White Paper
Micro Focus Visual COBOL

コンテナ環境下で COBOL アプリケーションを実行する

企業の財産であるレガシーアプリケーションは、長年蓄積された業務要件や仕様を反映し、現在もビジネスの重要な役割を担って稼働を続けています。一方、アプリケーションを稼働させるプラットフォームは時代と共に変化しており、現在はクラウド環境の利用が促進されつつあります。それと同時に Docker、Podman といったコンテナ化技術の利用が多くなってきました。既存の COBOL アプリケーションもバージョンアップのタイミングでコンテナ下にて実行したいというニーズが増えてきています。
マイクロフォーカス では DX（デジタルトランスフォーメーション）に速やかに対応するソリューションとしてコンテナ対応する製品をリリースしています。それらを利用することで COBOL アプリケーションを他の言語で作り替える手間も省け、ビジネスロジックを内包する COBOL コンポーネントを触らずにマイクロサービス化して、Java や.NET といった他言語から利用できる変化に強いシステムを構築することができます。

本ドキュメントではマイクロフォーカスが提供しているコンテナ対応製品の紹介および利用方法を説明いたします。

White Paper
Micro Focus Visual COBOL

目次

1. コンテナ利用のメリット

1) 仮想環境とコンテナ環境の違い

コンテナを利用した開発と実行方法についてご説明いたしますので図 1 をご覧ください。
一般的な仮想化技術を用いて複数環境を構築した場合、ハイパーバイザ上の各仮想ハードウェア環境にそれぞれ OS がインストールされるので、起動を必要とするばかりでなく必要なハードウェアリソースが多くなります。

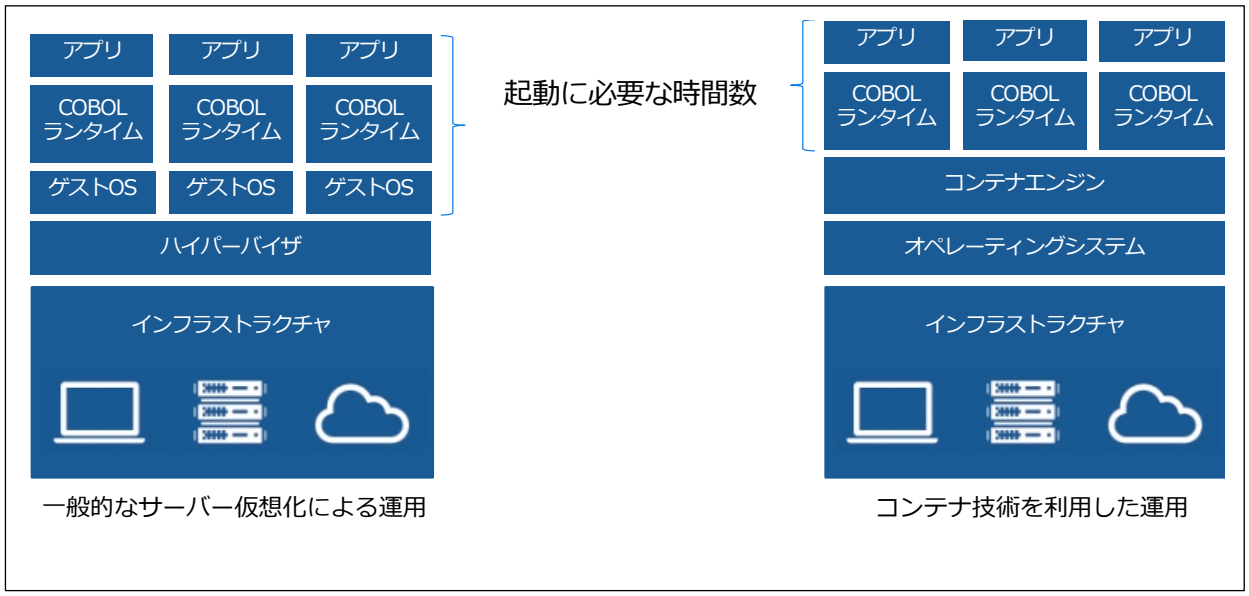


図 1

一方、Docker や Podman などのコンテナ技術を使用した場合、OS の上位にコンテナエンジンが入り、そこで分離されたアプリケーションがそれぞれ実行されるので CPU やメモリーなどのリソースが有効活用でき、簡単に環境の複製や削除をおこなうことができます。

