FUJITSU

FUJITSU Software
Interstage Application Server V12

Red Hat OpenShift上での動作手順書(J2EE編)

Linux(64)

8521-A-17-002
2017年9月

はじめに

本資料は、Interstage Application ServerのJ2EE機能をRed Hat OpenShift V3上で動かす手順について説明したものです。

なお本資料の中では、ソフトウェアの名称を、以下のように省略して表記します。

ソフトウェア名称	略称
Interstage Application Server	Interstage
Red Hat Enterprise Linux	RHEL
Red Hat Enterprise Linux 7(.x)	RHEL7(.x)(注)
Red Hat Open Shift	OpenShift

(注) マイナーバージョンを含んで書く場合があります。

前提知識

本書を読む場合、以下の知識が必要です。

- Red Hat Enterprise Linuxに関する基本的な知識
- Dockerに関する基本的な知識
- Red Hat OpenShiftに関する基本的な知識
- Interstage Application Serverに関する基本的な知識
-

商標

- Linux(R) は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。
- Red Hat(R)、Red Hat Enterprise Linux(R)、OpenShift(R)は米国およびその他の国において登録されたRed Hat, Inc.の商標です。
- OracleとJavaは、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- その他の記載されている商標および登録商標については、一般に各社の商標または登録商標です。

著作権

Copyright 2017 FUJITSU LIMITED

2017年9月 初版

目　次

第1章 動作環境 1
1.1 対象製品 1
1.2 準備するもの 1
1.3 本手順書の記載範囲 1

第2章 概要 2

第3章 Dockerイメージの作成 3
3.1 J2EEのベアイメージの作成 3
3.1.1 Interstageのサーバパッケージ DVDのマウント 3
3.1.2 インストールパラメーターCSVファイルを作成する 3
3.1.3 ホスト情報変更用シェルスクリプトを作成する 4
3.1.4 OpenShiftのlivenessProbe用シェルスクリプトを作成する 5
3.1.5 unitファイルを作成する 5
3.1.6 Dockerfileを作成する 6
3.1.7 Dockerイメージをビルドする 8
3.2 J2EEアプリケーションを配備したDockerイメージの作成 9

第4章 Red Hat OpenShift上でのJ2EEアプリケーションの実行 11

付録A 一括バックアップ・移入スクリプト 14

第1章 動作環境

1.1 対象製品

本資料の対象製品は、以下です。

- Interstage Application Server Enterprise Edition(64bit) V12

1.2 準備するもの

Interstage J2EE機能をOpenShiftで動作させるためには、以下を準備してください。

- Red Hat Enterprise Linux 7
- Red Hat OpenShift Container Platform
  Red Hat OpenShift Onlineは対象外です。
- Interstage Application Serverのサーバパッケージ DVD

本資料の手順は、以下の環境で検証しています。

- Red Hat Enterprise Linux 7.3
- Red Hat OpenShift Container Platform 3.3
- Interstage Application Server Enterprise Edition(64bit) V12.0.0

1.3 本手順書の記載範囲

- 本手順書は、一つのDockerコンテナ内で処理が完結するJ2EEアプリケーションをOpenShift上で動かすための手順を記載しています。
  ただし、以下は記載範囲外です。
  - 他のDockerコンテナとの連携方法
  - Dockerコンテナ外からのEJBアプリケーションの呼び出し方法
  - Entity Bean、Message Driven Beanの実行方法
- DockerイメージへのInterstage Application Serverへの緊急修正の適用方法については、本手順書の記載範囲外です。

第2章 概要

OpenShift上でInterstage J2EE機能を動作させるには、以下のことを行います。

1. Dockerイメージを作成する
2. Interstage J2EE機能をインストールしたDockerイメージを作成する
   Interstage J2EE機能をインストールした直後の状態のDockerイメージです。このDockerイメージをJ2EEのベアイメージと呼びます。
3. J2EEアプリケーションを配備したDockerイメージを作成する
4. Open ShiftでDockerイメージを実行する

