FUJITSU

FUJITSU Software
Interstage List Works V10.4.0

Red Hat OpenShift上での動作手順書

Linux

CIB-2913-19-LWP0050(00)
2019年8月

はじめに

本資料は、Interstage List WorksをRed Hat OpenShift V3上で動作させる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 List Works Enterprise Edition V10.4.0(拡張パッケージ)	List Works
Interstage Application Server Standard-J V11.1.1	Interstage
Red Hat OpenShift	OpenShift

前提知識

本資料を読む場合、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 List Worksに関する基本的な知識
- Interstage Application Serverに関する基本的な知識
- UpdateAdvisor(ミドルウェア)に関する基本的な知識

検証環境

本資料の手順は、以下を使用して検証しています。

- Red Hat Enterprise Linux 7.6
- Red Hat OpenShift Container Platform 3.9
- Linux(Intel64)版 Interstage List Works Enterprise Edition V10.4.0(拡張パッケージ)
- Linux(Intel64)版 Interstage Application Server Standard-J Edition V11.1.1

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

2019年8月 初版

目　次

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。

- Linux(Intel64)版 Interstage List Works Enterprise Edition V10.4.0(拡張パッケージ)
- Linux(Intel64)版 Interstage Application Server Standard-J Edition V11.1.1

1.2 準備するもの

List WorksをOpenShift上で動作させるためには、事前に以下を準備してください。

- Red Hat Enterprise Linux 7
- Red Hat OpenShift Container Platform
- List WorksのパッケージDVD(1枚目)
- Interstage Application ServerのパッケージDVD
- UpdateAdvisor(ミドルウェア)およびList Worksの最新の緊急修正

1.3 本資料の記載範囲

本資料は、OpenShift上でList Worksを使用してWeb連携機能を利用する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
- 永続ストレージの利用方法

第2章 概要

OpenShift上でList WorksのWeb連携機能を動作させるには、以下のことを行います。

1. List WorksとInterstageをインストールしたDockerイメージを作成する

   ポイント

   List WorksとInterstageをインストールした直後の状態のDockerイメージです。本資料では、このDockerイメージをList WorksとInterstageのベアイメージと呼びます。

2. List WorksとInterstageの環境をセットアップしたDockerイメージを作成する
3. OpenShiftでDockerイメージを実行する

第3章 Dockerイメージの作成

3.1 List Worksのベアイメージの作成

List WorksとInterstage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ユーザで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。
   - List WorksのパッケージDVD(1枚目)をマウントしたディレクトリ。以下の説明では“lw_mnt”としています。
   - InterstageのパッケージDVDをマウントしたディレクトリ。以下の説明では“iaps_mnt”としています。
   - List Works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“lw_param.csv”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“iapsv_param.csv”としています。
   - Dockerファイル。以下の説明では、“dockerfile”としています。
2. Dockerイメージをビルドします。

注意

List Worksのベアイメージのサイズは約5.95GBです。

List Worksのベアイメージをビルドする過程で、InterstageのパッケージDVDの内容(約1.29GB)とList WorksのパッケージDVDの内容(約1.53GB)をDockerイメージ内にコピーしますので、一時的に8.77GB程度のディスク容量が必要になります。本手順の実行は、ディスクの空き容量を十分確保した環境で行ってください。

3.1.1 対象製品のパッケージDVDのコピー

List WorksのパッケージDVDを、ローカルディスクに“lw_mnt”としてマウントします。また、InterstageのパッケージDVDを、ローカルディスクに“iaps_mnt”としてマウントします。マウント先は、後述のdockerfileの格納先と同じにします。パッケージDVDの内容はDockerイメージ内にコピーされ、List Worksのインストールに使用されます。

以下は、パッケージDVDの内容を/docker/lw_mnt、/docker/iaps_mntにコピーする例です。

1. ローカルディスクにマウント先のディレクトリを作成します。

```
# mkdir -p /docker/lw_mnt /docker/iaps_mnt
```