これによって仮想環境とは違う下記のようなメリットがあります。

・処理が早くて軽量
・環境構築に要する時間を大幅に削減
・開発からリリースまでのサイクルを高速化

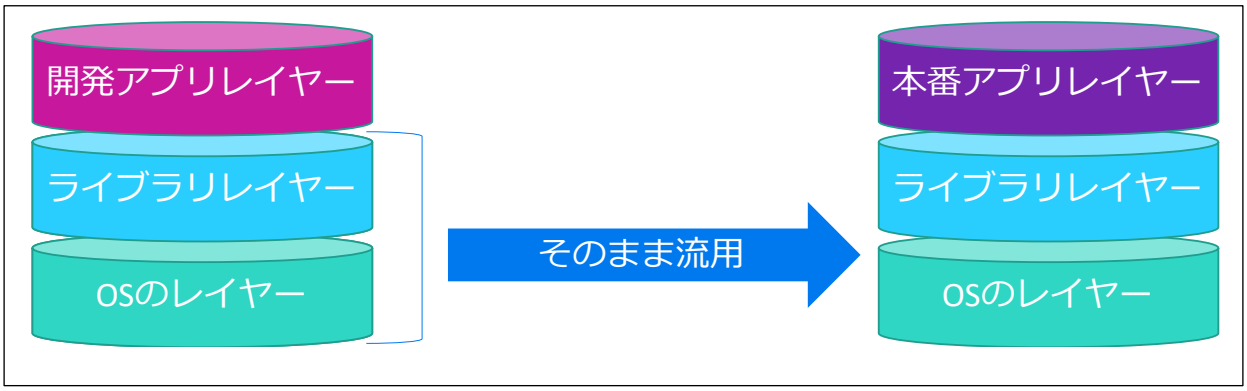
2) コンテナ環境を使った実行と開発

コンテナを利用する一番のメリットはレイヤーと呼ばれる層の中にプラットフォームの情報を隠蔽できるということではないでしょうか。

一般的な開発はプログラマー自身の PC 環境を使って OS や開発ツールのライブラリーを用意するので開発環境では動作するものの、結合テストのフェーズで用意された実行環境ではライブラリーの不足や、バージョンの異なるライブラリーがあるなどの要因から、想定通りにアプリケーションが動作せず、トラブルが収束しない場面をよく見かけます。

コンテナ環境では OS、ライブラリーのレイヤーを別々に用意し開発や実行を行うことで、このような環境に起因するトラブルを防止することができます。また、環境の準備が簡単で早いなどのメリットが出てきます。図 2 はそれらメリットを表したものです。

図 2



3) スケーラビリティのメリット

仮想環境でリソース不足に陥った時のシステム拡張や可用性を求めたクラスタリングを行うことは非常にコストのかかる作業です。この問題もコンテナに加え Kubernetes や RedHat OpenShift といったオーケストレーションソフトを利用することで負荷分散、スケールアウトを自動的に行えるので運用中の性能劣化を抑制することができます。

これらのメリットを享受できるマイクロフォーカスが提供するコンテナソリューションについて、次項からご説明いたします。

2. マイクロフォーカス のコンテナ対応ソリューション

1) 対応プラットフォーム

- Red Hat Enterprise Linux 7.4 以降
- SUSE Linux Enterprise Server 12 SP3 以降

2) 対応コンテナ製品

- Docker Community Edition をサポート
- RHEL 8.x では OCI コンテナ Podman をサポート

3) 開発環境製品

開発環境は、Red Hat Enterprise 7.4 以降では Docker コンテナに対応したインストーラー、Red Hat Enterprise Linux 8.x 以降には Podman に対応したインストーラーが提供されています。これらを通称 Visual COBOL Development Hub for Docker と呼んでいます。通常の Visual COBOL Development Hub では Windows 上の Eclipse IDE を利用してリモート開発が行えます。一方、このコンテナ対応製品では Eclipse IDE を利用したリモート開発機能は提供されていません。つまり、この製品はコマンドラインツールとしての機能のみを提供しています。本製品を利用することで、CI ツールと連携した DevOps プロセスといった開発と運用のサイクルを高速に回すための環境を構築することができます。詳細のインストール手順は製品マニュアルをご覧ください。

