Progress®



Run PAS for OpenEdge in a Docker Container

OpenEdge®

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

July 2021

Product version: Progress OpenEdge 12.4

Updated: 2021/06/30

Table of Contents

Preface....7

Learn about PAS for OpenEdge in a Docker container....9

Main steps: Using deploy scripts....11

Prerequisites to use the Docker image....13

Download the PAS for OpenEdge Docker image....15

Contents of a PAS for OpenEdge Docker image....16

Generate a license file....19

Generate a ZIP file of your ABL application....21

Ensure that the JDK image is available....23

Deploy applications using scripts....25

Available configuration and deployment options....25

Sample configuration for different deployment modes....27

Deploy and verify an ABL application....31

Other options for deployment....33

Upgrade a PAS for OpenEdge image in a container....37

Troubleshooting....39

Preface

Purpose

This manual describes how to use Docker container technology with Progress Application Server for OpenEdge (PAS for OpenEdge). It describes how to create and configure the Docker container for PAS for OpenEdge.

Audience

This manual is intended for ABL developers and OpenEdge system administrators who are familiar with PAS for OpenEdge. Familiarity with Docker technology is beneficial, but not required to understand the information in this manual. Additional learning sources are referenced throughout this manual.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

1

Learn about PAS for OpenEdge in a Docker container

Progress Application Server (PAS) for OpenEdge is available as a Docker image where applications can be developed, deployed, and run in an adaptive and isolated environment.

About Docker containers

Docker is a technology that enables developers to develop, deploy, and run an application in a stand-alone, virtualized environment called a container. Containers are a lightweight, standardized, and secure technique for executing software in an isolated environment. Using containers to isolate applications allows for faster and easier development and deployment.

As a developer, Docker containers enable you to deploy and upgrade your applications in a unified and simple process. Containers can automatically be created and destroyed to meet demand, providing scalability and flexibility. Containers also make it simple to use your applications across many machines and environments. These two factors allow you to develop software and deliver new features to customers with speed and agility.

Running PAS for OpenEdge in a Docker container is beneficial when you use PAS for OpenEdge and its deployed applications in a CI/CD pipeline where continuous updates and enhancements are required. When you run PAS for OpenEdge in a Docker container, both the application server instance and the applications deployed to the instance can be updated without affecting other running OpenEdge components (for example, the OpenEdge database).

Virtual machines offer a similar benefit as containers offer, but containers have the advantage because they do not virtualize hardware. Containers work on the application layer, and multiple containers share the same operating system kernel. This technology makes running multiple containers much more portable and efficient than running multiple virtual machines.

For more information about Docker, see What is a container? and the Docker Documentation.

Why use PAS for OpenEdge in a container?

Docker containers provide several advantages when used with PAS for OpenEdge:

- Docker containers help reduce overhead for packaging and deploying ABL applications, enabling you to streamline your application distribution processes. You only need to create a custom Docker container image with PAS for OpenEdge once, and then you can distribute that image and run as many containers as you need.
- When you upgrade your Docker containers for PAS for OpenEdge, you only need to upgrade one Docker image, and then you can easily redistribute the upgraded image.
- When you employ clustering tools such as Docker Swarm and Kubernetes, you can run multiple PAS for OpenEdge containers in a cluster, which provides high availability and elasticity.

Images and containers

A Docker image is the core component of a container. Docker builds images by reading instructions from a Dockerfile. Typically, a Docker container image includes its own execution environment with the code, run-time, system tools, libraries, and settings needed by the Docker daemon to instantiate the container and run an application. However, additional configuration and data can be injected into a container by using Docker volumes. A Docker container image can be used to run multiple containers, which provides flexibility and scalability. A Docker container is a runnable instance of a Docker image.

Sidecar design pattern

The sidecar design pattern uses two or more containers. Sidecar containers are attached to the primary application container, and they enhance and extend the primary application. Sidecar containers are created alongside the parent container and share the same life cycle. As an example, you could run your primary application in one container, and run your configuration settings for that application in a sidecar container. This structure enables you to be modular with your deployment, which means that you could swap out the sidecar container to change your configuration, or easily move your personalized configuration settings to another container.