2. List WorksのパッケージDVDをマウントします。

```
# mount -t iso9660 -r /dev/cdrom/ /mnt
```

3. List WorksのパッケージDVDをローカルディスクにコピーします。

```
# cp -a /mnt/* /docker/lw_mnt
```

4. List WorksのパッケージDVDをアンマウントします。

```
# umount /mnt
```

5. InterstageのパッケージDVDをマウントします。

```
# mount -t iso9660 -r /dev/cdrom/ /mnt
```

6. InterstageのパッケージDVDをローカルディスクにコピーします。

```
# cp -a /mnt/* /docker/iaps_mnt
```

7. Interstageのパッケージ DVDをアンマウントします。

```
# umount /mnt
```

3.1.2 インストールパラメーターCSVファイルの作成

Interstageのインストールパラメーター CSVファイルを、図1の内容に従って作成します。ファイル名は"iapsv_param.csv"で、/dockerに格納します。記載内容の詳細についてはInterstage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,Standard-J Edition
installInfo,Version,V11.1.1
parameters,ServerType,application
parameters,InstallType,custom
parameters,SecurityMode,secure
parameters,SecurityGroup,root
parameters,JavaSEKind,JDK
parameters,JavaEE6JdkVersion,JDK7
parameters,JavaEE6AdminUser,admin
parameters,JavaEE6AdminPassword,adminadmin
parameters,JavaEE6DomainAdminPort,12011
parameters,JavaEE6HttpListenerPort,28282
parameters,JavaEE6HttpsListenerPort,28383
parameters,JavaEE6IiopPort,23610
parameters,JavaEE6IiopSSLPort,23611
parameters,JavaEE6IiopMutualauthPort,23612
parameters,JavaEE6JmxAdminPort,18686
parameters,JavaEE6CommonDirectory,/var/opt/FJSVisje6
parameters,MngConsolePort,12000
parameters,MngConsoleSSL,Y
parameters,MngConsoleMessageManual,Y
parameters,WebServer22Port,81
parameters,CorbaPort,8010
parameters,FN_JAVAEE5,N
parameters,FN_WEBSERVER,N
parameters,FN_WEBSERVER22,Y
parameters,FN_SECURE_COMMUNICATION,Y
parameters,FN_SSO_BS,Y
parameters,FN_SSO_AS,Y
parameters,FN_SSO_RS,Y
parameters,FN_DIRECTORY_SERVICE,Y
parameters,FN_MANAGEMENT_CONSOLE,Y
parameters,FN_WEBSERVER_CONNECTOR,N
parameters,FN_J2EE,N
parameters,FN_WEBSERVER_CONNECTOR22,Y
parameters,FN_FRAMEWORK,Y
parameters,FN_JAVASE6,Y
parameters,FN_JAVASE7,Y
parameters,FN_SAMPLE_APL,Y
parameters,FN_XML,Y
parameters,FN_JAVAEE6,Y
```

図1 Interstageを/optにインストールするインストールパラメーターCSVファイルの内容

List Worksのインストールパラメーター CSVファイルを、図2の内容に従って作成します。ファイル名は"lw_param.csv"で、/dockerに格納します。記載内容の詳細についてはList Worksのインストールガイドを参照してください。

```
installInfo,softwareName,List Works Enterprise Edition
installInfo,OS,Linux
installInfo,Version,V10.4.0
installInfo,Edition,Enterprise Edition
installInfo,Name,islwinst
"parameters","InstallType_Sv","NO"
"parameters","InstallType_Gw","YES"
"parameters","InstallType_Pa","NO"
```

図2 List Worksを/optにインストールするインストールパラメーターCSVファイルの内容

3.1.3 ホスト情報変更用シェルスクリプトの作成

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3に示します。

