX   TEST03.MFSX
427FTEST01.DOF  427FtEST00.DIF  477FDEM092.DOF  DEM0001T.ACB    DEM092.MFSX   INTEST00.MID   OTDEM091.MOD  PRINT.MFSX    TEST012D.DAT  TRANCODE.TXT
```

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）

docker exec -it acct 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@c0c9aca6ff02 ~]$ mkdir jclshare
[esadm@c0c9aca6ff02 ~]$ 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。

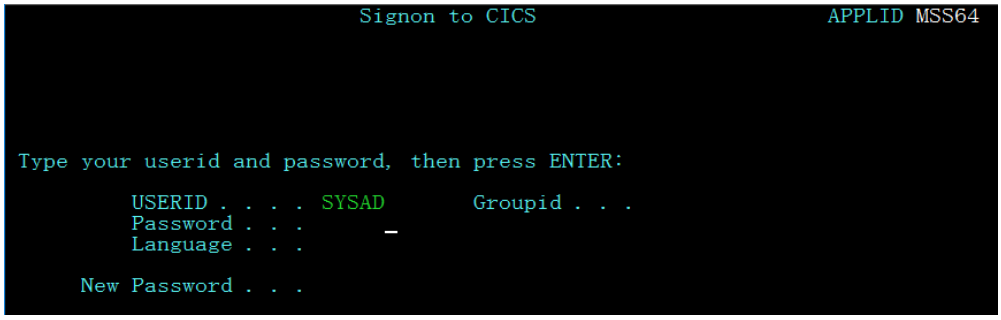
コマンド例）exit

12. CICS の実行

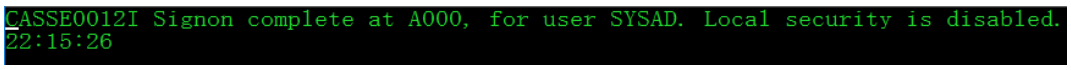
TN3270 エミュレータを使用して、前述で確認したリスナー番号へ接続します。

例)XXX.XXX.XXX.XXX:30040

1) CICS サインオン画面が表示されますので、[USERID]へ SYSAD を、[PASSWORD]へは、デフォルトセキュリティが有効な場合は前項で指定したパスワードを、無効の場合は SYSAD を入力して実行キーを押します。

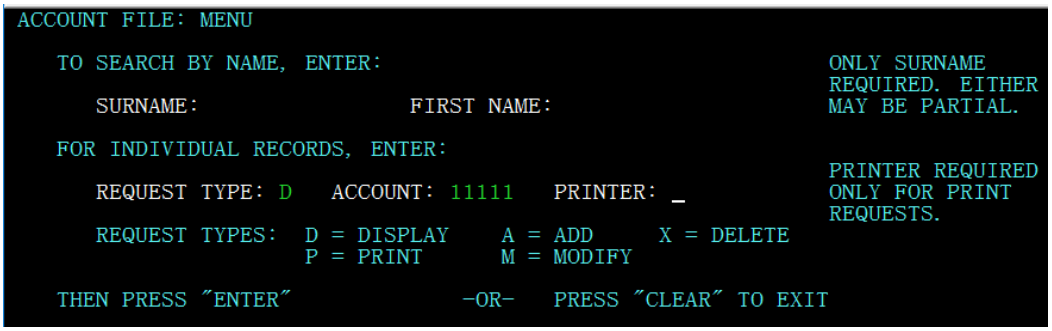


2) CICS へログインが成功したメッセージが表示されます。

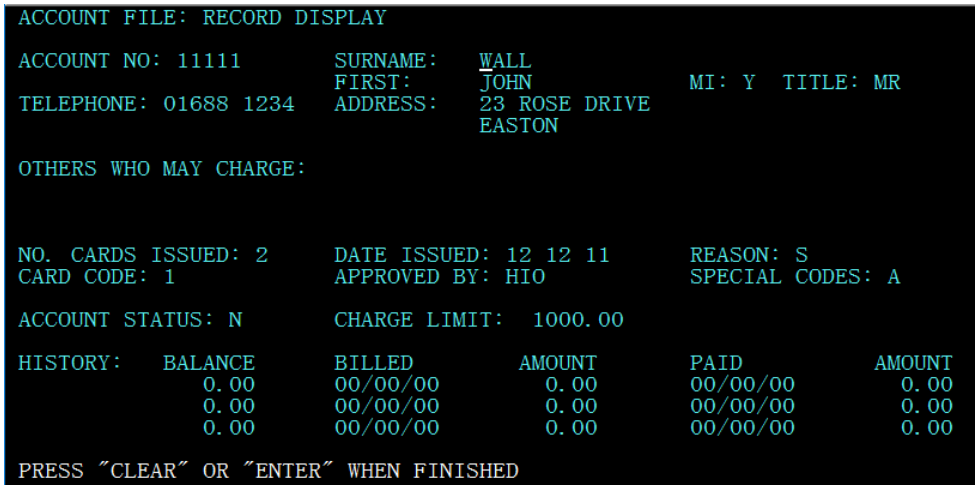


3) 画面をクリアして CICS トランザクションである ACCT を入力して Enter を押下します。

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。

7）コンテナを一旦停止します。
コマンド例)docker stop acct

8）コンテナを一旦除去します。
コマンド例)docker rm acct

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製品ベースの Examples ディレクトリ配下の 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）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 と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。
docker run --name jcldemo -v/home/tarot/mf/container/shareJCL:/home/esadm/jclshare:z -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d microfocus/ed-mss:amzn2_10.0_x64

4）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)docker exec -it jcldemo bash
確認コマンド)ls /home/esadm/jclshare

5）コンテナから抜けてホスト OS へ戻ります。
コマンド例)exit

6）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。

実行権限のある SYSAD ユーザーで実行しますが、acct コンテナは削除したため、指定するパスワードは初期パスワードもしくは acct 同様にログオン時に変更し、そのパスワードを指定します。本シュートリアルではパスワードを SYSAD に変更して実行しています。

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