第3章 Dockerイメージの作成

3.1 J2EEのベアイメージの作成

J2EEのベアイメージは以下の手順で作成します。

本手順は、Red Hat Enterprise Linuxにrootユーザログインして実行してください。

1. 以下の資材を作成し、同じディレクトリに配置します。以下の説明では、"/docker"に配置しています。
   - Interstageのサーバパッケージ DVDをマウントしたディレクトリ。以下の説明では"iaps"としています。
   - Interstageのインストールパラメーター CSVファイル。以下の説明では"j2ee.csv"としています。
   - Interstageのホスト情報を変更するために使用する以下のシェルスクリプトを格納したディレクトリ。以下の説明では"interstage/backup_restore"としています。
     - 資源の一括バックアップスクリプト isbackup
     - 資源の一括移入スクリプト isimport
     - isbackup、isimportを実行した後に、CORBAサービスのセットアップを行うスクリプト chhostname.sh
   - OpenShiftのlivenessProbeとして実行するためのシェルスクリプト(probe.sh)を格納するディレクトリ。以下の説明では、"interstage/probe"としています。
   - Interstage起動前後に必要な処理を実行する以下のunitファイルを格納するディレクトリ。以下の説明では"unitfiles"としています。
     - Intersgage起動前にchhostname.shを実行するunitファイル FJSVisas_start.service
     - Interstage起動後にInterstageの死活監視を開始するunitファイル FJSVisas_inspect_start.service
   - Dockerファイル。以下の説明では、"Dockerfile"としています。
2. Dockerイメージをビルドする

注意

J2EEのベアイメージのサイズは、約2.7GBです。

J2EEのベアイメージをビルドする過程で、Interstageのサーバパッケージの内容(約0.8GB)をDockerイメージ内コピーしますので、一時的に3.5GB程度のディスク容量が必要になります。本手順の実行はディスクの空きを十分確保した環境で行ってください。

3.1.1 Interstageのサーバパッケージ DVDのマウント

Interstageのサーバパッケージ DVDを、ローカルディスクに"iaps"としてマウントします。マウント先は、この後説明するDockerfileの格納先と同じにします。サーバパッケージ DVDの内容はDockerイメージ内にコピーされ、Interstageのインストールに使用されます。

Interstageのサーバパッケージ DVDは、Dockerイメージをビルド後にアンマウントしてください。

以下は、サーバパッケージ DVDを/docker/iapsにマウントする例です。

1. ローカルディスクにコピー先のディレクトリを作成します。

```
# mkdir -p /docker/iaps
```

2. サーバパッケージ DVDをDVDドライブに挿入し、以下のコマンドで/docker/iapsにマウントします。

```
# mount -t iso9660 -o loop /dev/cdrom /docker/iaps
```

3.1.2 インストールパラメーターCSVファイルを作成する

図1の内容のインストールパラメーターCSVファイルを、ファイル名"j2ee.csv"で作成し、/dockerに格納します。このパラメーターCSVファイルでInterstageをインストールすると、"install.sh"によるインストールで、"機能選択"で"J2EE互換"のみを選んだ場合と同じ機能がイン

ストールされます。インストールパラメーターCSVファイルの詳細については、Interstage Application Server Enterprise Editionのインストールガイドを参照してください。

```
installInfo,Name,isasinst
installInfo,OS,Linux
installInfo,softwareName,Interstage Application Server
installInfo,Edition,Enterprise Edition
installInfo,Version,V12.0.0
parameters,ServerType,application
parameters,InstallType,custom
parameters,SecurityMode,secure
parameters,SecurityGroup,root
parameters,JavaSEKind,JDK
parameters,MngConsolePort,12000
parameters,MngConsoleSSL,Y
parameters,MngConsoleMessageManual,Y
parameters,WebServerPort,80
parameters,CorbaPort,8002
parameters,FN_WEBSERVER,Y
parameters,FN_SECURE_COMMUNICATION,Y
parameters,FN_MANAGEMENT_CONSOLE,Y
parameters,FN_WEBSERVER_CONNECTOR,Y
parameters,FN_J2EE,Y
parameters,FN_JAVASE8,Y
```

図1 J2EE互換をインストールするインストールパラメーターCSVファイル

3.1.3 ホスト情報変更用シェルスクリプトを作成する

Dockerコンテナを起動したときのホスト名をInterstageの定義情報に反映するため、コンテナ起動時に一括バックアップ、および一括移入を実行し、その後でCORBAサービスの定義情報を変更するシェルスクリプト(chhostname.sh)を作成し、/docker/interstage/backup_restoreディレクトリに格納してください。chhostname.shの内容を図2に示します。

