



Build your own OpenEdge container images

A Workshop to start using OpenEdge in a Docker environment

Authors	Version
Laurent Kieffer, Ruben Droge , Stefan Bolte	1.0

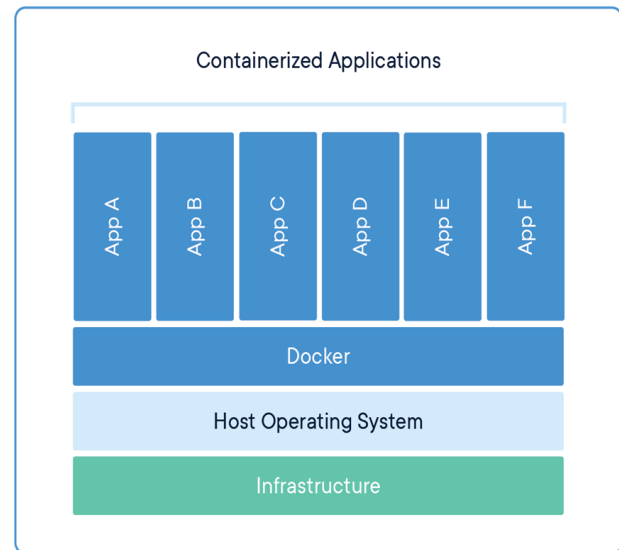
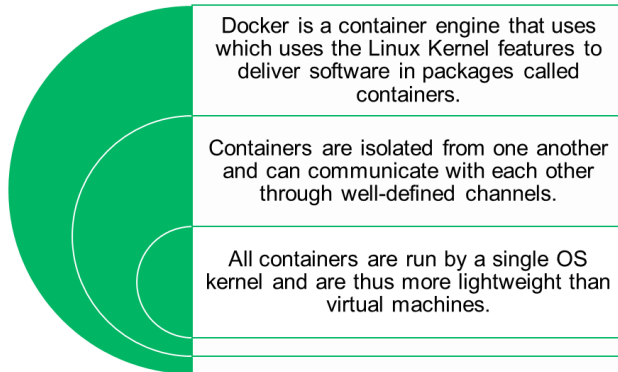


Table of Contents

Table of Contents	1
Introduction	2
Chapter 1 – Docker Desktop and CLI	2
Prerequisites : before starting the labs	3
LAB 1 – Create your first Dockerfile : Base image	3
LAB 2 – Dockerfile and OpenEdge : OpenEdge Image	4
LAB 3 – OpenEdge Container access to OpenEdge Database	7
LAB 4 – OpenEdge Container with oepas1 and deploy services	7
LAB 5 – OpenEdge Container with oepas1 and APSV	9
LAB 6 – Save your OpenEdge Container to an image.....	10
LAB 7 – Use the Database image from Progress ESD	10
LAB 8 – Use the PASOE image from Progress ESD	14
LAB 9 – Sample App deployment with PASOE Docker image	15
LAB 10 – How to define multi container : docker-compose.....	19
Appendix A : Useful Docker commands	20
Appendix B : Docker , Docker Desktop , JDK	22

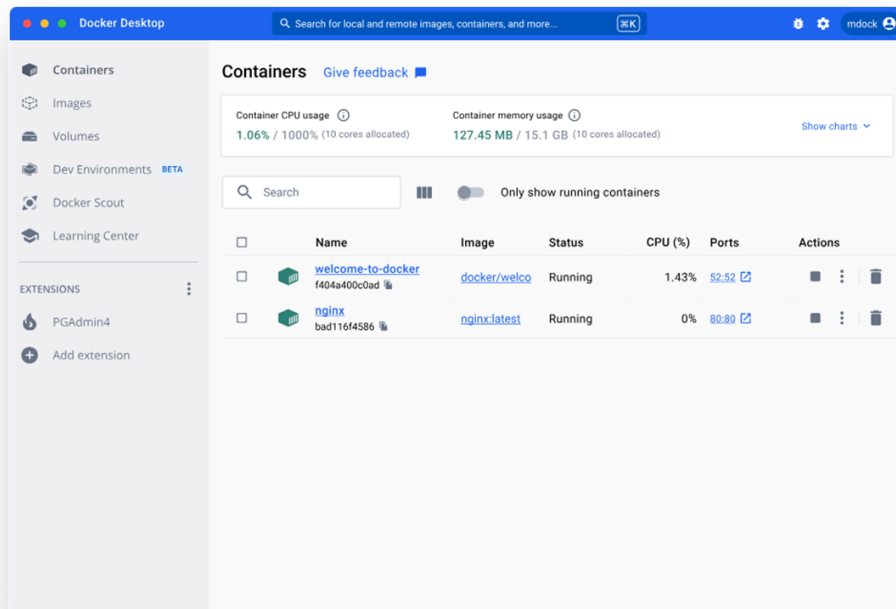
Introduction

What is Docker?