Kubernetes

Kubernetes is an orchestration system that enables you to deploy and manage large clusters of containers automatically. With Kubernetes, applications scale dynamically and efficiently based on the level of demand. Progress recommends the Kubernetes platform if you use Docker containers. Kubernetes, see the Kubernetes website.

For details, see the following topics:

- Main steps: Using deploy scripts

Main steps: Using deploy scripts

To run the PAS for OpenEdge Docker container:

1. Check the Prerequisites to use the Docker image on page 13 and ensure that they are all accounted for.
2. Download the PAS for OpenEdge Docker image on page 15.
3. Generate a license file on page 19 and place it in the proper directory.
4. Generate a ZIP file of your ABL application on page 21.
5. Ensure that the JDK image is available on page 23 to your Docker environment.
6. Deploy applications using scripts on page 25 for the Docker, Docker Compose, and Minikube configurations. If you want to use a different configuration, then customize your deployment method, see Other options for deployment on page 33.
7. Deploy and verify your application, see Deploy and verify an ABL application on page 31.

2

Prerequisites to use the Docker image

This topic lists the prerequisites for using the PAS for OpenEdge Docker image.

System requirements

- Red Hat Enterprise Linux 7.5 (RHEL)
- Docker Enterprise Edition 18.03.1-ee-3 (Docker-EE)

Required artifacts

Before deploying applications, you must provide the following artifacts:

Generate a license file on page 19	Copy the generated Progress license file into *deploy folder*/license/progress.cfg.
Ensure that the JDK image is available on page 23	Ensure that the JDK image is available in the Docker environment.
Generate a ZIP file of your ABL application on page 21	Copy the created ABL application archive files into *deploy folder*/ablapps/*ABL app name*.zip.
Runtime configurations	Place the runtime.properties file into *deploy folder*/conf
Custom Fluent Bit configurations	If you are using Fluent Bit and want to customize the default configuration, then copy the configuration files into *deploy folder*/conf/logging.

Prerequisites for using the deploy scripts

- Java version 8 or later
- Apache Ant version 1.9.4 or later

3

Download the PAS for OpenEdge Docker image

This topic describes the download locations of the PAS for OpenEdge Docker image.

You can download the PAS for OpenEdge Docker image from the Docker Hub and the Progress Download Center.

Docker Hub

PAS for OpenEdge is available in the public Docker Hub repository. To download the image, use the following command from your Docker environment:

```
docker pull store/progresssoftware/pasoe:Openedge version
```

You must provide your login credentials on the Docker Hub. After logging in, accept the Terms and Conditions to pull the image.

To use the deploy scripts with the Docker Hub image, download them with the sample app on the Progress Community site.

Progress Download Center

You can also get the Docker image from the Progress Download Center, which is also known as the Electronic Software Distribution (ESD). To download the image:

1. Search for the `PROGRESS_PASOE_CONTAINER_IMAGE_12.2.0_LNX_64` archive file using the Product Search option.
2. Download the archive file.
3. Extract the archive file where the Docker environment is available. The archive file contains:
   - Deploy scripts inside the `deploy` folder
   - `NOTICE.txt`
   - `PROGRESS_PASOE_CONTAINER_IMAGE_12.2.0_LNX_64.tar.gz`
   - `README.txt`
4. Enter the following in the command line:

   `docker load -i` *`pasoe image archive name`*

   This command loads the `store/progresssoftware/pasoe:12.2.0` image.

For details, see the following topics:

- Contents of a PAS for OpenEdge Docker image

Contents of a PAS for OpenEdge Docker image

A PAS for OpenEdge container image contains the following:

- The container operating system, which is CentOS Linux Release 7.6.1810 (Core)
- PAS for OpenEdge, which is located in the `/psc/dlc` directory
- The starter scripts, which are located in the `/deploy/scripts` directory
- Fluent Bit Version 0.14.5
  - The runnable instance is located in the `/usr/local/bin/fluent-bit` directory.
  - The configuration files are located in the `/etc/fluent-bit` directory.

