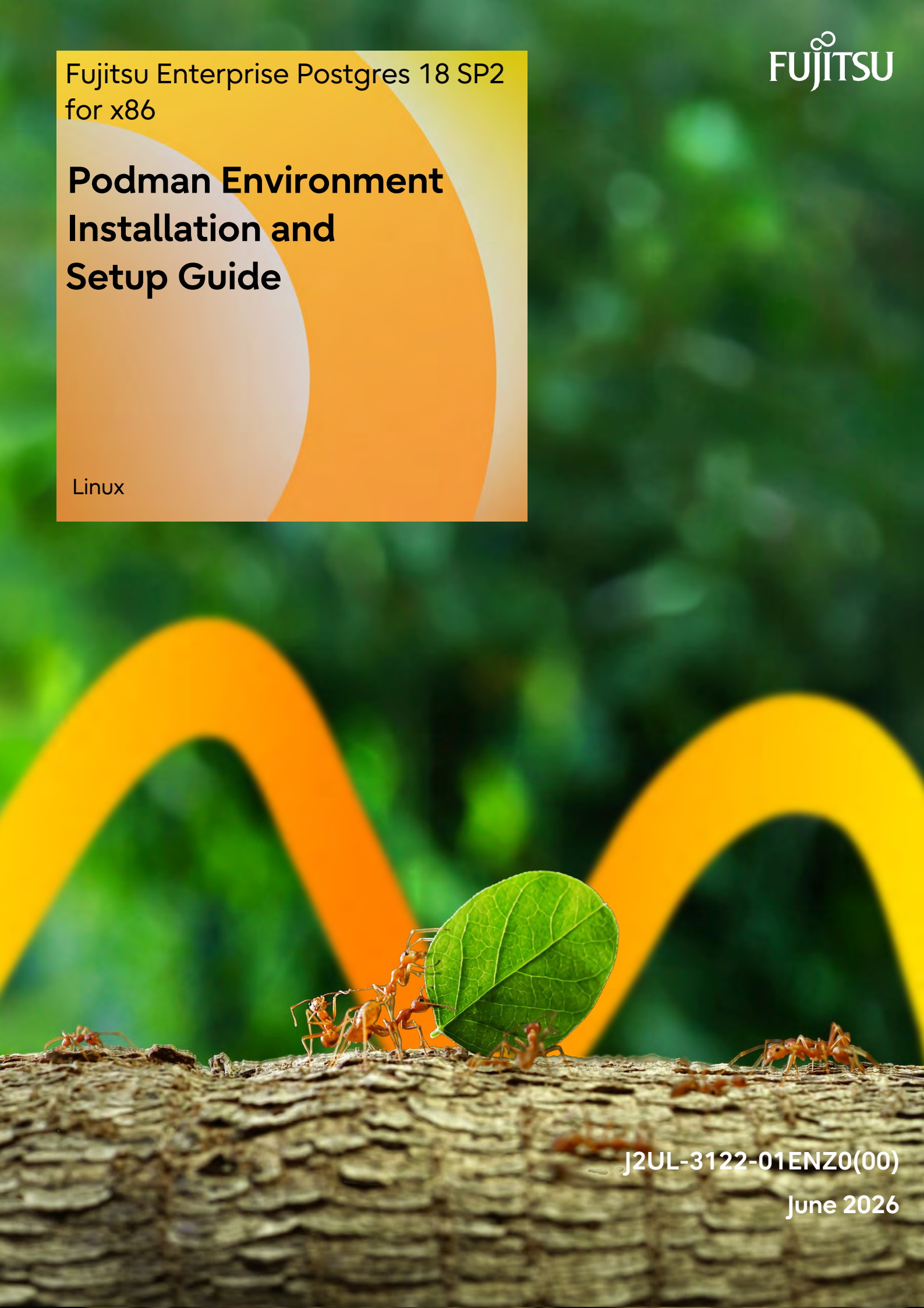
Fujitsu Enterprise Postgres 18 SP2 for x86

Podman Environment Installation and Setup Guide

Linux



J2UL-3122-01ENZ0(00)

June 2026

Preface

Purpose of this document

This document describes the introduction procedure for using Fujitsu Enterprise Postgres on Podman.

Intended readers

To read this document, the following knowledge is required.

- Fujitsu Enterprise Postgres
- Podman
- Linux

Abbreviations

The following abbreviations are used in this manual:

Full Name	Abbreviations
Red Hat(R) Enterprise Linux(R) 8	RHEL8
Red Hat(R) Enterprise Linux(R) 9	RHEL9
Universal Base Image	UBI

Trademarks

- Linux is a registered trademark or trademark of Mr. Linus Torvalds in the U.S. and other countries.
- Red Hat and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

Other product and company names mentioned in this manual are the trademarks or registered trademarks of their respective owners.

