Rocket software

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 ： Red Hat Enterprise Linux 9.4
- Enterprise Developer 11J 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、Podman や Docker、OCI 準拠コンテナツールの利用をサポートしています。Docker を使用する場合は、Podman コマンドを適時 Docker コマンドに読み替えてください。

詳しくは製品マニュアルをご確認ください。

・Amazon Linux 2, Amazon Linux 2023

・Red Hat Enterprise Linux 8.2 以降

・SUSE Enterprise 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)rhel9.4_11.0_x64_pu02

2） 対話型ログインのベースイメージ: タグ名の例)rhel9.4_11.0_x64_pu02_login

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 9 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

rhel9.4_11.0_x64

例)Red Hat Enterprise Linux 9 環境に コンテナ製品の PatchUpdate02 を展開したイメージタグ名

rhel9.4_11.0_x64_pu02

6. コンテナツールの準備

Red Hat Enterprise Linux に Podman をインストールします。

コマンド例)dnf install podman -y

```
#dnf install podman -y
```

バージョン確認コマンド例)podman version

```
#podman version
Client:       Podman Engine
Version:      4.9.4-rhel
API Version:  4.9.4-rhel
Go Version:   go1.21.13 (Red Hat 1.21.13-4.el9_4)
Built:        Fri Oct 11 18:02:32 2024
OS/Arch:      linux/amd64
```

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_11.0_redhat_x64_patchupdate02.tar

```
#tar xvf entdev_dockerfiles_11.0_redhat_x64_patchupdate02.tar
EntDev/.dockerignore
EntDev/Containerfile
```

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
```

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 images

```
#podman images
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.funcs   bld.sh.env          post_setup.sh
Enterprise-Developer-Linux-Docker-Named-User.xml  README.txt   bld.pfuncs  dotnet_install.sh   prodver.env
README-tree.png                               bld.env.rhel  bld.sh      license_install.sh  setup_user_rc.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
 NOTE: No Base Image Version Suffix set
 You can amend/override this by using the following environment variables:
  NO_PLATFORM_VERSION_ID_SUFFIX or PLATFORM_VERSION_ID_SUFFIX=number

./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 rocketsoftware/entdevhub images

REPOSITORY:TAG                                                IMAGE ID      SIZE        <no value>
localhost/rocketsoftware/entdevhub:rhel9.4_11.0_x64_pu02_login  f1d500c70602  1.74 GB     1 second ago
localhost/rocketsoftware/entdevhub:rhel9.4_11.0_x64_pu02        e5cc28a7d4b1  1.74 GB     50 seconds ago

 To use:
   podman run  --rm -ti rocketsoftware/entdevhub:rhel9.4_11.0_x64_pu02_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 images
REPOSITORY                                TAG                          IMAGE ID      CREATED             SIZE
localhost/rocketsoftware/entdevhub        rhel9.4_11.0_x64_pu02_login  f1d500c70602  59 seconds ago      1.74 GB
localhost/rocketsoftware/entdevhub        rhel9.4_11.0_x64_pu02        e5cc28a7d4b1  About a minute ago  1.74 GB
```

追加された2つのベースイメージタグ名）

rhel9.4_11.0_x64_pu02_login

rhel9.4_11.0_x64_pu02

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、内容を確認します。

```
#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、内容を確認します。

```
#cd CICS
#ls
Containerfile       README.txt    bld.pfuncs  bld_package.sh  prodver.env
MSS64_combined.xml  bld.env.rhel  bld.sh      common_setup    sample_setup
README.html         bld.funcs     bld.sh.env  mssdata         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、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 Containerfile ファイル
イメージを作成するために使用される Containerfile 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、README.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
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が存在するか確認します。

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

```
Image rocketsoftware/ed-mss:rhel9.4_11.0_x64_pu02 created

Default User/Password   :
   User                 : SYSAD
   Password             : GHivs5D5