```
#!/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3 chhostname.sh

3.1.4 unitファイルの作成

Interstage起動前にchhostname.shを実行するためのunitファイル(FJSVisas_start.service)を図4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 FJSVahs_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 ¥
FJSVahs_stop.service
Before=FJSVtd_start.service FJSVod_start.service ¥
FJSVihs_start.service FJSVisjmx_start.service ¥
FJSVisgui_start.service FJSVjs2su_start.service ¥
FJSVsvmon_start.service FJSVahs_start.service
[Service]
Type=oneshot
ExecStart=/interstage/backup_restore/chhostname.sh
ExecStop=/bin/true

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

図4 FJSVisas_start.service

3.1.5 dockerfileの作成

図5は、List WorksとInterstageのベアイメージを作成するためのdockerfileの例です。図5の下線部は環境に合わせて修正してください。dockerfile内の各部分の意味については、後述の(1)～(14)の説明を参照してください。

```
# Get Base image …(1)
FROM registry.access.redhat.com/rhel7.6

MAINTAINER <イメージの作者情報を示す任意の文字列>

# Create directories
RUN mkdir /work /interstage

# Copy List Works and Interstage install DVD to the docker image …(2)
COPY ./lw_mnt /work/lw_mnt/
COPY ./iaps_mnt /work/iaps_mnt/

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

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

RUN yum -y update && yum clean all

RUN /bin/mkdir -p /etc/selinux/targeted/contexts/
RUN echo '<busconfig><selinux></selinux></busconfig>' /etc/selinux/targeted/contexts/dbus_contexts

# 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     ¥
    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 List Works and 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 ¥
    glibc.x86_64 glibc.i686 nss-softokn-freebl.x86_64 nss-softokn-freebl.i686 ¥
    libgcc.x86_64 libgcc.i686 libstdc++.x86_64 libstdc++.i686 zip unzip ncompress

# 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 "fs.file-max = 65536" >> /etc/sysctl.conf && ¥
echo "kernel.shmmax = 18446744073692774399" >> /etc/sysctl.conf && ¥
echo "kernel.shmmni = 4256" >> /etc/sysctl.conf && ¥
echo "kernel.sem = 2500 329600 320 4380" >> /etc/sysctl.conf && ¥
echo "kernel.msgmax = 8192" >> /etc/sysctl.conf && ¥
echo "kernel.msgmnb = 16384" >> /etc/sysctl.conf && ¥
echo "kernel.msgmni = 3743" >> /etc/sysctl.conf && ¥
echo "kernel.shmall = 18446744073703682559" >> /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
RUN /usr/bin/chmod 0644 /lib/systemd/system/FJSVisas_start.service && ¥
/usr/bin/systemctl enable FJSVisas_start.service

RUN /bin/mv /usr/bin/systemctl /usr/bin/systemctl.org
RUN echo 'exit 0' > /usr/bin/systemctl
RUN /usr/bin/chmod +x /usr/bin/systemctl

# Install List Works and Interstage …(12)
WORKDIR /work
ENV CIR_INST_SKIP yes
ENV TERM xterm
RUN /work/lw_mnt/installer2/citool/silentinstall /work/lw_param.csv
RUN /work/iaps_mnt/install.sh -s /work/iapsv_param.csv

RUN /bin/mv /usr/bin/systemctl.org /usr/bin/systemctl

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

# Remove List Works and Interstage installer …(14)
WORKDIR /
RUN rm -fr /work/lw_param.csv /work/lw_mnt
RUN rm -fr /work/iapsv_param.csv /work/iaps_mnt
```

図5 List WorksとInterstageのベアイメージ用dockerfileの例

(1)ベースとなるDockerイメージを指定します。

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

(3)3.1.2で作成したインストールパラメーターCSVファイルをDockerイメージの/work/lw_param.csvおよび、/work/iapsv_param.csvにコピーします。

(4)3.1.3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)List WorksとInterstageの必須パッケージ(rpm)をインストールします。必須パッケージについては、List WorksとInterstageのインストールガイドを参照してください。

(8)コンテナ(pod)内にシステムログを出力する場合はsyslogdをインストールします。本手順を省略すると、システムログはコンテナ(pod)内に出力されません。