一括バックアップ用スクリプト、および一括移入用スクリプトは付録Aを参照して作成し、同じく/docker/interstage/backup_restoreディレクトリに格納してください。

```
#!/bin/sh
DIR=`dirname $0`
HOST=`/usr/bin/hostname`; export HOST
COMMON_PATH=${DIR}/backup ; export COMMON_PATH

if [ -f ${DIR}/hostname ] ; then
    PRE_HOSTNAME=`/usr/bin/cat ${DIR}/hostname`
else
    PRE_HOSTNAME=""
fi

/usr/bin/date > ${DIR}/chhostname.log 2>&1

while :
do
    /opt/FJSVisjmx/bin/isjmxstat >> ${DIR}/chhostname.log 2>&1
    if [ $? -eq 1 ] ; then
        break
    fi
    sleep 1
done

if [ "${PRE_HOSTNAME}" != "${HOST}" ] ; then
   if [ ! -d ${COMMON_PATH} ] ; then
      /bin/csh -f ${DIR}/isbackup >> ${DIR}/chhostname.log 2>&1
   fi
   if [ $? -eq 0 ] ; then
```

```
      /bin/csh -f ${DIR}/isimport >> ${DIR}/chhostname.log 2>&1
   fi
   if [ $? -eq 0 ] ; then
      /opt/FJSVod/bin/OD_set_env -n  ${HOST}                          >> ${DIR}/chhostname.log 2>&1
      /opt/FJSVod/bin/odchgservice -h ${HOST} InterfaceRep            >> ${DIR}/chhostname.log 2>&1
      /opt/FJSVod/bin/odchgservice -h ${HOST} InterfaceRep_e          >> ${DIR}/chhostname.log 2>&1
      /opt/FJSVod/bin/odchgservice -h ${HOST} InterfaceRepLock        >> ${DIR}/chhostname.log 2>&1
      /opt/FJSVod/bin/odchgservice -h ${HOST} InterfaceRepository     >> ${DIR}/chhostname.log 2>&1
      /opt/FJSVod/bin/odchgservice -h ${HOST} InterfaceRepository_e   >> ${DIR}/chhostname.log 2>&1
      /opt/FJSVod/bin/odchgservice -h ${HOST} NameService             >> ${DIR}/chhostname.log 2>&1
      echo ${HOST} > ${DIR}/hostname
      exit 0
   else
      /usr/bin/rm -fr ${DIR}/hostname
      exit 1
   fi
else
   echo "hostname is ${HOST}. It is same as the one used at a previous time." >> ${DIR}/chhostname.log
   exit 0
fi
```

図2 chhostname.sh

3.1.4 OpenShiftのlivenessProbe用シェルスクリプトを作成する

OpenShiftのlivenessProbeとして実行するスクリプト(probe.sh)を以下の内容で作成し、/docker/interstage/probeディレクトリに格納してください。

```
#!/bin/sh
/bin/ps -ef | /bin/grep /opt/FJSVisas/etc/HA/PRIMECLUSTER/IS_INSPECT | /bin/grep -v grep
if [ $? -eq 0 ] ; then
    exit 0
else
    exit 1
fi
```

図3 probe.sh

3.1.5 unitファイルを作成する

Interstage起動前にchhostname.shを実行するためのunitファイル(FJSVisas_start.service)と、Interstage起動後にInterstageの死活監視を開始するためのunitファイル(FJSVisas_inspect_start.service)を以下の内容で作成し、/docker/unitfilesに格納してください。

```
# Change hostname before IAPS start
[Unit]
Description=Change hostname before IAPS start
Wants=FJSVtd_start.service FJSVod_start.service ¥
       FJSVihs_start.service FJSVisjmx_start.service ¥
       FJSVisgui_start.service FJSVjs2su_start.service ¥
       FJSVsvmon_start.service
After=network.target network-online.target rsyslog.service ¥
      remote-fs.target nss-lookup.target ¥
      FJSVtd_stop.service FJSVod_stop.service FJSVihs_stop.service ¥
      FJSVisjmx_stop.service FJSVisgui_stop.service ¥
      FJSVjs2su_stop.service FJSVsvmon_stop.service
Before=FJSVtd_start.service FJSVod_start.service ¥
       FJSVihs_start.service FJSVisjmx_start.service ¥
       FJSVisgui_start.service FJSVjs2su_start.service ¥
       FJSVsvmon_start.service
[Service]
Type=oneshot
ExecStart=/interstage/backup_restore/chhostname.sh
```

```
ExecStop=/bin/true

[Install]
WantedBy=multi-user.target
```

図4 FJSVisas_start.service

```
# Inspect whether IAPS is alive
[Unit]
Description=Inspect whether IAPS is alive
Wants=FJSVtd_start.service FJSVod_start.service
After=FJSVtd_start.service FJSVod_start.service

[Service]
Type=simple
ExecStart=/opt/FJSVisas/etc/HA/PRIMECLUSTER/IS_INSPECT
ExecStop=/bin/true

[Install]
WantedBy=multi-user.target
```