イメージ例：Red Hat 7
microfocus/vcdevhub:rhel7_6.0_x64

イメージ例：Red Hat 8.2
microfocus/vcdevhub:rhel8.2_6.0_x64

コンテナ内でコマンドラインを利用した開発を行うには下記を実行します。
docker run -ti microfocus/vcdevhub:rhel7_6.0_x64_login

podman run --rm -ti microfocus/vcdevhub:rhel8.2_6.0_x64_login

4) 実行環境製品

実行環境にはコンテナに対応した COBOL Server for Docker が提供されています。通常の Visual COBOL Development Hub もしくは Docker 用の Development Hub どちらの製品を利用して作成したバイナリーでも実行環境コンテナ内で実行できます。また、JVM COBOL のバイナリーを実行するためのイメージも用意されています。

イメージ例：Red Hat 7 の場合

microfocus/cobolserver	rhel7_6.0_x64
microfocus/cobolserver	rhel7_6.0_x64_java

イメージ例：Red Hat 8.2 の場合

microfocus/cobolserver	rhel8.2_6.0_x64
microfocus/cobolserver	rhel8.2_6.0_x64_java

3. コンテナ対応製品を利用した開発スタイル

コンテナ環境で実行されるアプリケーションの開発スタイルは、必ずしも従来型のスタイルを変える必要はありません。様々な開発スタイルが選択可能であり、以下のそのいくつかを説明します。

1) クロス開発

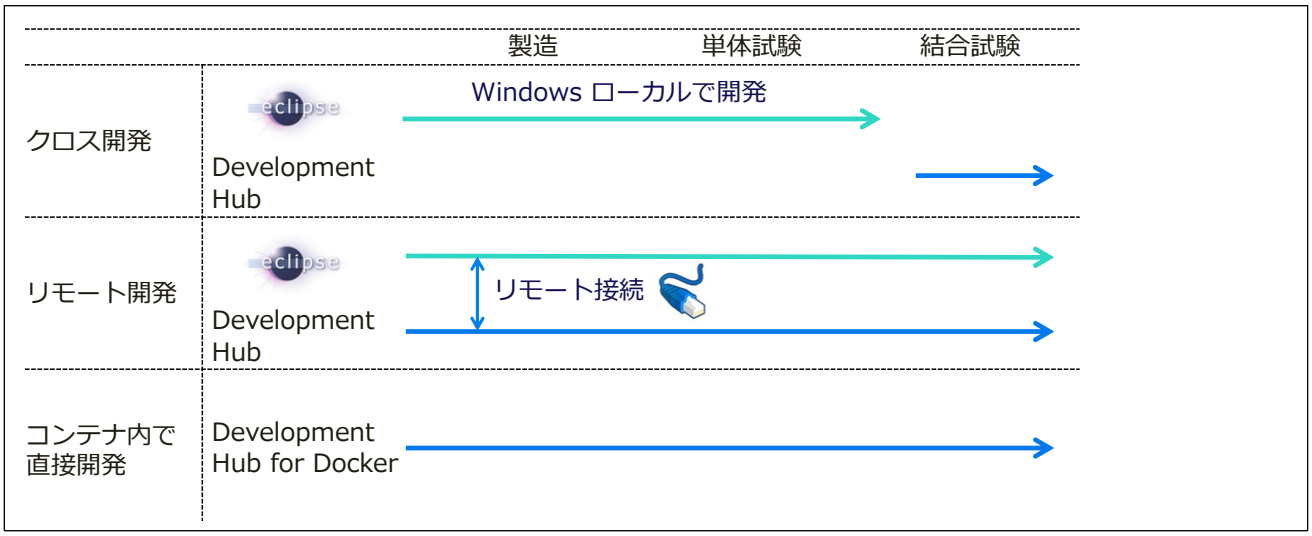
開発生産性に優れた Windows 環境の Visual COBOL for Eclipse 上で開発を行います。その後、Linux 環境にインストールされた Visual COBOL Development Hub が入っている環境にソースコードを転送してコンパイルを行い実行形式のバイナリーを作成します。作成したバイナリーは、COBOL Server for Docker 上で実行します。