The following environment variables are in the PAS for OpenEdge Docker image:

`DLC`	Defines the path of the PAS for OpenEdge installation as `/psc/dlc`.
`WRKDIR`	Defines the working directory for your PAS for OpenEdge installation as `/psc/wrk`.
`JAVA_HOME`	Defines the Java path for your PAS for OpenEdge installation as `/usr/java`.

Deploy scripts directory structure

The following three files are located in the `deploy` directory of the provided deploy scripts:

`build.xml`	A build script to create the image
`config.properties`	A configuration file to define the properties of the image
`Dockerfile`	A Docker-specific set of configurations to define the properties of the image

In addition to these three files, there are a number of directories:

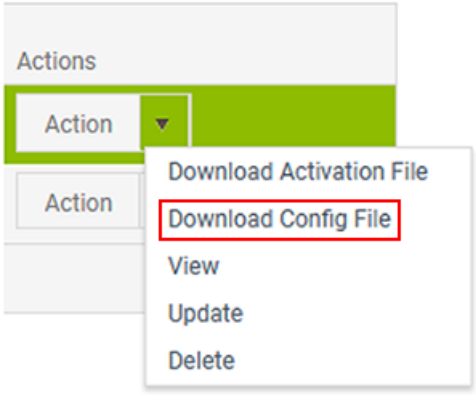
- `ablapps`—The location where you should provide your ABL application
- `conf`—Contains a logging folder and the `runtime.properties` file
- `lib`—Contains the `ant-contrib-0.6.jar` file
- `license`—The location for your license
- `scripts`—Contains the provided build scripts for Docker, Docker Compose, and Minikube
- `tasks`—Contains common build tasks

4

Generate a license file

To generate the license file for the PAS for OpenEdge Docker container:

1. Log in to the CSS portal with your Progress account credentials.
2. Search for your account using your customer identification details (either customer number or name).
3. Navigate to the **Account Info** page.
4. Select **Account** > **Licenses** > **Download OpenEdge Activation Files**.
5. Download the `progress.cfg` file for PAS for OpenEdge:



6. Place the generated license file in the *`deploy folder`*`/license/progress.cfg` folder.

5

Generate a ZIP file of your ABL application

You can create ZIP file of an ABL application so it can be deployed in a Docker container. For more information about ABL applications, see ABL application architecture.

Directory structure of an ABL application

An ABL applicationhas the following directories:

- `conf`—ABL application-level configurations are defined in this directory. This directory includes files such as `startup.pf`, which provides any database details that are required.
- `openedge`—This directory contains common, compiled ABL code.
- `tlr`—This directory contains the run-time merge properties and the `openedge.properties.merge` file, which provides run-time merge property details.
- `webapps`—This directory contains the WebApp WAR files for ABL web applications.

Create an ABL application

To create an application archive that can be used with a PAS for OpenEdge container, add the following artifacts to the project. The artifacts are located in the Sample App directory on the Progress Community site. The artifacts copied are:

Artifacts	Description
`.scripts`	Copy this directory and its contents from the Sample App directory to the ABL application directory.
`conf`	Copy this directory and its contents from the Sample App directory to the ABL application directory.
`tlr`	Copy this directory and its contents from the Sample App directory to the ABL application directory.
`PASOEContent/build.properties`	Copy this from the `PASOEContent` directory inside Sample App to the `PASOEContent` directory in the ABL application directory.
`PASOEContent/build.xml`	Copy this from the `PASOEContent` directory inside Sample App to the `PASOEContent` directory in the ABL application directory.
`build.properties`	Copy this file from the Sample App directory to the ABL application directory.
`build.xml`	Copy this file from the Sample App directory to the ABL application directory.

After adding the artifacts, open Proenv and run `ant package` to build and package the project. An ABL application archive file is created with the ABL application directory structure. Within the `Output` directory, the `package-output`is created, where the archive file will be located:

sample_ABLApp`\output\package-output\`*sample_ABLApp.zip*

Note: You can generate an ABL archive in other ways and without adhering to the structure given in this topic. This topic describes the approach that Progress recommends.

6

Ensure that the JDK image is available

To run PAS for OpenEdge in a Docker container, JDK is required in `/usr/java`, inside the PAS for OpenEdge container.

Note: The **Product Availability Guide (PAG)** provides a list of all the supported JDK versions that can be used for the JDK image.