図5 FJSVisas_inspect_start.service

3.1.6 Dockerfileを作成する

図6はJ2EEのベアイメージを作成するためのDockerfileの例です。図6の赤字の部分は環境に合わせて修正してください。Dockerファイル内の各部分の意味については、後述の (1)～(13)の説明を参照してください。

```
# Get Base image                                                    …(1)
FROM registry.access.redhat.com/rhel7.3

MAINTAINER 〈任意の文字列〉

# Create directories
RUN mkdir /work /interstage
# Copy Interstage install DVD to the docker image                   …(2)
COPY  ./iaps /work/iaps

# Copy parameter csv file to the docker image                       …(3)
COPY  ./j2ee.csv /work/j2ee.csv

# Copy shell scripts to the docker image                            …(4)
COPY  ./interstage /interstage
RUN chmod 0544 /interstage/backup_restore/* /interstage/probe/*

RUN yum -y update && yum clean all

# set Japanese locale                                               …(5)
RUN yum clean all
RUN rm -f /etc/rpm/macros.image-language-conf          && ¥
    sed -i '/^override_install_langs=/d' /etc/yum.conf && ¥
    yum -y reinstall glibc-common
env LANG=ja_JP.UTF-8 ¥
    LANGUAGE="ja_JP:ja" ¥
    LC_ALL="ja_JP.UTF-8"
RUN yum -y reinstall tzdata && ¥
    yum -y install yum-langpacks system-config-language   && ¥
    ln -snf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime && ¥
    sed -ri 's/en_US/ja_JP/' /etc/locale.conf

#systemd set up                                                    …(6)
ENV container docker
RUN yum -y swap -- remove fakesystemd -- install systemd systemd-libs
```

```
RUN (cd /lib/systemd/system/sysinit.target.wants/; for i in *; do [ $i == ¥
systemd-tmpfiles-setup.service ] || rm -f $i; done)      && ¥
rm -f /lib/systemd/system/multi-user.target.wants/*      && ¥
rm -f /etc/systemd/system/*.wants/*                      && ¥
rm -f /lib/systemd/system/local-fs.target.wants/*        && ¥
rm -f /lib/systemd/system/sockets.target.wants/*udev*    && ¥
rm -f /lib/systemd/system/sockets.target.wants/*initctl* && ¥
rm -f /lib/systemd/system/basic.target.wants/*           && ¥
rm -f /lib/systemd/system/anaconda.target.wants/*

# install rpms for Interstage                                    …(7)
RUN yum -y install cpp.x86_64 gcc.x86_64 gcc-c++.x86_64 gdb.x86_64 glibc-devel.x86_64 ¥
glibc-headers.x86_64 krb5-workstation.x86_64 libICE.x86_64 libSM.x86_64 libX11.x86_64 ¥
libX11-common.noarch libXau.x86_64 libXext.x86_64 libXi.x86_64 libXp.x86_64 libXrender.x86_64 ¥
libXt.x86_64 libXtst.x86_64 libstdc++-devel.x86_64 libtool-ltdl.x86_64 libxcb.x86_64 mpfr.x86_64 ¥
perl.x86_64 perl-Module-Pluggable.noarch perl-Pod-Escapes.noarch perl-Pod-Simple.noarch ¥
perl-libs.x86_64 perl-version.x86_64 redhat-lsb.x86_64 strace.x86_64 tcsh.x86_64 unixODBC.x86_64 ¥
lksctp-tools.x86_64 kernel-headers.x86_64 systemd.x86_64 rsyslog.x86_64

# Install syslogd                                                …(8)
RUN yum install -y rsyslog
RUN sed 's/$ModLoad imjournal/# $ModLoad imjournal/' -i /etc/rsyslog.conf    && ¥
    sed 's/$OmitLocalLogging on/$OmitLocalLogging off/' -i /etc/rsyslog.conf && ¥
    sed 's/$IMJournalStateFile imjournal.state/# $IMJournalStateFile imjournal.state/' -i /etc/rsyslog.conf

# For minimum ipc resources                                      …(9)
RUN echo '### for IAPS ###'            >> /etc/sysctl.conf && ¥
echo 'kernel.sem = 1100 35406 200 800' >> /etc/sysctl.conf && ¥
echo 'kernel.shmall =  4294967296' >> /etc/sysctl.conf && ¥
echo 'kernel.shmmax = 68719476736' >> /etc/sysctl.conf && ¥
echo 'kernel.shmmni =        4315' >> /etc/sysctl.conf && ¥
echo 'kernel.msgmax =       65536' >> /etc/sysctl.conf && ¥
echo 'kernel.msgmnb =     4194304' >> /etc/sysctl.conf && ¥
echo 'kernel.msgmni =        8199' >> /etc/sysctl.conf

# Change kernel parameters and remove IPCs before systemd start  …(10)
RUN echo ''                                >> /etc/rc.local && ¥
echo '# change kernel parameters'          >> /etc/rc.local && ¥
echo '/usr/sbin/sysctl -p'                 >> /etc/rc.local && ¥
echo '# remove IPCs before systemd start' >> /etc/rc.local && ¥
echo "QUES=¥`/usr/bin/ipcs -q | /usr/bin/grep 0x | /usr/bin/awk '{ print(¥$2) }' | ¥
            /usr/bin/tr '¥n' ' ' ¥`" >> /etc/rc.local && ¥
echo "MSGS=¥`/usr/bin/ipcs -m | /usr/bin/grep 0x | /usr/bin/awk '{ print(¥$2) }' | ¥
            /usr/bin/tr '¥n' ' ' ¥`" >> /etc/rc.local && ¥
echo "SEMS=¥`/usr/bin/ipcs -s | /usr/bin/grep 0x | /usr/bin/awk '{ print(¥$2) }' | ¥
            /usr/bin/tr '¥n' ' ' ¥`" >> /etc/rc.local && ¥echo ''  >> /etc/rc.local && ¥
echo 'for QUE in ${QUES}'                  >> /etc/rc.local && ¥
echo 'do'                                  >> /etc/rc.local && ¥
echo '    /usr/bin/ipcrm -q ${QUE}'        >> /etc/rc.local && ¥
echo 'done'                                >> /etc/rc.local && ¥
echo ''                                    >> /etc/rc.local && ¥
echo 'for MSG in ${MSGS}'                  >> /etc/rc.local && ¥
echo 'do'                                  >> /etc/rc.local && ¥
echo '    /usr/bin/ipcrm -m ${MSG}'        >> /etc/rc.local && ¥
echo 'done'                                >> /etc/rc.local && ¥
echo ''                                    >> /etc/rc.local && ¥
echo 'for SEM in ${SEMS}'                  >> /etc/rc.local && ¥
echo 'do'                                  >> /etc/rc.local && ¥
echo '    /usr/bin/ipcrm -s ${SEM}'        >> /etc/rc.local && ¥
echo 'done'                                >> /etc/rc.local && ¥
/usr/bin/chmod u+x /etc/rc.local                            && ¥
/usr/bin/systemctl enable rc-local.service > /dev/null 2>&1
```

```
# Install unit files for Interstage                                  …(11)
COPY ./unitfiles/FJSVisas_start.service /lib/systemd/system
COPY ./unitfiles/FJSVisas_inspect_start.service /lib/systemd/system
RUN /usr/bin/chmod 0644 /lib/systemd/system/FJSVisas_start.service        && ¥
/usr/bin/systemctl enable FJSVisas_start.service                          && ¥
/usr/bin/chmod 0644 /lib/systemd/system/FJSVisas_inspect_start.service    && ¥
/usr/bin/systemctl enable FJSVisas_inspect_start.service

# Install Interstage                                                 …(12)
ENV CIR_INST_SKIP yes
ENV TERM xterm
RUN /work/iaps/installer/install.sh -s /work/j2ee.csv

# specfy executable when run this docker image                       …(13)
CMD [ "/usr/sbin/init" ]

# remove Interstage installer                                        …(14)
RUN rm -fr /work/j2ee.csv /work/iaps
```