Export restrictions

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Issue date and version

```
Edition 1.0: June 2026
```

Copyright

Contents

Chapter 1 Overview

1.1 Available Features of Fujitsu Enterprise Postgres

The availability of Fujitsu Enterprise Postgres features when used on Podman is shown below.

Features	Availability
WebAdmin	Y
Database multiplexing	N
Multi-master replication	N
Cluster system using Patroni	Y
Failover (integration with PRIMECLUSTER)	N
Backup/recovery using user exits	Y (*1)
Connection Manager	Y
Application driver (JDBC, ODBC, .NET Data Provider, libpq, Embedded SQL (C/COBOL), psycopg, Go language	Y (*2)
Features compatible with Oracle databases	Y
Transparent data encryption	Y
Transparent data encryption using a key management system	Y
Data masking	Y
Audit log	Y
Confidentiality management	Y
Privileged Access Management	Y
Policy-based Login Security	Y
Knowledge data management	Y
In-memory feature	Y
High-speed data load	Y
Global Meta Cache	Y
Local Meta Cache Limit	Y
Database Monitoring with Amazon CloudWatch	Y
Autoscale of Replica Servers through Amazon EC2 Auto Scaling Integration	N
Enhanced query plan stability	Y

Y: Available

N: Unavailable

*1: The advanced copy function of the ETERNUS disk array is not available.

*2: Only Linux clients are supported.

1.2 Operating Environment

It can operate in the following environment.

Supported Products

Fujitsu Enterprise Postgres 18 SP2 or later

Supported OS

- Host OS: RHEL8, Container Base OS: UBI8 (ubi8/ubi)
- Host OS: RHEL9, Container Base OS: UBI8 (ubi9/ubi)

Related Software

Podman 5.6.0

1.3 Scope of Support

Fujitsu Enterprise Postgres is supported to start and operate normally on Podman, but container abnormal termination and automatic restart caused by Podman are not supported.

Chapter 2 Installation/Settings

2.1 Installation Procedure

Prerequisites

- Podman is installed
- RHEL subscription registered on the host OS

Installation Procedure

1. Prepare the following scripts used to build the container image and deploy them to the directory where the build will be executed.

   For script samples, refer to "2.1.1 Samples".

   - Dockerfile: Dockerfile to create a server image or client image
   - rpmlist_ubi.txt: List of required packages to be installed from UBI packages
   - rpmlist_rhel.txt: List of required packages to be installed from RHEL packages
   - Entrypoint: Entrypoint script executed when the container starts

2. Create a cdrom directly under the build context and insert and mount the Fujitsu Enterprise Postgres "Server Program" or "Client Program" DVD into the DVD drive. For details, refer to "Installation" in the Installation and Setup Guide for Server and Installation and Setup Guide for Client.

```
./ (build context)
├── Dockerfile
├── rpmlist_ubi.txt
├── rpmlist_rhel.txt
├── entrypoint.sh
└── cdrom/
```

2.1.1 Samples

2.1.1.1 Sample of Database Server Image

Below is a sample script for creating a server image on RHEL9. Edit the content according to your environment.

When enabling extensions, you will need to set up. Add the process to copy the necessary OSS files from the OSS directory. In the sample file, an example of enabling pgvector in "4. Create postgres user and setting" is described.

For samples using RHEL8, refer to "A.1 Sample of Database Server Image".