How to obtain the JDK Docker image

You can obtain the JDK Docker image in one of the following ways:

- Download a JDK Docker image from AdoptOpenJDK, Docker Hub, or any other site certified by Progress and listed in the Product Availability Guide. You must use a Linux 64 image. The JDK version required is `jdk-8u221-linux-x64.tar.gz` or later.
- Build a custom Docker image with a downloaded JDK:

Note: If you use an AdoptOpenJDK Docker image, then you can skip these steps.

1. Download the compressed version of JDK 8 for Linux x64. The filename may be in following pattern: `jdk-8u*-linux-x64.tar.gz`
2. Create a Dockerfile in the same folder as the downloaded archive file with following code, and build the image. Then, give the image a suitable name. You can edit the code to reflect the file name to match the downloaded `.tar` file:

```
FROM alpine:3.8

        ENV JAVA_PKG=jdk-8u*-linux-x64.tar.gz

        ADD $JAVA_PKG /usr/java/

        RUN export JAVA_DIR=$(ls -1 -d /usr/java/*) && \
        ln -s $JAVA_DIR /usr/java/latest && \
        ln -s $JAVA_DIR /usr/java/default
```

3. To build an image from the Dockerfile, run the following command in the same folder:

```
$ docker build -t imageName:tag
```

   Note the values for *imageName*:*tag* because you will use them in a later step.

7

Deploy applications using scripts

This topic explains how to deploy applications using the deploy scripts.

Progress provides deploy scripts for ease of use. The deploy scripts enable you to deploy applications in different deployment modes (Docker, Docker Compose, and Minikube). The scripts use the sidecar approach for deployment.

If you downloaded the image from the Progress Download Center, then the deploy scripts are provided by default in the deploy folder inside the archive file.

For details, see the following topics:

- Available configuration and deployment options

Available configuration and deployment options

This topic explains the configuration and deployment options for a PAS for OpenEdge container.

Docker

The Docker deployment mode uses the `docker run` command to instantiate and run a Docker container. The command creates a container from the specified image and then runs that container.

Using the Docker deployment method, you can use the parameter `-v` or `--volume` with the `docker run` command to mount a host directory as a volume to a container directory. This parameter allows you to use data external to the container.

For more information about Docker deployment, see the Docker documentation.

Docker Compose

Docker Compose is a tool for defining and running complex, multi-container Docker applications. A Docker Compose file is a YAML (`docker-compose.yaml`) file that contains configuration details for one or more services that represent the different parts of your application. For example, one service can represent your business logic while another service represents the client. For more information about Docker Compose, see the Docker Compose documentation.

To run Docker containers using Docker Compose:

```
docker-compose up –d
```

Minikube

Minikube allows Kubernetes to run locally inside a virtual machine.

To learn how to to use and configure Minikube, see the Minikube documentation.

Configure the config.properties file

Configure the `config.properties` file according to deployment requirements. The `config.properties` file is located inside the `deploy` folder of PAS for OpenEdge image.

To define configuration properties for your PAS for OpenEdge container, open the `config.properties` file, and modify the settings as follows:

Configuration	Description
DEPLOYMENT.MODE	The following deployment modes are available: • Docker • Docker Compose • Minikube You can only use one mode at a time.
APP.DOCKER.IMAGE.NAME	The name of the Docker image for the ABL application.
APP.DOCKER.IMAGE.TAG	The tag for the Docker image for the ABL application.
JDK.DOCKER.IMAGE.NAME	The name of the Docker image that contains the JDK.
JDK.DOCKER.IMAGE.TAG	The tag for the Docker image for the JDK.
JDK.DOCKER.IMAGE.JAVA.LOCATION	The location of the JDK inside the container.
PAS.INSTANCE.NAME	The name of the PAS for OpenEdge instance.
PASOE.DOCKER.IMAGE.NAME	The name of the PAS for OpenEdge Docker image. **Note:** You do not need to change this value because it already configured. This value should only be modified for a different PAS for OpenEdge image name and tag.
PASOE.DOCKER.IMAGE.TAG	The tag for the PAS for OpenEdge Docker image. The tag value corresponds to the version of OpenEdge.
PASOE.HTTPS.PORT	The port number used by PAS for OpenEdge to communicate. Specify a port number from the machine on which Docker is running.
FLUENTBIT.LOGGING	Specifies whether or not Fluent Bit logging is enabled. The default setting is true.