図6 J2EEのベアイメージ用Dockerfile

(1) ベースとなるDockerイメージを指定します。

(2) Interstageのインストール媒体を展開したディレクトリiapsをDockerイメージにコピーします。COPYコマンドでは、./iaps/*がDockerイメージの/work/iapsにコピーされます。

(3) 3.1.2節で作成したパラメーターCSVファイルをDockerイメージの/work/j2ee.csvにコピーします。

(4) 3.1.3節と3.1.4節で作成した資材をDockerイメージの/interstageにコピーし、実行権を付与します。

(5) Dockerイメージのシステムロケール、タイムゾーンを日本に変更します。

(6) systemdをインストールしてDockerコンテナでは不要なサービスを起動しないようにします。

(7) Interstageの必須rpmをインストールします。必須rpmについては、"Interstage Application Serverインストールガイド"の"前提基本ソフトウェア"を参照してください。

(8) syslogdをインストールします。

(9) コンテナ起動時に"sysctl -p"を実行してIPCリソースの値を変更するために/etc/sysctl.confを修正します。IPCリソースの値（赤字）は、"Interstage Application Serverチューニングガイド"に従って見積もってください。

(10) コンテナ起動時に"sysctl -p"を実行し、不要なIPCリソースを削除するために、/etc/rc.localファイルを編集し、systemdの初期化時に実行するようにします。

(11) 3.1.5節で作成したunitファイルをインストールし、有効にします。

(12) サイレントモードでInterstageのJ2EE機能をインストールします。

(13) Dockerイメージを起動したときに実行されるコマンドを指定します。

(14) 不要になったファイルをDockerイメージから削除します。

rootのパスワードを設定したい場合は、Dockerfileに以下の行を追加します。

RUN echo "root:<rootのパスワード>" | chpasswd

rootのパスワードはDockerコンテナを起動したあとでも設定/変更することができます。

3.1.7 Dockerイメージをビルドする

"docker build"コマンドでDockerイメージをビルドしてください。

以下は、Dockerファイルなどの資材格納ディレクトリが"/docker"、ターゲット名が"j2eebear"でビルドする例です。

```
# docker build -t j2eebear /docker
```