Dockerfile

```
FROM registry.access.redhat.com/ubi9/ubi
LABEL maintainer="Fujitsu Enterprise Postgres" \
      description="Docker sample for build FEP"
# ============================================================
# 0. Parameters edit by user
# ============================================================
ARG LOCALE_LANG=ja_JP
ARG LOCALE_CHARSET=UTF-8
ARG TIMEZONE=Asia/Tokyo
ARG PGUSER=fsepuser
ARG PGPASSWORD=postgres
ARG PGDATA=/home/fsepuser/primary
ARG WEBADMIN_PORT=27515
ARG PGPORT=27500
```

```
ARG RHEL=RHEL9
ARG INSTALL_DIR=/opt/fsepv<x>server64
# ============================================================
ARG DNF_OPTS_UBI="--disablerepo=* --enablerepo=ubi-9-baseos-rpms --enablerepo=ubi-9-appstream-rpms --
enablerepo=ubi-9-codeready-builder-rpms"
ARG DNF_OPTS_RHEL="--disablerepo=* --enablerepo=rhel-9-for-x86_64-baseos-rpms --enablerepo=rhel-9-
for-x86_64-appstream-rpms --enablerepo=codeready-builder-for-rhel-9-x86_64-rpms"
# ============================================================
# 1. OS Update & Locale Settings
# ============================================================
RUN dnf -y update ${DNF_OPTS_UBI} && \
    dnf -y clean all

RUN dnf -y install ${DNF_OPTS_UBI} glibc-locale-source && \
    localedef -i ${LOCALE_LANG} -c -f ${LOCALE_CHARSET} \
    -A /usr/share/locale/locale.alias ${LOCALE_LANG}.${LOCALE_CHARSET} && \
    echo "LANG=${LOCALE_LANG}.${LOCALE_CHARSET}" > /etc/locale.conf && \
    ln -sf /usr/share/zoneinfo/${TIMEZONE} /etc/localtime

ENV LANG=${LOCALE_LANG}.${LOCALE_CHARSET} \
    TZ=${TIMEZONE}
# ============================================================
# 2. Install requirement packages
# ============================================================
RUN mkdir -p /fepdb

COPY ./rpmlist_ubi.txt ./rpmlist_rhel.txt /fepdb/

RUN dnf -y install ${DNF_OPTS_UBI} yum-utils

RUN cat /fepdb/rpmlist_ubi.txt | \
    xargs dnf -y install ${DNF_OPTS_UBI} && \
    dnf -y clean all

RUN cat /fepdb/rpmlist_rhel.txt | \
    xargs dnf -y install ${DNF_OPTS_RHEL} && \
    dnf -y clean all
# ============================================================
# 3. Install FEP Server
# ============================================================
COPY ./cdrom/ /fepdb/cdrom/

RUN echo "==> Installing" && \
    rpm -ivh /fepdb/cdrom/SERVER/Linux/packages/${RHEL}/FJSVfsep-server-*.rpm
# ============================================================
# 4. Create postgres user and setting
# ============================================================
RUN useradd -m -s /bin/bash ${PGUSER} && \
    echo "${PGUSER}:${PGPASSWORD}" | chpasswd && \
    echo "${PGUSER} ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/${PGUSER} && \
    cp -r ${INSTALL_DIR}/OSS/pgvector/* ${INSTALL_DIR}

ENV PATH=${INSTALL_DIR}/bin:${PATH} \
    MANPATH=${INSTALL_DIR}/share/man:${MANPATH} \
    PGDATA=${PGDATA} \
    PGPORT=${PGPORT} \
    PGENCODING=${LOCALE_CHARSET} \
    PGUSER=${PGUSER}
# ============================================================
# 5. Entrypoint
# ============================================================
COPY ./entrypoint.sh /home/${PGUSER}/entrypoint.sh
RUN chmod +x /home/${PGUSER}/entrypoint.sh
```

```
USER ${PGUSER}
WORKDIR /home/${PGUSER}/

EXPOSE ${WEBADMIN_PORT} ${PGPORT}

ENTRYPOINT ["/bin/bash","-c","/home/${PGUSER}/entrypoint.sh"]
```

UBI Packages

Below is a list of UBI packages. If there are additional UBI packages to install, add them to this sample. For package details, refer to "Operating Environment" in the in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server.

rpmlist_ubi.txt

```
alsa-lib.x86_64
audit-libs.x86_64
cyrus-sasl-lib.x86_64
glibc.x86_64
glibc.i686
libnsl2.x86_64
libicu.x86_64
libgcc.x86_64
libmemcached-awesome.x86_64
libstdc++.x86_64
libtool-ltdl.x86_64
libzstd.x86_64
lz4-libs.x86_64
ncurses-libs.x86_64
net-tools.x86_64
nss-softokn-freebl.x86_64
pam.x86_64
perl-libs.x86_64
libselinux.x86_64
tcl.x86_64
unzip.x86_64
xz-libs.x86_64
zlib.x86_64
gdb.x86_64
rsync.x86_64
sudo.x86_64
liburing.x86_64
numactl-libs.x86_64
iputils
systemd
gcc
python3.11-devel
python3.12
```

Since the following packages cannot be installed with UBI, please install them using the RHEL repository.

rpmlist_rhel.txt

```
pcp-system-tools.x86_64
llvm-libs-17.0.6
sysstat.x86_64
```

Entrypoint

Below is an example of an Entrypoint. Using this example, the database can be started when the container is created and stopped when the container is stopped.