After the configuration properties are set, save the config.properties file and close the editor.

Note: Use the same name, tag, and Java location of the JDK Docker image for JDK.DOCKER.IMAGE.NAME, JDK.DOCKER.IMAGE.TAG, and JDK.DOCKER.IMAGE.JAVA.LOCATION, respectively.

Sample configuration for different deployment modes

This topic contains sample configurations for Docker, Docker Compose, and Minikube. You can use any of these tools to deploy an ABL application to a PAS for OpenEdge container. However, you must use the mode defined in the config.properties file.

Sample Docker configuration

The following code is the configuration for the Docker deployment:

```
# Deployment mode can be one of: docker/docker-compose/minikube
DEPLOYMENT.MODE=docker

# Name and tag with which the app container image will be built
# Same name will be used as the APP_NAME for fluent bit logging
APP.DOCKER.IMAGE.NAME=<sports>
APP.DOCKER.IMAGE.TAG=<latest>

# Container image that contains JDK (compatible) in it
JDK.DOCKER.IMAGE.NAME=<adoptopenjdk>
JDK.DOCKER.IMAGE.TAG=<11.0.4_11-jdk-hotspot>
# Location/Path to JDK inside container
JDK.DOCKER.IMAGE.JAVA.LOCATION=</opt/java/openjdk>

PAS.INSTANCE.NAME=<oepas4>
PASOE.DOCKER.IMAGE.NAME=<progresssoftware/pasoe>
PASOE.DOCKER.IMAGE.TAG=<12.2.0>
# In case of kubernetes, provided port should be in the default node port range:
30000-32767
PASOE.HTTPS.PORT=<50813>

# Flag to enable fluent bit logging, defaults to true
FLUENTBIT.LOGGING=<true>
```

See also:

- command-line entry for docker run
- reference material for docker run

Sample Docker Compose configuration

The following code is the configuration for the Docker Compose deployment. The only change in the `config.properties` file is `DEPLOYMENT.MODE` is set to `docker-compose`:

```
# Deployment mode can be one of: docker/docker-compose/minikube
DEPLOYMENT.MODE=docker-compose

# Name and tag with which the app container image will be built
# Same name will be used as the APP_NAME for fluent bit logging
APP.DOCKER.IMAGE.NAME=<sports>
APP.DOCKER.IMAGE.TAG=<latest>

# Container image that contains JDK (compatible) in it
JDK.DOCKER.IMAGE.NAME=<adoptopenjdk>
JDK.DOCKER.IMAGE.TAG=<11.0.4_11-jdk-hotspot>
# Location/Path to JDK inside container
JDK.DOCKER.IMAGE.JAVA.LOCATION=</opt/java/openjdk>

PAS.INSTANCE.NAME=<oepas4>
PASOE.DOCKER.IMAGE.NAME=<progresssoftware/pasoe>
PASOE.DOCKER.IMAGE.TAG=<12.2.0>
# In case of kubernetes, provided port should be in the default node port range:
30000-32767
PASOE.HTTPS.PORT=<50813>

# Flag to enable fluent bit logging, defaults to true
FLUENTBIT.LOGGING=<true>
```

See also:

- Docker Compose documentation

Sample Minikube configuration

To use the Minikube deploy script, make the following changes in your configuration:

- `DEPLOYMENT.MODE`—Provide the deployment mode as `minikube`.
- `PASOE.HTTPS.PORT`—Provide the port number. The number provided must be between **30000** and **32767**.

```
# Deployment mode can be one of: docker/docker-compose/minikube
DEPLOYMENT.MODE=minikube
# Name and tag with which the app container image will be built
# Same name will be used as the APP_NAME for fluen tbit logging
APP.DOCKER.IMAGE.NAME=<sports>
APP.DOCKER.IMAGE.TAG=<latest>

# Container image that contains JDK (compatible) in it
JDK.DOCKER.IMAGE.NAME=<adoptopenjdk>
JDK.DOCKER.IMAGE.TAG=<11.0.4_11-jdk-hotspot>
# Location/Path to JDK inside container
JDK.DOCKER.IMAGE.JAVA.LOCATION=</opt/java/openjdk>

PAS.INSTANCE.NAME=<oepas4>
PASOE.DOCKER.IMAGE.NAME=<progresssoftware/pasoe>
PASOE.DOCKER.IMAGE.TAG=<12.2.0>

# In case of kubernetes, provided port should be in the default node port range:
30000-32767
PASOE.HTTPS.PORT=<30813>

# Flag to enable fluent bit logging, defaults to true
FLUENTBIT.LOGGING=<true>
```

8

Deploy and verify an ABL application

After the configuration of the `config.properties` file is complete, you can deploy the ABL application to the PAS for OpenEdge Docker container. To deploy the ABL application, enter `ant deploy` in the command line.

If the application deployment is successful, then a `BUILD SUCESSFUL` message appears. A deployment can fail for various reasons. For more information, see Troubleshooting on page 39.

To view the deployed package and verify that your application is running, access the service URL (`https://localhost:`*`PASOE.HTTPS.PORT`*).

Note: It can take several minutes after deployment for a service to be available.

Undeploy an ABL application

You can undeploy an instance after the successful completion of a deployment.

1. To undeploy the application, use the same `config.properties` file that was used for deploying the same instance.
2. Enter `ant undeploy` in the command line.

If the undeployment of the application is successful, then a `BUILD SUCESSFUL` message appears.

Redeploy an ABL application to a PAS for OpenEdge Docker container

You can redeploy a container after it was undeployed. To redeploy, review the following conditions. Some conditions include modifications of the `config.properties` file.

Note: The sidecar approach supports the deployment of multiple PAS for OpenEdge instance simultaneously for any deployment mode. You only need to provide a unique instance name and port number.

- **No change in configurations**—If there are no changes in the configurations, then deployment can be performed sequentially. Manual undeployment is not required. The script performs the undeployment and also redeploys the instance.
- **Change in configurations**—There are multiple conditions for changes in configurations.
  - **Change the instance name**—This is a new deployment. For every deployment of Docker and Docker Compose, the port number must be unique.
  - **Change the deployment mode**—This is for new deployments.
  - **Change other configurations with the same instance name**—With this condition, the script undeploys and then redeploys.

9

Other options for deployment

You can create your Docker deployment without using the provided deploy scripts. Docker can build a custom container image by reading instructions from a Dockerfile. This topic provides examples of how to update a Dockerfile to customize PAS for OpenEdge within a custom container image.

Create a PAS for OpenEdge instance

An instance consists of a set of OpenEdge ABL applications and instance-level configurations. The following is the directory structure:

`conf`	Instance-level configurations are defined in this directory.
`tlr`	This directory contains the run-time properties to be tailored at the instance level.
`ablapps`	This directory contains the archive files for your ABL application.

Create an instance archive file

To create an instance archive file, create an `ablapps` directory, and place your ABL application archives inside it. If you have `conf` and `tlr` directories, then keep them with the `ablapps` directory. After setting up your directories, create an archive file from the three directories.

Note: It is not mandatory to have the `conf` and `tlr` folders to create an instance archive.

Deploy using the stand-alone Docker image

You can create a stand-alone Docker image containing PAS for OpenEdge and an ABL application by using a Dockerfile. Everything needed for a PAS for OpenEdge application is contained in this one image.

You must configure environment variables and provide some configurations when you create a Dockerfile:

INSTANCE_NAME	This environment variable defines the name of the PAS for OpenEdge instance. The name is used when creating a PAS for OpenEdge instance in the Docker container. This environment variable is mandatory.
APP_NAME	(Optional) This environment variable defines the application name used when Fluent Bit logging is enabled. The name is logged as a new field in each log entry.
FLUENTBIT_LOGGING	(Optional) Fluent Bit logging is enabled by default. To disable logging, change this environment variable property to false.