(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)3.1.4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List WorksとInterstage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。

3.1.6 Dockerイメージのビルド

“docker build”コマンドでDockerイメージをビルドします。

以下は、dockerfileなどの資材格納ディレクトリが“/docker”、ターゲット名が“lwee_bare”でビルドする例です。

```
# docker build -t lwee_bare /docker
```

3.1.7 対象製品のパッケージDVDの削除

3.1.1でコピーしたList WorksおよびInterstageのパッケージDVDの内容を削除します。

```
# rm -fr /docker/lw_mnt
# rm -fr /docker/iaps_mnt
```

3.2 List Worksの環境設定とDockerイメージの作成

3.1で作成したベアイメージを実行し、List Worksの環境設定を行います。

3.2.1 ベアイメージからのDockerコンテナの起動

必ず以下のオプションをつけて起動してください。

--privileged

以下は、DockerイメージからDockerコンテナを起動する例です。以下の例ではWeb連携機能を使用するポートをDockerホストの81ポートにマッピングしています。

- Web連携サーバのDockerコンテナの起動例（Dockerイメージ名：lwee_bare、Dockerコンテナ名：lwee_bare_c1、ホスト名：RH76LWEE）

```
# docker run --name lwee_bare_c1 --privileged -di --hostname=RH76LWEE -p 8081:81 lwee_bare
```

3.2.2 List Worksの環境設定

List Worksの環境設定を行います。

参考

アクセスログの格納ディレクトリは、必要に応じて永続ストレージを使用してください。

参照

List Worksの環境設定の詳細については、List Worksのオンラインマニュアル「帳票保管活用機能 セットアップガイド(拡張パッケージ」を参照してください。

3.2.3 List Worksの緊急修正の適用

“docker exec”コマンドでDockerコンテナにログインし、UpdateAdvisor(ミドルウェア)のインストールおよび、List Worksの最新の緊急修正の適用を行います。

- Dockerコンテナへのログイン例

```
# docker exec -it lwee_bare_c1 /bin/bash
[root@RH76LWEE /]#
```

参考

UpdateAdvisor(ミドルウェア)のインストール資材、および緊急修正モジュールは、“docker cp”コマンドを使用してDockerコンテナに複写します。

参照

UpdateAdvisor(ミドルウェア)のインストール方法については、UpdateAdvisor(ミドルウェア)のヘルプを参照してください。

List Worksの緊急修正の適用方法については、緊急修正の修正情報ファイル、およびUpdateAdvisor(ミドルウェア)のヘルプを参照してください。

3.2.4 Web連携機能の環境構築

Web連携機能のセットアップ用シェルを実行して、List WorksのWeb連携機能の環境を構築します。

```
[root@RH76LWEE work]# /opt/FJSVisje6/var/pcmi/isje6/FJSVpcmi start
[root@RH76LWEE work]# /opt/FJSVisje6/glassfish/bin/asadmin start-domain
[root@RH76LWEE work]# echo AS_ADMIN_PASSWORD=adminadmin > /work/passwordfile
[root@RH76LWEE work]# /opt/FJSVlw-gw/bin/lw-gwsetenv.sh 1 3 12011 admin /work/passwordfile
```

参照

セットアップ用シェルの詳細については、オンラインマニュアルの「帳票保管活用機能 セットアップガイド(拡張パッケージ)」を参照してください。

3.2.5 Dockerイメージの作成

以下の手順で、Dockerイメージをコミットし、新しいDockerイメージを作成します。

1. 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 シスログの出力先(/var/log/messages)

```
[root@RH76LWEE work]# rm -f /var/log/messages
[root@RH76LWEE work]# exit
```

2. List Worksの環境設定を行ったDockerコンテナをコミットし、新しいDockerイメージを作成します。

   ー 新しいDockerイメージ名がlweeappの場合の例

```
# docker commit lwee_bare_c1 lweeapp
```

3. Dockerコンテナを停止し、削除します。

   ー Dockerコンテナ名がlwee_bare_c1の場合の例

```
# docker stop lwee_bare_c1
lwee_bare_c1
# docker rm lwee_bare_c1
lwee_bare_c1
```