   To use:
        podman run --name acct -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d rocketso
ftware/ed-mss:rhel9.4_11.0_x64_pu02
        xdg-open http://localhost:7443
        xdg-open http://localhost:30086
        podman logs -f acct
        podman kill acct
        podman 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、これらの値を記憶しておいてください。この例ではデフォルトユーザーは SYSAD、初期パスワードは GHivs5D5 です。

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

```
#podman images
REPOSITORY                                TAG                        IMAGE ID      CREATED             SIZE
localhost/rocketsoftware/ed-mss           rhel9.4_11.0_x64_pu02      00615bac7d63  About a minute ago  1.74 GB
localhost/rocketsoftware/entdevhub        rhel9.4_11.0_x64_pu02_login f1d500c70602  17 minutes ago      1.74 GB
localhost/rocketsoftware/entdevhub        rhel9.4_11.0_x64_pu02      e5cc28a7d4b1  18 minutes ago      1.74 GB
```

7） 生成時に表示されたメッセージを参考に、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)コマンドは1行です。

podman run --name acct -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d rocketsoftware/ed-mss:rhel9.4_11.0_x64_pu02

```
#podman run --name acct -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d rocketsoftware/ed-mss:rhel9.4_11.0
_x64_pu02
9e3cb85d292d315e58aefe79537a8b1c9b67b2175646b48c67826cb071c39e07
```

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

```
#podman ps -a
CONTAINER ID  IMAGE                                          COMMAND               CREATED         STATUS
      PORTS                                                                                  NAMES
e284e6b4b33a  localhost/rocketsoftware/ed-mss:rhel9.4_11.0_x64_pu02  /home/esadm/bin/s...  6 seconds ago  Up 6 seconds (unheal
thy)  0.0.0.0:7443->7443/tcp, 0.0.0.0:30040-30050->30040-30050/tcp, 0.0.0.0:30086->10086/tcp  acct
```

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 acct bash

```
#podman exec -it acct bash
[esadm@9e3cb85d292d ~]$
```

コンテナに 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@9e3cb85d292d ~]$ ls
MSS  MSS.tar.gz  MSS64_combined.xml  MSS64_grps.log  MSS64_sit.log  MSS64_stul.log  bin  mssdata
[esadm@9e3cb85d292d ~]$ cd MSS
[esadm@9e3cb85d292d 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@9e3cb85d292d MSS]$ cd loadlib64
[esadm@9e3cb85d292d loadlib64]$ ls
ACCTOO.so  ACCTO2.so  ACCTO4.so    DEMOOO1B.so  DEMOOO6L.so  JCLCREAT.so  RGHTJUST.so  TESTOO2T.so
ACCTO1.so  ACCTO3.so  CDLIDEMO.so  DEMOOO1T.so  EXECDEMO.so  JCLREAD.so   TESTOO1T.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@9e3cb85d292d 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配置されています。

```
[esadm@9e3cb85d292d 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@9e3cb85d292d loadlib64]$ cat /var/mfcobol/es/MSS64/console.log
260116 02063029                    CASCD0100I ES Threaded Daemon Initialized (Ver CAS 11.0.00) process-id = 123 (02:06:30.27
 02:06:30
260116 02063029                    CASCD0099I ES Build Tag: ED11.0/20251201_PU2 02:06:30
260116 02063029                    CASCD0028I Console Daemon running with effective user ID = 01010 02:06:30
260116 02063128                    CASCD0120I Server manager created for ES MSS64, process-id = 128 02:06:31
260116 02063131        128 MSS64   CASSI0000I Server manager initialization started 02:06:31
260116 02063132        128 MSS64   CASSI4005I Retrieving ES configuration from MFDS (127.0.0.1:30090) 02:06:31
260116 02063138        128 MSS64   CASSI0011I External security manager configured, external shutdown will require sign on 02
:06:31
```

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@9e3cb85d292d 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@9e3cb85d292d 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 と称す）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.XXX: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、ESCWA 表示時においても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、前項でコンテナ生成時に取得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。

上部のメニューから［オペレーション］を選択し、左側ペインで［Directory Server]>[Default］を選択すると、現在稼働中のMSS64インスタンスが表示されます。