You can provide run-time configurations by creating a `runtime.properties file` and placing it in `/deploy/scripts/config`. For example, to provide the database configuration for an ABL application, provide the following configuration:

ABLAPP-NAME`.DB.CONNECTION.PARAMS=-db` *`additional parameters`*

There are two ways to create a Dockerfile:

- Create a Dockerfile using JDK as a Docker image:

```
FROM JDK Docker image name:JDK Docker image tag as builder-jdk

              FROM store/progresssoftware/pasoe:12.2.0

              USER pscadmin

              COPY --chown=pscadmin:pscadmin Path of the license file
/psc/dlc/progress.cfg

             COPY --from=builder-jdk --chown=pscadmin:pscadmin Java location inside
 the JDK Docker image /usr/java

              COPY --chown=pscadmin:pscadmin Path to deployable application
/deploy/artifacts/

              #COPY --chown=pscadmin:pscadmin Path to runtime properties file
/deploy/scripts/config

              ENV \
              INSTANCE_NAME="instance name" \
              APP_NAME="application name" \
              FLUENTBIT_LOGGING=true/false

              EXPOSE 8811

              CMD ["/bin/sh", "-c", "sh /deploy/scripts/startServer.sh"]
```

- Create a Dockerfile using the downloaded JDK:

```
FROM store/progresssoftware/pasoe:12.2.0

              USER pscadmin

              COPY --chown=pscadmin:pscadmin Path of the license file
/psc/dlc/progress.cfg

              COPY --chown=pscadmin:pscadmin Path of the installed JDK location
/usr/java

              COPY --chown=pscadmin:pscadmin Path to deployable application
/deploy/artifacts/

              #COPY --chown=pscadmin:pscadmin Path to runtime properties file
/deploy/scripts/config

              ENV \
              INSTANCE_NAME="instance name" \
              APP_NAME="application name" \
              FLUENTBIT_LOGGING=true/false

              EXPOSE 8811

              CMD ["/bin/sh", "-c", "sh /deploy/scripts/startServer.sh"]
```

After the Dockerfile is created, build and run the container:

1. Build the new image:

   ```
   docker build . -t PAS image name
   ```

2. Run the container:

   ```
   docker run --name Container name -p host port:8811 PAS image name
   ```

To verify that the PAS for OpenEdge server is running, enter `https://host ip:host port` in your browser.

See also:

- Command-line entry for docker run
- Reference material for docker run

10

Upgrade a PAS for OpenEdge image in a container

You may need to update your PAS for OpenEdge container to a newer version of PAS for OpenEdge. The new image can be based on a newer version of CentOS, a new service pack release, or a new PAS for OpenEdge release. Do not modify a container to upgrade the OS or the PAS for OpenEdge server. Docker containers are immutable. Any changes made to a container are lost when the container is removed. Instead, rebuild the image based on the new version of PAS for OpenEdge. All newly created containers are based on the new base image.

To upgrade a custom container image:

1. Download the new base PAS for OpenEdge image from the specified location.
2. Load the image into your Docker environment.
3. Integrate the license support.
4. Build a custom container image based on the image loaded into the Docker environment. Use a Dockerfile and a `docker-compose` file (if required) to build the custom container image.

   You can use the `docker tag` functionality to tag a custom container image. The tag functionality gives you easy access to a specific build or version of an application.
5. Test the custom container image.
6. If the test succeeds as part of your CI/CD pipeline, then you can use the image in your production environment. To deploy the container in a production environment:

   a. Exit and delete the containers that currently run an older version of PAS for OpenEdge.

   b. Use the custom container image to create and run containers.

You can combine this workflow with the Rolling Update functionality in orchestration software such as Kubernetes to achieve minimal downtime.

11

Troubleshooting

If you receive the `BUILD SUCCESSFUL` message, but the deployed container is not active and running, then do the following:

- Verify that the JDK version you are using is supported, by checking the **Product Availability Guide (PAG)**.
- Check the container logs for errors in the JDK path or the JDK image name:
  - To check simple Docker logs (Docker and Docker Compose), run: `docker container logs` *`container_name`*
  - To check Minikube logs, check the pod status: `kubectl logs` *`pod_name`*. For more information about pods, run: `kubectl describe pods/`*`pod_name`*