If you need to modify postgresql.conf, refer to the example below and edit the Entrypoint script. In the following sample, the port number set with PG_PORT is reflected in postgresql.conf.

entrypoint.sh

```
#!/bin/bash
# ============================================================
# Fujitsu Enterprise Postgres - Container Entrypoint
# ============================================================

# ============================================================
# 1. Init DB
# ============================================================
if [ ! -f "${PGDATA}/PG_VERSION" ]; then
    initdb -D "${PGDATA}" --encoding=${PGENCODING} --locale=${LANG} --data-checksums
    sed -i "s/^#*port\s*=.*/port = ${PGPORT}/" "${PGDATA}/postgresql.conf"
    sed -i "s/^#*listen_addresses\s*=.*/listen_addresses = '*'/" "${PGDATA}/postgresql.conf"
   echo "host    all            all            0.0.0.0/0             trust" >> "${PGDATA}/pg_hba.conf"
fi

# ============================================================
# 2. Start PostgreSQL
# ============================================================
echo "=== Starting PostgreSQL ==="
if [ -n "${PG_CONF_FILE}" ] && [ -f "${PG_CONF_FILE}" ]; then
    echo "=== Using external config: ${PG_CONF_FILE} ==="
    pg_ctl -D "${PGDATA}" start -o "-c config_file=${PG_CONF_FILE}"
else
    pg_ctl -D "${PGDATA}" start
fi

RETRIES=30
until pg_isready -q -p "${PGPORT}" || [ "${RETRIES}" -eq 0 ]; do
    RETRIES=$((RETRIES - 1))
    sleep 1
done

if [ "${RETRIES}" -eq 0 ]; then
    echo "ERROR: PostgreSQL did not become ready. Check logs at: ${PGDATA}/log/"
fi
echo "=== PostgreSQL is ready (port: ${PGPORT}) ==="

# ============================================================
# 3. Graceful Shutdown on Container Stop
#    podman stop sends SIGTERM -> pg_ctl stop -> container exits
# ============================================================
_term() {
    pg_ctl -D "${PGDATA}" stop -m fast
    exit 0
}
trap _term SIGTERM

while true; do
    sleep 600 &
    wait $!
done
```

2.1.1.2 Sample of Client Image

Below is a sample script for creating a client image on RHEL9. Edit the content according to your environment. The installation of necessary packages for the client is done in "3. Install relational package". Install the necessary packages.

For samples using RHEL8, refer to "A.2 Sample of Client Image".

Dockerfile

```
FROM registry.access.redhat.com/ubi9/ubi
LABEL maintainer="Fujitsu Enterprise Postgres" \
      description="Docker sample for build FEP Client"
# ============================================================
# 0. Parameters edit by user
# ============================================================
ARG LOCALE_LANG=ja_JP
ARG LOCALE_CHARSET=UTF-8
ARG TIMEZONE=Asia/Tokyo
ARG PGUSER=fsepuser
ARG INSTALL_DIR=/opt/fsepv<x>client64
ARG RHEL=RHEL9
# ============================================================
ARG DNF_OPTS="--disablerepo=* --enablerepo=ubi-9-baseos-rpms --enablerepo=ubi-9-appstream-rpms --
enablerepo=ubi-9-codeready-builder-rpms"
# ============================================================
# 1. OS Update & Locale Settings
# ============================================================
RUN dnf -y update ${DNF_OPTS} && \
    dnf -y clean all

RUN dnf -y install ${DNF_OPTS} glibc-locale-source && \
    localedef -i ${LOCALE_LANG} -c -f ${LOCALE_CHARSET} \
    -A /usr/share/locale/locale.alias ${LOCALE_LANG}.${LOCALE_CHARSET} && \
    echo "LANG=${LOCALE_LANG}.${LOCALE_CHARSET}" > /etc/locale.conf && \
    ln -sf /usr/share/zoneinfo/${TIMEZONE} /etc/localtime

ENV LANG=${LOCALE_LANG}.${LOCALE_CHARSET} \
    TZ=${TIMEZONE}
# ============================================================
# 2. Install FEP Client requirement
# ============================================================
RUN mkdir -p /fepdb
COPY ./rpmlist_ubi.txt /fepdb/

RUN dnf -y install ${DNF_OPTS} yum-utils

RUN cat /fepdb/rpmlist_ubi.txt | \
    xargs dnf -y --allowerasing install ${DNF_OPTS} && \
    dnf -y clean all
# ============================================================
# 3. Install relational package
# ============================================================
#RUN dnf -y install ${DNF_OPTS} \
#    java-17-openjdk-devel \
#    dotnet-sdk-8.0 \
#    gcc \
#    python3.11devel\
#   && dnf -y clean all
# ============================================================
# 4. Install FEP Client
# ============================================================
COPY ./cdrom /fepdb/cdrom

RUN echo "==> Installing" && \
    rpm -ivh /fepdb/cdrom/CLIENT/Linux/packages/${RHEL}/FJSVfsep-client-*.rpm
# ============================================================
# 5. Create postgres user
# ============================================================
RUN useradd -m -s /bin/bash ${PGUSER}
```

```
ENV PATH=${INSTALL_DIR}/bin:${PATH} \
    LD_LIBRARY_PATH=${INSTALL_DIR}/lib:${LD_LIBRARY_PATH} \
    MANPATH=${INSTALL_DIR}/share/man:${MANPATH} \
    PGLOCALEDIR=${INSTALL_DIR}/share/locale \
    JAVA_HOME=/usr/lib/jvm/java-17-openjdk \
    PGUSER=${PGUSER} \
    PG_ENCODING=${LOCALE_CHARSET}

# ============================================================
# 6. Entrypoint
# ============================================================
COPY ./entrypoint.sh /home/${PGUSER}/entrypoint.sh
RUN chmod +x /home/${PGUSER}/entrypoint.sh

USER ${PGUSER}

ENTRYPOINT ["/bin/bash", "-c", "/home/${PGUSER}/entrypoint.sh"]
```