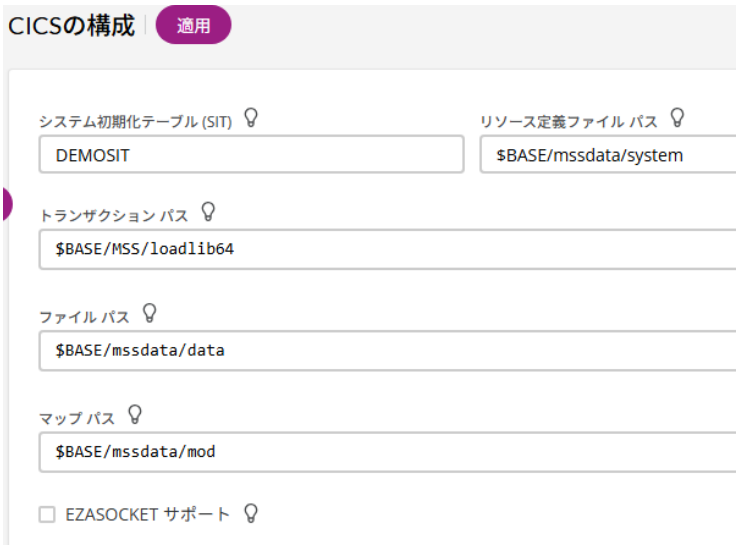
2） ESCWA は Directory Server(以降 MFDS)のポートへ接続して、登録されている Enterprise Server インスタンスを管理する画面です。MFDS 表示時、コンテナ内の MFDS ポートが 30090 であることを確認してください。

Containerfile 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 環境変数は CICS ディレクトリの Containerfile に記述されているパスを指します。

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@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  DEM001T.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一般的な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 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@9e3cb85d292d ~]$ mkdir jclshare
[esadm@9e3cb85d292d ~]$ 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。

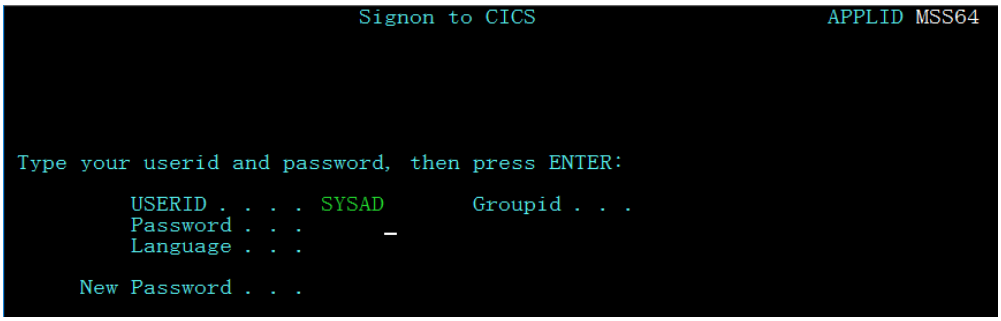
コマンド例）exit

12. CICS の実行

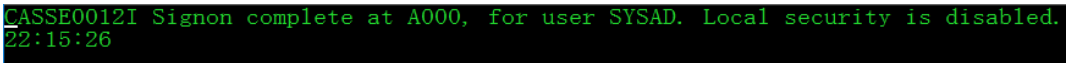
TN3270 エミュレータを使用して、前述で確認したリスナー番号へ接続します。

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]へは前項で指定したパスワードを入力して実行キーを押します。