注意

Dockerイメージをビルド中、Interstageのインストールフェーズにおいて、以下のメッセージや警告が出力されますが、無視してください。

Failed to get D-Bus connection: Operation not permitted

warning: %post(FJSVod-14.0.0-1.0.x86_64) scriptlet failed, exit status 1

warning: %post(FJSVisgui-12.0-0.0.x86_64) scriptlet failed, exit status 1

3.2 J2EEアプリケーションを配備したDockerイメージの作成

3.1で作成したJ2EEのベアイメージを実行し、IJServerワークユニットの作成、およびアプリケーションの配備を行います。

1. 3.1で作成したDockerイメージからDockerコンテナを起動します。
   必ず以下のオプションを付けて起動してください。
   --privileged
   --hostname=<ホスト名>

   以下は、Dockerイメージ"j2eebear"から、Dockerコンテナ"j2eebear_c"をホスト名"RH73IAPS"で起動する例です。以下の例では、Interstage管理コンソールのポートをDockerホストの12000ポートにマッピングしています。

```
# docker run --privileged --name j2eebear_c -di --hostname=RH73IAPS -p 12000:12000 j2eebear
```

   ポイント

   この後に行うIJServerの作成、およびアプリケーションの配備でInterstage管理コンソールを使用しない場合、-pオプション(ポートマッピング)の指定は必須ではありませんが、-pオプションを付けたDockerコンテナをイメージ化することで、4章で説明するOpenShiftのpodとして動作するためのテンプレートファイルに、Serviceの定義が追加されるようになります。4章の説明では、12000ポートを使うサービスが一つ存在するテンプレートファイルが出力される前提で説明しています。

2. DockerコンテナでIJServerワークユニットを作成し、J2EEアプリケーションを配備します。
   コマンドでIJServerワークユニットの作成、アプリケーションの配備を行う場合は、アプリケーションやIJServer定義ファイルをDockerコンテナに配置してください。以下の例では、"docker cp"コマンドでアプリケーション(myApp.ear)とIJServer定義ファイル(ijserver.xml)をDockerコンテナにコピーし、isj2eeadminコマンド、およびijsdeploymentコマンドを使って、IJServerワークユニットの作成とアプリケーションの配備を行っています。

```
# docker cp myApp.ear j2eebear_c:/work/
# docker cp ijserver.xml j2eebear_c:/work/
# docker exec -it j2eebear_c /bin/bash
[root@RH73IAPS /]# /opt/FJSVj2ee/bin/isj2eeadmin ijserver -a -f /work/ijserver.xml
UX:isj2eeadmin: 情報: isj2ee2100:IJServerを登録しました NAME=IJServer
[root@RH73IAPS /]# /opt/FJSVj2ee/bin/ijsdeployment -n IJServer -f /work/myApp.ear
UX:DEPLOY: 情報: DEP5050: 配備処理が完了しました: ファイル名=/work/myApp.ear
```

3. Dockerコンテナ内でInterstageを停止します。
   /opt/FJSVisas/bin/isservicestop.shコマンドを使うと、すべての機能を停止することができます。

```
[root@RH73IAPS /]# /opt/FJSVisas/bin/isservicestop.sh
```

4. OpenShiftのlivenessProbeを使ってInterstageの監視を行うため、以下の設定を行ってください。

   ー Interstage運用環境にInterstage HTTP Serverサービスを追加する。

   ー Interstage HTTP Serverの停止時にInterstageを停止するようにする。

```
[root@RH73IAPS /]# ismodifyservice -a FJapache
UX:ismodifyservice: 情報: is30199:コマンドが正常に終了しました
[root@RH73IAPS /]# ismodifyservice -m mode1
UX:ismodifyservice: 情報: is30199:コマンドが正常に終了しました
```

5. 3.1.3で用意した一括バックアップスクリプトを使って、アプリケーションを配備した状態の資源をバックアップします。
   バックアップ先ディレクトリは、chhostname.shのCOMMON_PATHに設定しているディレクトリです。

```
[root@RH73IAPS /]# COMMON_PATH=/interstage/backup_restore/backup /bin/csh -f /interstage/backup_restore/isbackup
```

   ここでバックアップした資源は、Dockerコンテナ起動時にInterstageの資源内に保持しているホスト情報を変更するために使用されます。

   Dockerコンテナ内のInterstageに変更を加えた場合は、必ずこの手順を実施してからDockerイメージを作成してください。

6. 次回のDockerコンテナの起動に不要な以下のファイルを削除し、Dockerコンテナからexitします。

   ー ホスト情報変更用シェルスクリプト(chhostname.sh)がホスト名を保持するファイル、/interstage/backup_restore/hostname