UBI Packages

Below is a list of UBI packages. If there are additional UBI packages to install, add them to this sample. For package details, refer to "Operating Environment" in the in the Fujitsu Enterprise Postgres Installation and Setup Guide for Client.

rpmlist_ubi.txt

```
bzip2-libs
glibc
glibc.i686
go
libcurl
libnsl2
libgcc
libmemcached-awesome
libstdc++
libtool-ltdl
libzstd
lz4-libs
ncurses-libs
nss-softokn-freebl
rsync
sudo
unixODBC
unzip
xz-libs
zlib
systemd
iputils
```

Entrypoint

Below is an example of an Entrypoint. This is a sample for processing a database server. Please describe how to run the client application.

entrypoint.sh

```
#!/bin/bash
# Connection settings (can be overridden by environment variables)
# NOTE: DB_HOST must be set to the FEP server hostname or IP.
export PGHOST="${DB_HOST:?ERROR:Specify the FEP server hostname or IP.}"
export PGPORT="${DB_PORT:-27500}"
export PGUSER="${DB_USER:-postgres}"
export PGDATABASE="${DB_NAME:-postgres}"
export PGPASSWORD="${DB_PASSWORD:-postgres}"

echo "FEP Client: Connecting to ${PGHOST}:${PGPORT}"
```

```
# Connect to server and show version and table list
psql <<'SQL'
SELECT version();
\dt
SQL

echo "Done"
```

2.2 Creating a Container Image

create a container image using the podman command. Use -t option to specify the image name to be created. For the context (./), specify the directory where the materials used for installation are located. Environment variables specified in the Dockerfile can be overridden during build time with the --build-arg parameter. The following example shows how to rewrite the variable "PGPASSWORD".

```
$ podman build --build-arg PGPASSWORD=postgres -t fep-container ./
```

After the build, use the following command to check the Podman image you created.

```
$ podman images
REPOSITORY                          TAG         IMAGE ID      CREATED       SIZE
localhost/fep-container             latest      358b46467161  3 hours ago   9.01 GB
```

2.3 Setup

Explain about building Podman container images.

2.3.1 Creating Persistent Volumes

Data saved in the container's internal file system is not retained when the container is stopped or recreated. For data that needs to be persistently stored, such as databases and configuration files, use persistent volumes. Recommend storing data under PG_DATA in a persistent volume. If necessary, persist WAL, Tablespace, logs, etc., outside of PG_DATA. For information on how to mount, refer to "2.4.1 Starting a Container".

Persistent volumes are created using the following command.

```
$ podman volume create fep-pgdata
$ podman volume ls
DRIVER      VOLUME NAME
local       fep-pgdata
```

2.3.2 Fixed IP Address

In a Podman environment, an IP address is dynamically assigned each time a container is launched. When using Fujitsu Enterprise Postgres and utilizing functions that enable communication between containers, a Podman user-defined network is created and a static IP address is assigned to the container to fix the IP address. For information on how to assign to a container, refer to "2.4.1 Starting a Container".

Create a subnet using the following command.

```
$ podman network create --subnet SUBNET fep-net
fep-net
# Network check
$ podman network ls
NETWORK ID    NAME        DRIVER
9f580f168de4  fep-net     bridge
```

2.3.3 When using Transparent Data Encryption

When using transparent data encryption, key information required for encryption and decryption should be stored in a persistent volume. Additionally, when using a Key Management Service, key management system connection information and files should also be stored in a persistent volume.