2) リモート開発

Windows 環境の Visual COBOL for Eclipse と Linux 環境にて Visual COBOL Development Hub を使用します。両者の間を RSE という Eclipse のプラグインを使用して接続を行います。通常の Eclipse の操作を維持しながら、ファイルは Linux にあり、コンパイルも Linux 上で直接行われるので、クロス開発のようにファイルを転送する手間が省けます。これは通常のコンテナを使用しない COBOL アプリケーションの開発スタイルです。作成したバイナリーは、COBOL Server for Docker 上で実行します。

3) コンテナ内で直接開発

このスタイルは、開発作業を 1)や 2)のスタイルで開発を行っていることが前提となります。コンテナ環境内はコマンドラインベースの作業が可能ですが、生産性の面では効率が悪くお勧めしません。よって、コンテナ内でビルドを行う場合は、Development Hub for Docker を Jenkins などの CI ツールと連動させて利用することが多くなると思われます。この場合、GitHub からソースコードをチェックアウトしてビルドやテストを担当するなどの方法が最適でしょう。



4. インストールから製品付属サンプルプログラムの稼働まで

1) Visual COBOL 6.0 Development Hub for Docker のインストール

■Red Hat Linux 8.2 上にインストールファイル、ライセンスファイル(mflic)を用意

```
$ ls
devhub_dockerfiles_6.0_redhat8_x64.tar
Visual-COBOL-SOA-Dev-Hub-Docker.mflic
```

■インストールファイルを解凍

```
$ tar xvf devhub_dockerfiles_6.0_redhat8_x64.tar
$ ls
DevHub Examples README.html README.txt bld.env.rhel
Visual-COBOL-SOA-Dev-Hub-Docker.mflic
```

■ライセンスファイルをインストールディレクトリに移動

```
$ cd DevHub
$ mv ../Visual-COBOL-SOA-Dev-Hub-Docker.mflic .
$
```

■root 権限のユーザーでビルドスクリプトの実行

```
# ./bld.sh lacceptEULA
#
```

インストール完了後、以下のメッセージが表示されます。

```
Completed - we have the following microfocus/vcdevhub images
localhost/microfocus/vcdevhub   rhel8.2_6.0_x64_login   04d5a34c9e4d   3 seconds ago   999 MB
localhost/microfocus/vcdevhub   rhel8.2_6.0_x64         b90ed94022fe   3 seconds ago   999 MB

 To use:
    podman run --rm -ti microfocus/vcdevhub:rhel8.2_6.0_x64_login
```

■パッチが提供されている場合、パッチも適用（例: Visual COBOL 6.0 Patch Update 3 の場合）

```
$ ls
devhub_dockerfiles_6.0_redhat8_x64_patchupdate03.tar
```

■パッチアップデートファイルを解凍

```
$ tar xvf devhub_dockerfiles_6.0_redhat8_x64_patchupdate03.tar
```

■root 権限のユーザーでパッチアップデートの実行

```
# ./bld_docker_applypatch.sh
  setup_visualcobol_devhub_6.0_patchupdate03_266370_redhat_x86_64.gz lacceptEULA
```

行は続いています

インストール完了後、以下のメッセージが表示されます。

```
Completed - we have the following microfocus/vcdevhub images
localhost/microfocus/vcdevhub     rhel8.2_6.0_x64_pu03_login 82a1c69cf284 3 seconds ago    1.01 GB
localhost/microfocus/vcdevhub     rhel8.2_6.0_x64_pu03    072b6f9c73ea    About a minute ago 1.01 GB
localhost/microfocus/vcdevhub    rhel8.2_6.0_x64_login    04d5a34c9e4d     1 hours ago       999 MB
localhost/microfocus/vcdevhub    rhel8.2_6.0_x64             b90ed94022fe     1 hours ago       999 MB


 To use:
    podman run --rm -ti microfocus/vcdevhub:rhel8.2_6.0_x64_pu03_login
```

パッチアップデート後も既存イメージは残ります。

2) サンプルプログラムのコンパイル

■サンプルプログラムが入っているディレクトリに移動

```
$ cd Examples¥Build_HelloWorld
```

■Patch Update 3 のベースイメージを利用するように Containerfile を修正

```
ARG BASEIMAGE=microfocus/vcdevhub:rhel8.2_6.0_x64_PU03
```