   ー シスログの出力先、/var/log/messages

```
[root@RH73IAPS /]# /usr/bin/rm -f /interstage/backup_restore/hostname /var/log/messages
[root@RH73IAPS /]# exit
```

7. J2EEアプリケーションを配備したDockerコンテナをコミットして、J2EEアプリケーションを配備した状態のDockerイメージを作成します。
   以下は、"j2eemyapp"という名前のDockerイメージを作成する例です。

```
# docker commit j2eebear_c j2eemyapp
```

8. Dockerコンテナ(j2eebear_c)を停止し、削除します。

```
# docker stop j2eebear_c
j2eebear_c
# docker rm j2eebear_c
j2eebear_c
```

第4章 Red Hat OpenShift上でのJ2EEアプリケーションの実行

以下の手順を実施して、Interstage Application Serverに配備したJ2EEアプリケーションをOpenShift上で実行します。

参考

Interstage Application ServerのJ2EE機能を含むDockerコンテナは、root権限で特権コンテナとして実行する必要があります。root権限、かつ特権コンテナとしてDockerイメージを実行できるように、OpenShiftのユーザ、およびプロジェクトを設定してください。詳細は以下を参照してください。

```
Red Hat OpenShift Documentation(https://docs.openshift.com/)
    OpenShift Container Platform
        Cluster Administration
            Managing Security Context Constraints
```

1. OpenShiftからアクセス可能なリポジトリに、3.2で作成したDockerイメージを登録します。
   本手順では、Dockerイメージを作成した環境で実行します。
   以下は、OpenShiftからアクセス可能なリポジトリのIPアドレスが192.168.100.102、リポジトリのポート番号が5000の例です。イメージ名"j2eemyapp", タグ名"latest"で登録しています。

```
# docker tag j2eemyapp 192.168.100.102:5000/j2eemyapp:latest
# docker push 192.168.100.102:5000/j2eemyapp:latest
```

2. OpenShiftにログインします。

3. OpenShift上でリポジトリに登録したDockerイメージを取得してpod上で動くInterstageのテンプレートを作成します。以下の例では、j2eemyapp.yamlというファイル名のテンプレート(yaml形式)が出力されます。

```
# oc new-app --docker-image=192.168.100.102:5000/j2eemyapp:latest --name j2eemyapp -o yaml > j2eemyapp.yaml
```

4. 3で出力されたテンプレートを編集して、以下の設定を行います。

   - Interstageが動作するDockerコンテナを特権コンテナとして起動する。
   - libvenessProbeでInterstageを監視する。

   "securityContext"と"livenessProbe"(以下の赤字の部分)を、DeploymentConfigのspec.template.spec.containersの中に追加します。インデントは以下の例(image行と同じところ)に合わせてください。

```
--抜粋開始--
      spec:
        containers:
        - image: 192.168.100.102:5000/j2eemyapp:latest
          name: j2eemyapp
          ports:
          - containerPort: 12000
            protocol: TCP
          resources: {}
          securityContext:
            privileged: true
          livenessProbe:
            exec:
              command:
              - /interstage/probe/probe.sh
            initialDelaySeconds: 60
            timeoutSeconds: 1
    test: false
--抜粋終了--
```

注意

- 追加する行の場所やインデントは、必ず上の例に合わせてください。追加する箇所のインデントがずれていると特権コンテナとして起動できません。
- Interstageの起動完了前にlivenessProbeが実行されると、podが再起動を繰り返すため、initialDelaySecondsに指定する値は、systemdの起動時間(秒)より大きい値(systemdの起動時間の2倍を目安)とし、環境に合わせて調節してください。
  podが再起動を繰り返す場合、シスログを参照し、"Started Inspect whether IAPS is alive."のメッセージが出力されておらず、かつ"Received SIGTERM."のメッセージが出力されている場合は、initialDelaySecondsの値を大きくしてください。
  systemdの起動時間は、3.2節の手順のあとにアプリケーションを配備したDockerコンテナを起動した状態、あるいは、livenessProbeを指定せずにOpenShift上でpodを起動した環境で、systemd-analyzeコマンドを実行することで確認できます。
- Interstageのpodを監視する必要が無い場合は、libenessProbeの設定を書く必要はありません。

5. 4で編集したテンプレートを更に編集して、Interstage HTTPサービスのWebサーバFJapacheをPod上のサービスとして定義します。

   テンプレートの以下の部分をコピーして、この記述のすぐ後ろにペーストしてください。

```
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
      app: j2eemyapp
    name: j2eemyapp
  spec:
    ports:
    - name: 12000-tcp
      port: 12000
      protocol: TCP
      targetPort: 12000
    selector:
      app: j2eemyapp
      deploymentconfig: j2eemyapp
  status:
    loadBalancer: {}
```