第4章 Red Hat OpenShift上でのアプリケーションの実行

以下の手順を実施して、OpenShift上でList WorksのWeb連携機能を使用します。

参考

- List Worksを動作させる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の設定をしてください。
- 本章ではWeb連携での操作手順を説明しています。

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“lweeapp”、タグ名“latest”で登録しています。

```
# docker tag lweeapp 192.168.100.102:5000/lweeapp:latest
# docker push 192.168.100.102:5000/lweeapp: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List Worksのテンプレートを作成します。以下の例では、lweeapp.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/lweeapp:latest --name lweeapp -o yaml > lweeapp.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。

   ー List Worksが動作するDockerコンテナを特権コンテナとして起動する。

   ー ServiceのExternal-IPとPortsを設定する。

   “securityContext”(下線部)を、DeploymentConfigのspec.template.spec.containersの中に追加します。下記の例では、SCC(SECURITY CONTEXT CONSTRAINTS)が割り当てられたサービスアカウント“privsvcacct”を指定しています。インデントは以下の例に合わせてください。

```
～～抜粋開始～～
      spec:
        containers:
        - image: 192.168.100.102:5000/lweeapp:latest
          name: lweeapp
          ports:
          - containerPort: 81
            protocol: TCP
          resources: {}
          securityContext:
            privileged: true
        securityContext:
          runAsUser: 0
        serviceAccount: privsvcacct
        serviceAccountName: privsvcacct
    test: false
～～抜粋終了～～
```

   ServiceのexternalIPsとPortsの設定を行います。

```
～～抜粋開始～～
  kind: Service
```

```
  metadata:
    annotations:
      openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
      app: lweeapp
    name: lweeapp
  spec:
    type: ClusterIP
    externalIPs:
    - <IPアドレス>       …(*)
    ports:
    - name: 8081-tcp
      port: 8081
      protocol: TCP
      targetPort: 81
～～抜粋終了～～
```

(*) <IPアドレス>には、OpenShiftシステムの外部ネットワークに公開するIPアドレスを設定してください。

5. 4で編集したテンプレートを使用してコンテナを起動します。

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

6. コンテナが起動したことを確認します。

```
# oc get all
NAME                              REVISION   DESIRED   CURRENT   TRIGGERED BY
deploymentconfigs/lweeapp         1          1         1         config

NAME                 DOCKER REPO                              TAGS      UPDATED
imagestreams/lweeapp 192.168.100.102:5000/lweeapp             latest    10 minutes ago

NAME                         READY     STATUS    RESTARTS   AGE
po/lweeapp-1-xxxxx           1/1       Running   0          5m

NAME                  DESIRED   CURRENT   READY     AGE
rc/lweeapp-1          1         1         1         5m

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP      PORT(S)    AGE
svc/lweeapp          ClusterIP   xxx.xxx.xxx.xxx  yyy.yyy.yyy.yyy  8081/TCP   5m
```

7. Web連携機能を起動します。

```
[root@lweeapp-1-xxxxx /]# oc rsh po/lweeapp-1-xxxxx bash
[root@ lweeapp-1-xxxxx work]# /opt/FJSVisje6/var/pcmi/isje6/FJSVpcmi start
[root@ lweeapp-1-xxxxx work]# /opt/FJSVisje6/glassfish/bin/asadmin start-domain
[root@ lweeapp-1-xxxxx work]# echo AS_ADMIN_PASSWORD=adminadmin > /work/passwordfile
[root@ lweeapp-1-xxxxx work]# /opt/FJSVlw-gw/bin/lw-gwsetenv.sh 1 3 12011 admin /work/passwordfile
```

8. Web連携機能を確認します。

Internet Explorerを使用して、List Worksのログイン画面を開き、List Worksサーバのユーザ名、パスワード、ホスト名でログインできることを確認します。