```
#cassub -j/home/tarot/mf/shareJCL/ESJCL.jcl -stcp:192.168.      :30041 -uSYSAD -pSYSAD

JCLCM0187I J0001000 JCLTEST  JOB  SUBMITTED (JOBNAME=JCLTEST,JOBNUM=0001000) 00:39:33
JCLCM0180I J0001000 JCLTEST  Job ready for execution. 00:39:33
```

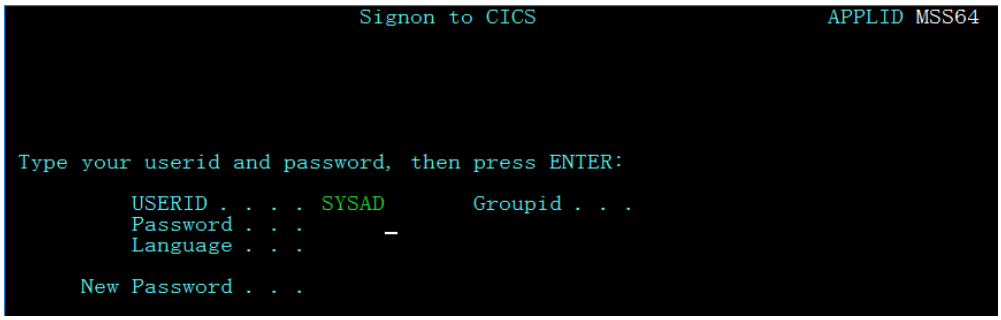
9）ESCWA の JES スプール機能から実行結果を確認します。

指定した SYSAD ユーザーで実行され、条件コードがゼロであることから、jcldemo コンテナに向けて実行した JCL が正常に終了したことがわかります。各スプール内容もご確認ください。

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稼働中の jcldemo コンテナを使用して実行します。

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。



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 を実行権限のある SYSAD ユーザーで実行します。前述の JCL と同様に、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 -uSYSAD -pSYSAD

```
#cassub -j/home/tarot/mf/shareJCL/IMSBATCH.JCL -stcp:192.168.      :30041 -uSYSAD -pSYSAD
```

7） ESCWA の JES スプール機能から実行結果を確認します。

指定した SYSAD ユーザーで実行され、条件コードがゼロであることから、正常に終了しています。

各スプール内容もご確認ください。

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）docker stop jcldemo

2） 停止したコンテナを除去します。

コマンド例）docker 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）docker ps -a

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

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。

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