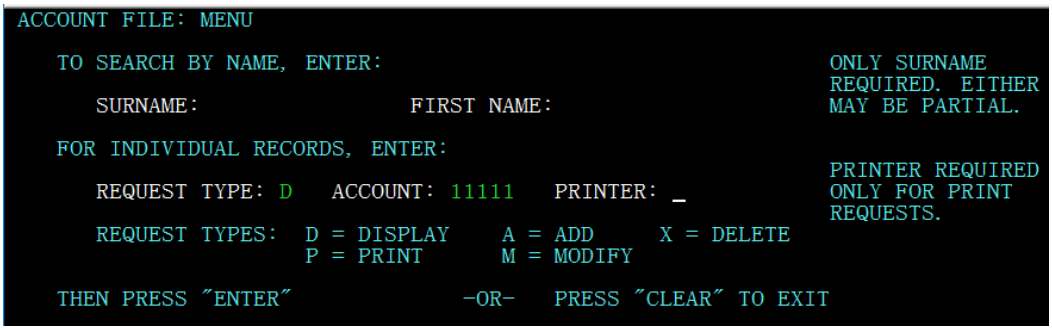


2） CICS へログインが成功したメッセージが表示されます。

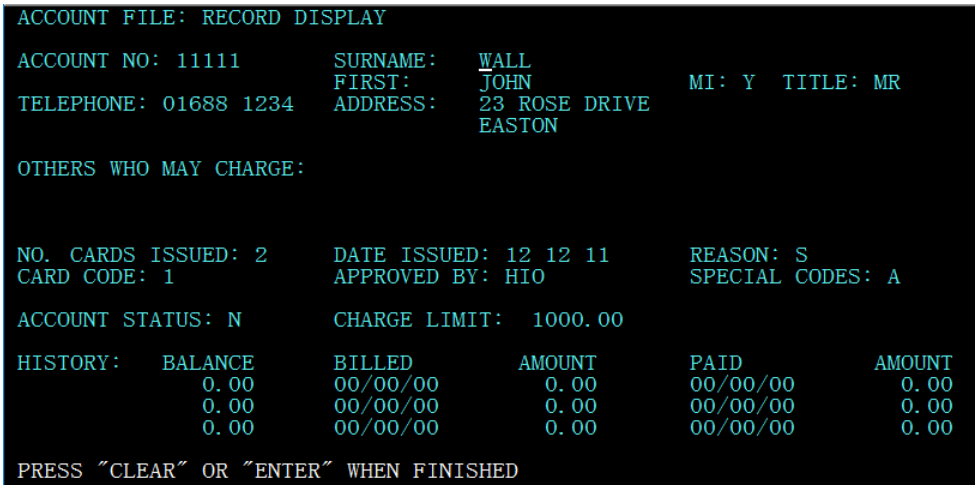


3） 画面をクリアして CICS トランザクションである ACCT を入力して Enter を押下します。

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） コンテナを停止します。
コマンド例)podman stop acct

8） コンテナを除去します。
コマンド例)podman 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/container11/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/container11/shareJCL

3） ホストマシンの/home/tarot/mf/container11/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。
podman run --name jcldemo -v/home/tarot/mf/container11/shareJCL:/home/esadm/jclshare:z -p 30086:10086 -p 30040-30050:30040-30050 -p 7443:7443 -d rocketsoftware/ed-mss:rhel9.4_11.0_x64_pu02

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

```
[esadm@fcea1d20cfc7 ~]$ ls /home/esadm/jclshare
ESJCL.jcl  IMSBATCH.JCL
```

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

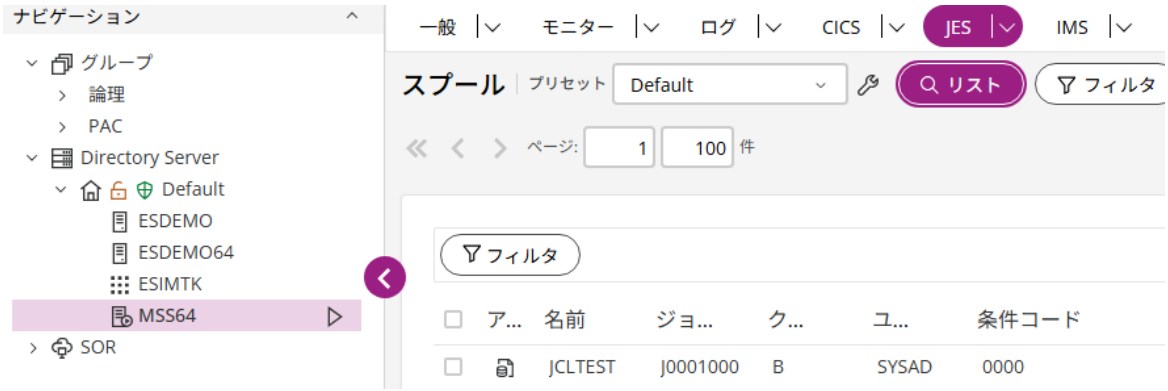
6） ホストマシン存在する、コンテナと共有済の JCL を Web Services and J2EE リスナーポートに向けて実行します。実行権限のある SYSAD ユーザーで実行しますが、acct コンテナは削除したため、指定するパスワードは初期パスワードもしくは acct 同様に ESCWA ログオン時に変更し、そのパスワードを指定します。本シュートリアルではパスワードを SYSAD に変更して実行しています。