■サンプルプログラムのビルド

```
$ ./bld.sh


          ～　途中省略　～
Image microfocus/vcdevhub -helloworld:rhel8.2_6.0_x64 created
To use:
   microfocus/vcdevhub-helloworld:rhel8.2_6.0_x64
```

■開発環境上でサンプルプログラムの実行

```
$ podman run --rm microfocus/vvcdevhub-helloworld:rhel8.2_6.0_x64
Hello from Docker!

This message shows that your installation appears to be working correctly.
You have created your first COBOL docker image and run it!

You have chosen to execute this example on the following:
 Platform     : Linux/Unix
 Executable : Native
 64bit          : yes
 Debug          : no
```

3) COBOL Server for Docker のインストール

■Red Hat Linux 8.2 上にインストールファイル、ライセンスファイル(mflic)を用意

```
$ ls
cobol_server_dockerfiles_6.0_redhat8_x64.tar
COBOL-Server-SOA-Docker.mflic
```

■インストールファイルを解凍

```
$ tar xvf cobol_server_dockerfiles_6.0_redhat8_x64.tar

$ ls
CobolServer Examples README.html README.txt bld.env.rhel
```

■ライセンスファイルをインストールディレクトリに移動

```
$ cd CobolServer
$ mv ../COBOL-Server-SOA-Docker.mflic .
$
```

■root 権限のユーザーでビルドスクリプトの実行

```
# ./bld.sh lacceptEULA
#
```

```
Image microfocus/cobolserver:rhel8.2_6.0_x64 created
Completed - we have the following microfocus/cobolserver images
localhost/microfocus/cobolserver        rhel8.2_6.0_x64     414941092642     1 hours ago     513 MB
```

■パッチが提供されている場合、パッチも適用（例: COBOL Server 6.0 Patch Update 3 の場合）

```
$ ls
cobol_server_dockerfiles_6.0_redhat8_x64_patchupdate03.tar
```

■パッチアップデートファイルを解凍

```
$ tar xvf cobol_server_dockerfiles_6.0_redhat8_x64_patchupdate03.tar
```

■root 権限のユーザーでパッチアップデートの実行

```
# ./bld_docker_applypatch.sh
  setup_cobol_server_for_docker_6.0_patchupdate03_266370_redhat_x64.gz lacceptEULA
```

行は続いています

インストール完了後、以下のメッセージが表示されます。

```
Completed - we have the following microfocus/cobolserver images
  localhost/microfocus/cobolserver   rhel8.2_6.0_x64_pu03    035ab4f6d175   1hours ago    544 MB
  localhost/microfocus/cobolserve    rhel8.2_6.0_x64          414941092642   6 hours ago     513 MB
```

パッチアップデートは別イメージとして作成されます。

4) サンプルプログラムのコンパイル

■サンプルプログラムが入っているディレクトリに移動

```
$ cd Examples¥HelloWorld
```

■Patch Update 3 のベースイメージを利用するように Containerfile を修正

```
ARG BASEIMAGE=microfocus/cobolserver:rhel8.2_6.0_x64_PU03
```

■サンプルプログラムのビルド

```
$ ./bld.sh

        ～　途中省略　～

Image microfocus/cs-helloworld:rhel8.2_6.0_x64 created
To use:
   podman run --rm microfocus/cs-helloworld:rhel8.2_6.0_x64
```

■実行環境上でサンプルプログラムの実行

```
$ podman run --rm microfocus/cs-helloworld:rhel8.2_6.0_x64
Hello from Docker!

This message shows that your installation appears to be working correctly.
You have created your first COBOL docker image and run it!

You have chosen to execute this example on the following:
 Platform     : Linux/Unix
 Executable : Native
 64bit          : yes
 Debug          : no
```

5. おわりに

DX（デジタルトランスフォーメーション）の促進に伴い、コンテナ化のメリットである開発環境と実行環境をパッケージングするニーズは今後増加することが予想されます。そして、コンテナにおけるアプリケーション開発も、.NET や Java 以外の選択肢として COBOL が利用できることは、企業の財産であるレガシーアプリケーションを利用したモダナイゼーションを検討する上で非常に重要なことです。

マイクロフォーカス は常に COBOL の資産価値を高める機能拡張を行っています。コンテナ対応に関しても今後より利便性や開発生産性を意識した製品拡張が行われていくでしょう。

マイクロフォーカス合同会社
https://www.microfocus.co.jp/