If you enable automatic keystore opening, you need to fix the hostname because it changes every time the container starts. Fixing the hostname, similar to fixing the IP address, is specified using options in the podman run command. Refer to "2.4.1 Starting a Container".

2.3.4 When using Connection Manager

In a Podman environment, it is possible to build and launch Connection Manager as an independent container. Please note that when Connection Manager is operated as a container, the liveness monitoring function will monitor the liveness of the container itself. We recommend adopting Connection Manager for the purpose of utilizing features such as connection aggregation and transparent connections.

For Connection Manager setup, refer to the "Fujitsu Enterprise Postgres Connection Manager User's Guide".

In a Podman environment, if you start a container with the default settings, the container's IP address will be dynamically assigned each time it starts. Connection Manager requires a static IP address. Therefore, when using Connection Manager in a container, you need to create a Podman user-defined network and assign static IP addresses to the Connection Manager container and the server container before starting them. For information on how to assign static IP addresses to containers, refer to "2.4.1 Starting a Container".

2.3.5 About Logs

It is recommended to output server logs, audit logs, and WebAdmin logs as files and store them in persistent volumes. By storing these logs in persistent volumes and mounting them using the -v option of the podman run command as specified below, logs can be continuously retained even after restarts or recreations.

Example: For server logs

```
-v Mount destination for volume: /opt/fsepv<x>server64/
```

"<*x*>" indicates the product version.

Point

For audit logs and WebAdmin log storage locations, please refer to the following:

- Audit logs: "pgaudit Configuration File" in the Fujitsu Enterprise Postgres Security Operations Guide
- WebAdmin: "Creating Instances" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server

2.4 Starting a Container

Explain how to start the container.

For information on setting up Fujitsu Enterprise Postgres after container startup, refer to the Fujitsu Enterprise Postgres Installation and Setup Guide for Server.

2.4.1 Starting a Container

Start Podman with a pre-built container image using the following command.

Refer to the Podman documentation for details on the options.

```
$ podman run -it -p 27500:27500 -v fep-pgdata:/home/postgres/primary --name fep-container localhost/
fep-container
```

When using WebAdmin

When using WebAdmin with Podman, you need to start systemd when launching the container.

Start it with the following command. Refer to the Podman documentation for details on the options.

```
$ podman run -it -d \
--systemd=true --entrypoint /sbin/init \
--user root -p 27500:27500 -p 27515:27515 \
-v fep-pgdata:/home/postgres/primary \
--ip IPaddress \
--network subnet-name \
--name fep-webadmin-container localhost/fep-container
```

Point

[Examples of options to specify]

- -p *Host port*:*Container port*
  Specifies the port number to forward.
- -v *Mount persistent volume*
- --name *Container name*
- --ip *IP address to fix*
- --network *Subnet name*
- --hostname *Host name*

2.4.2 When Using Model Management in the Database of Knowledge Data Management Feature

For the Triton Inference Server container to be able to reference files written to the model repository by the Fujitsu Enterprise Postgres container, both containers must mount the same directory on the host side.

Below is an example of a startup command.

When starting the Fujitsu Enterprise Postgres server

```
$ podman run -v /host/model-repository:/var/fep/model-repository:Z
```

When the Triton container starts

```
$ podman run -v /host/model-repository:/models:Z
```

The container may not be able to access the mounted host directory. If necessary, specify ":z" (shared by multiple containers) or ":Z" (container-specific) in the -v option.

Also, set privileges so that Triton can read files written by the Fujitsu Enterprise Postgres server.

See

For model management in the database of the knowledge data management feature, please refer to "Model Management in the Database" in the Fujitsu Enterprise Postgres Knowledge Data Management Feature User's Guide.

2.5 Connecting to Container

You can connect to the Podman container as the postgres user with the following command.

```
$ podman exec -it -u postgres fep-container bash
```

Also, you can execute psql commands from the client as follows.

```
$ podman exec -u fsepuser <client-container> psql -h <server-container-ip> -U postgres -d postgres
```

For more details on how to use it, please refer to the Podman documentation.

2.6 Stopping and Deleting a Container

To stop the container, use the following command. When stopping the container, the database will also be stopped.

```
$ podman stop fep-container
```

Also, to delete a container, use the following command. You need to stop the container before deleting it.

```
$ podman rm fep-container
```

Appendix A Sample File

Here is an example of creating a container image.

A.1 Sample of Database Server Image

Here is an example of creating a server. This is an example when using RHEL8.