コマンド例)1行で入力してください。

cassub -j/home/tarot/mf/container11/shareJCL/ESJCL.jcl -stcp:<ホストマシンの IP アドレス>:30041 -uSYSAD -pSYSAD

```
#cassub -j/home/tarot/mf/container11/shareJCL/ESJCL.jcl -stcp:192.          :30041 -uSYSAD -pSYSAD
JCLCM0187I JOB01000 JCLTEST  JOB  SUBMITTED (JOBNAME=JCLTEST,JOBNUM=0001000) 06:15:06
JCLCM0180I JOB01000 JCLTEST  Job ready for execution. 06:15:06
Processed "/home/tarot/mf/container11/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。



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 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) 1984-2025 Rocket Software, Inc.
//* or its affiliates.  All Rights Reserved
//*
//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=*
//*
```

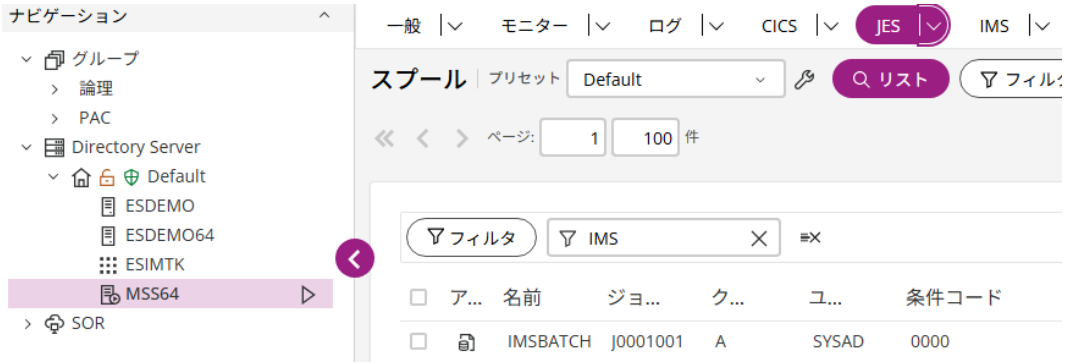
6） IMSBATCH.JCL を実行権限のある SYSAD ユーザーで実行します。前述の JCL と同様に、コンテナ内で設定している Web Services and J2EE リスナーポートに向けて実行します。

コマンド例)1行で入力してください。

cassub -j/home/tarot/mf/container11/shareJCL/IMSBATCH.JCL -stcp:<ホストマシンの IP アドレス>:30041 -uSYSAD -pSYSAD

```
#cassub -j/home/tarot/mf/container11/shareJCL/IMSBATCH.JCL -stcp:192.[redacted]:30041 -uSYSAD -pSYSAD
JCLCM0187I J0001001 IMSBATCH JOB  SUBMITTED (JOBNAME=IMSBATCH,JOBNUM=0001001) 06:26:54
JCLCM0180I J0001001 IMSBATCH Job ready for execution. 06:26:54
Processed "/home/tarot/mf/container11/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。



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)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)docker ps -a

```
#podman 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、リプラットフォームに留まることなく、将来を見据えた段階的なシステムのモダナイズ計画において、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