   以下の赤字の部分(FJapache用のサービスの定義のmetadata.name、spec.ports配下のportおよびtargetPort)を変更してください。

   metadata.nameは、他のサービスと重複しない名前を指定してください。

   以下は、Interstage管理コンソール、およびFJapacheの、外部からアクセスするポート番号をそれぞれ12000、8082とする例です。

```
--抜粋開始--
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
      app: j2eemyapp
    name: j2eemyapp-isadmin
  spec:
    ports:
    - name: 12000-tcp
      port: 12000
      protocol: TCP
      targetPort: 12000
    selector:
      app: j2eemyapp
```

```
      deploymentconfig: j2eemyapp
  status:
    loadBalancer: {}
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
      app: j2eemyapp
    name: j2eemyapp-fjapache
  spec:
    ports:
    - name: 8082-tcp
      port: 8082
      protocol: TCP
      targetPort: 80
    selector:
      app: j2eemyapp
      deploymentconfig: j2eemyapp
  status:
    loadBalancer: {}
--抜粋終了--
```

参考

- 上の例では、サービス名と機能の対応の分かりやすさのためにInterstage管理コンソール用のサービス名をj2eemyappからj2eemyapp-isadminに変更していますが、必須ではありません。
- OpenShift上でInterstage管理コンソールを使用しない場合は、Interstage管理コンソール用のサービスの定義を削除しても構いません。

6. 5で編集したテンプレートを使ってコンテナを起動します。

```
# oc create -f j2eemyapp.yaml
```

7. FJapacheをアクセスするURLを外部へ公開するための、routeを作成します。

   以下は、サービスj2eemyapp-fjapacheをURL http://j2eemyapp.fj-iaps.comで公開するrouteを作成する例です。アプリケーションには、http://j2eemyapp.fj-iaps.com /<Webアプリケーション名>等でアクセスできるようになります。赤字の部分を適宜変更してください。

```
# oc expose service/j2eemyapp-fjapache --hostname=j2eemyapp.fj-iaps.com
```

8. 7で--hostnameオプションに指定したホスト名を名前解決できるようにしてください。

参考

- 本資料の説明では、Interstage管理コンソールはSSL通信するようにインストールされています。SSL通信用のrouteを作成する場合は、"oc create route edge"コマンドを使用します。
- サービスごとにExternal IPを指定することで、URLを公開することもできます。その場合、URLはhttp(s)://<External IP>:<Port>/<Webアプリケーション名>等となります。

付録A　一括バックアップ・移入スクリプト

/opt/FJSVisas/sample/backup_restoreにインストールされるisbackup、isimportをコピー、修正して作成してください。

isbackupの修正

以下の修正を行ってください。

1. "DEFINITION_PART:"の前に以下の命令を追加し、[Common]セクションの変数COMMON_PATHの設定行をコメントアウトしてください。
   /usr/bin/rm -fr $COMMON_PATH

```
#<Notice>
#
#
#============================================================
/usr/bin/rm -fr $COMMON_PATH
#============================================================================
DEFINITION_PART:
#============================================================================
#[Common]
#set COMMON_PATH=/tmp/backup
#------------------------------------------------------------
```

2. 以下の変数の値をoffにしてください。
   ES_TARGET、OTS_TARGET、IHS_TARGET、SSOSV_TARGET、SSOAC_TARGET、SSOAZ_TARGET、WSC_TARGET、ISSCS_TARGET、IREP_TARGET、ISCM_TARGET、JAVAEE6_TARGET、JAVAEE7_TARGET

3. "set JMS_TARGET=on"の行の直前に、以下の命令を追加してください。
   setenv CLASSPATH /opt/FJSVj2ee/lib/isj2ee.jar:/opt/FJSVjms/lib/fjmsprovider.jar
   setenv LD_LIBRARY_PATH /opt/FJSVjms/lib

```
#------------------------------------------------------------
#[JMS]
setenv CLASSPATH /opt/FJSVj2ee/lib/isj2ee.jar:/opt/FJSVjms/lib/fjmsprovider.jar
setenv LD_LIBRARY_PATH /opt/FJSVjms/lib
set JMS_TARGET=on
```

4. "set J2EE_TARGET=on"の行の直前に、以下の命令を追加してください。
   setenv PATH /opt/FJSVawjbk/jdk8/bin:${PATH}

```
#------------------------------------------------------------
#[J2EE]
setenv PATH /opt/FJSVawjbk/jdk8/bin:${PATH}
set J2EE_TARGET=on
```

isimportの修正

以下の修正を行ってください。

1. [Common]セクションの変数HOSTおよびCOMMON_PATHの設定行をコメントアウトしてください。

```
#==========================================================================
#[Common]
#set HOST=import_host
#set COMMON_PATH=/tmp/backup
```

2. isbackupの2, 3, 4と同じ修正をしてください。