Dockerfile

```
FROM registry.access.redhat.com/ubi8/ubi
LABEL maintainer="Fujitsu Enterprise Postgres" \
      description="Docker sample for build FEP"
# ============================================================
# 0. Parameters edit by user
# ============================================================
ARG LOCALE_LANG=ja_JP
ARG LOCALE_CHARSET=UTF-8
ARG TIMEZONE=Asia/Tokyo
ARG PGUSER=fsepuser
ARG PGPASSWORD=postgres
ARG PGDATA=/home/fsepuser/primary
ARG WEBADMIN_PORT=27515
ARG PGPORT=27500
ARG RHEL=RHEL8
ARG INSTALL_DIR=/opt/fsepv<x>server64
# ============================================================
ARG DNF_OPTS_UBI="--disablerepo=* --enablerepo=ubi-8-baseos-rpms --enablerepo=ubi-8-appstream-rpms --
enablerepo=ubi-8-codeready-builder-rpms"
ARG DNF_OPTS_RHEL="--disablerepo=* --enablerepo=rhel-8-for-x86_64-baseos-rpms --enablerepo=rhel-8-
for-x86_64-appstream-rpms --enablerepo=codeready-builder-for-rhel-8-x86_64-rpms"
# ============================================================
# 1. OS Update & Locale Settings
# ============================================================
RUN dnf -y update ${DNF_OPTS_UBI} && \
    dnf -y clean all

RUN dnf -y install ${DNF_OPTS_UBI} glibc-locale-source && \
    localedef -i ${LOCALE_LANG} -c -f ${LOCALE_CHARSET} \
    -A /usr/share/locale/locale.alias ${LOCALE_LANG}.${LOCALE_CHARSET} && \
    echo "LANG=${LOCALE_LANG}.${LOCALE_CHARSET}" > /etc/locale.conf && \
    ln -sf /usr/share/zoneinfo/${TIMEZONE} /etc/localtime

ENV LANG=${LOCALE_LANG}.${LOCALE_CHARSET} \
    TZ=${TIMEZONE}
# ============================================================
# 2. Install requirement packages
# ============================================================
RUN mkdir -p /fepdb

COPY ./rpmlist_ubi.txt ./rpmlist_rhel.txt /fepdb/

RUN dnf -y install ${DNF_OPTS_UBI} yum-utils

RUN cat /fepdb/rpmlist_ubi.txt | \
    xargs dnf -y install ${DNF_OPTS_UBI} && \
    dnf -y clean all

RUN cat /fepdb/rpmlist_rhel.txt | \
    xargs dnf -y install ${DNF_OPTS_RHEL} && \
    dnf -y clean all
# ============================================================
```

```
# 3. Install FEP Server
# ============================================================
COPY ./cdrom/ /fepdb/cdrom/

RUN echo "==> Installing" && \
    rpm -ivh /fepdb/cdrom/SERVER/Linux/packages/${RHEL}/FJSVfsep-server-*.rpm
# ============================================================
# 4. Create postgres user and setting
# ============================================================
RUN useradd -m -s /bin/bash ${PGUSER} && \
    echo "${PGUSER}:${PGPASSWORD}" | chpasswd && \
    echo "${PGUSER} ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/${PGUSER} && \
    cp -r ${INSTALL_DIR}/OSS/pgvector/* ${INSTALL_DIR}

ENV PATH=${INSTALL_DIR}/bin:${PATH} \
    MANPATH=${INSTALL_DIR}/share/man:${MANPATH} \
    PGDATA=${PGDATA} \
    PGPORT=${PGPORT} \
    PGENCODING=${LOCALE_CHARSET} \
    PGUSER=${PGUSER}
# ============================================================
# 5. Entrypoint
# ============================================================
COPY ./entrypoint.sh /home/${PGUSER}/entrypoint.sh
RUN chmod +x /home/${PGUSER}/entrypoint.sh

USER ${PGUSER}
WORKDIR /home/${PGUSER}/

EXPOSE ${WEBADMIN_PORT} ${PGPORT}

ENTRYPOINT ["/bin/bash","-c","/home/${PGUSER}/entrypoint.sh"]
```

rpmlist_ubi.txt

```
alsa-lib.x86_64
audit-libs.x86_64
cyrus-sasl-lib.x86_64
glibc.x86_64
glibc.i686
libnsl2.x86_64
libicu.x86_64
libgcc.x86_64
libstdc++.x86_64
libtool-ltdl.x86_64
libzstd.x86_64
lz4-libs.x86_64
ncurses-libs.x86_64
net-tools.x86_64
nss-softokn-freebl.x86_64
pam.x86_64
perl-libs.x86_64
libselinux.x86_64
tcl.x86_64
unzip.x86_64
xz-libs.x86_64
zlib.x86_64
gdb.x86_64
rsync.x86_64
sudo.x86_64
numactl-libs.x86_64
iputils
```

```
systemd
python3.12
```

rpmlist_rhel.txt

```
pcp-system-tools.x86_64
llvm-libs-17.0.6
sysstat.x86_64
```

A.2 Sample of Client Image

Here is an example of creating a client. This is an example when using RHEL8.

Dockerfile

```
FROM registry.access.redhat.com/ubi8/ubi
LABEL maintainer="Fujitsu Enterprise Postgres" \
      description="Docker sample for build FEP Client"
# ============================================================
# 0. Parameters edit by user
# ============================================================
ARG LOCALE_LANG=ja_JP
ARG LOCALE_CHARSET=UTF-8
ARG TIMEZONE=Asia/Tokyo
ARG PGUSER=fsepuser
ARG INSTALL_DIR=/opt/fsepv<x>client64
ARG RHEL=RHEL8
# ============================================================
ARG DNF_OPTS="--disablerepo=* --enablerepo=ubi-8-baseos-rpms --enablerepo=ubi-8-appstream-rpms --
enablerepo=ubi-8-codeready-builder-rpms"
# ============================================================
# 1. OS Update & Locale Settings
# ============================================================
RUN dnf -y update ${DNF_OPTS} && \
    dnf -y clean all

RUN dnf -y install ${DNF_OPTS} glibc-locale-source && \
    localedef -i ${LOCALE_LANG} -c -f ${LOCALE_CHARSET} \
    -A /usr/share/locale/locale.alias ${LOCALE_LANG}.${LOCALE_CHARSET} && \
    echo "LANG=${LOCALE_LANG}.${LOCALE_CHARSET}" > /etc/locale.conf && \
    ln -sf /usr/share/zoneinfo/${TIMEZONE} /etc/localtime

ENV LANG=${LOCALE_LANG}.${LOCALE_CHARSET} \
    TZ=${TIMEZONE}
# ============================================================
# 2. Install FEP Client requirement
# ============================================================
RUN mkdir -p /fepdb
COPY ./rpmlist_ubi.txt /fepdb/

RUN dnf -y install ${DNF_OPTS} yum-utils

RUN cat /fepdb/rpmlist_ubi.txt | \
    xargs dnf -y --allowerasing install ${DNF_OPTS} && \
    dnf -y clean all
# ============================================================
# 3. Install relational package
# ============================================================
#RUN dnf -y install ${DNF_OPTS} \
#    java-17-openjdk-devel \
#    dotnet-sdk-6.0 \
```

```
#    gcc \
#    python38-devel \
#   && dnf -y clean all
# ============================================================
# 4. Install FEP Client
# ============================================================
COPY ./cdrom /fepdb/cdrom

RUN echo "==> Installing" && \
    rpm -ivh /fepdb/cdrom/CLIENT/Linux/packages/${RHEL}/FJSVfsep-client-*.rpm
# ============================================================
# 5. Create postgres user
# ============================================================
RUN useradd -m -s /bin/bash ${PGUSER}

ENV PATH=${INSTALL_DIR}/bin:${PATH} \
    LD_LIBRARY_PATH=${INSTALL_DIR}/lib:${LD_LIBRARY_PATH} \
    MANPATH=${INSTALL_DIR}/share/man:${MANPATH} \
    PGLOCALEDIR=${INSTALL_DIR}/share/locale \
    JAVA_HOME=/usr/lib/jvm/java-17-openjdk \
    PGUSER=${PGUSER} \
    PG_ENCODING=${LOCALE_CHARSET}
# ============================================================
# 6. Entrypoint
# ============================================================
COPY ./entrypoint.sh /home/${PGUSER}/entrypoint.sh
RUN chmod +x /home/${PGUSER}/entrypoint.sh

USER ${PGUSER}

ENTRYPOINT ["/bin/bash", "-c", "/home/${PGUSER}/entrypoint.sh"]
```

rpmlist_ubi.txt

```
alsa-lib.x86_64
audit-libs.x86_64
cyrus-sasl-lib.x86_64
glibc.x86_64
glibc.i686
libnsl2.x86_64
libicu.x86_64
libgcc.x86_64
libmemcached.x86_64
libstdc++.x86_64
libtool-ltdl.x86_64
libzstd.x86_64
lz4-libs.x86_64
ncurses-libs.x86_64
net-tools.x86_64
nss-softokn-freebl.x86_64
pam.x86_64
perl-libs.x86_64
libselinux.x86_64
tcl.x86_64
unzip.x86_64
xz-libs.x86_64
zlib.x86_64
gdb.x86_64
rsync.x86_64
sudo.x86_64
liburing.x86_64
numactl-libs.x86_64
```