Chapter 1 – Docker Desktop and CLI

First we will use Docker Desktop to use an existing Docker image to start a container and play with it. Look at the images and containers available



Prerequisites : before starting the labs

To install on your laptop before starting :

- In your local directory c:\openedge\wrk\, download sample code from :
<https://github.com/lkieffer2002/pug2024OEDocker>
 - o You should have a LabsDocker directory and a DockerComposeTest directory
- Some material will be provided by the Progress Team to copy in your existing LabsDocker directory
 - o Linux OpenEdge installation package 12.8
 - o Open JDK
- Docker Desktop : download and install
 - o <https://docs.docker.com/desktop/install/windows-install/>
- JDK 17 : download or use material provided
 - o [Latest Releases | Adoptium](#) (if you use this you will need to change in some labs)

LAB 1 – Create your first Dockerfile : Base image

What is a Dockerfile

- A text document that contains all the commands a user could call on the command line to assemble an image.
- “docker build” executes several command-line instructions in succession and build the image

The goal will be to create a Docker image based on linux Ubuntu:22.04

Remark : when using docker commands be aware be careful when naming files as it is case sensitive

Actions to perform
Use Proenv to open a command line and position in the OpenEdge working directory Type “prompt” to show the directory
Create a Docker folder Cd Docker
Create a text file named DockerFileBase
Add the following lines in the file FROM ubuntu:22.04 RUN apt-get update && apt-get install -y iputils-ping

<pre> RUN mkdir /psc RUN mkdir /psc/jdk RUN mkdir /psc/install RUN mkdir /psc/install/tmp RUN echo 'tcp 6 TCP' >> /etc/protocols </pre>
<p>On the command line</p> <pre> docker build -t baseimage -f DockerFileBase . </pre> <p>or</p> <pre> docker build -t baseimage -f DockerFileBase.txt . (if you created a txt file) </pre>
<p>See in Docker Desktop the new image</p> <p>On the command line : docker images</p>

LAB 2 – Dockerfile and OpenEdge : OpenEdge Image

As you have a first Docker image ready we will add Openedge aspects.

The goal is to perform a OpenEdge installation in silent mode using components we want to use.

As from OE 12.1 first step will be to define a correct JDK version to use , then install OpenEdge

In the working directory you should find :

- OpenJDK17U-jdk_x64_linux_hotspot_17.0.6_10.tar : jdk to use
- PROGRESS_OE_12.8.3_LNX_64.tar : OpenEdge installation package
- Response.ini file : to use for the silent installation

The files can be found in the LabsDocker directory (c:\openedge\wrk\labsdocker)

Actions to perform
Create a text file named DockerFile128 or DockerFile128.txt
Add the following lines in the file
<pre> FROM baseimage COPY ./OpenJDK17U-jdk_x64_linux_hotspot_17.0.6_10.tar /psc/jdk RUN tar xvf /psc/jdk/OpenJDK17U-jdk_x64_linux_hotspot_17.0.6_10.tar -C /psc/jdk COPY ./PROGRESS_OE_12.8.3_LNX_64.tar /psc/install </pre>

```
RUN tar xvf /psc/install/PROGRESS_OE_12.8.3_LNX_64.tar -C /psc/install
COPY ./response.ini /psc/install
RUN /psc/install/proinst -b /psc/install/response.ini -l /psc/install/tmp/silentinstall.log
```

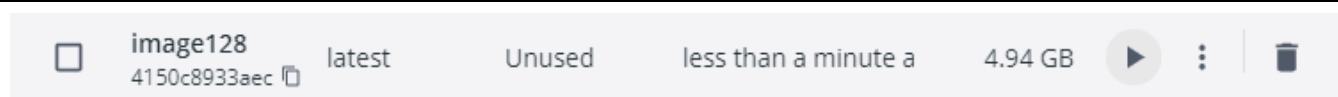
```
RUN rm /psc/install/PROGRESS_OE_12.8.3_LNX_64.tar
```

```
RUN /psc/dlc/bin/proenv
```

```
EXPOSE 8810
EXPOSE 8811
EXPOSE 8820
```

On the command line
`docker build -t image128 -f DockerFile128 .`

See in Docker Desktop the new image
On the command line : `docker images`



In the docker Desktop you should see the image128.
Click on the “run” button then choose “optional settings”.
You can give the Container a name
Ports on the left allows to map the host port with the port opened in container.
As you see 8810,8811,8812 are the default ports used when using a oepas1 PASOE instance

Volumes

Host path

...

Container path

+

Environment variables

Cancel

Run

From the CLI you can also run a container based on an image through

```
Docker run -dt --name container128 -p 8810:8810 -p 8811:8811 -p 4000-5000:4000-5000 -d image128
```

This command will run a “container128” container from image128 with port 8810,8811 and from 4000 to 5000.

Meanings :

- p (port) hostport:container port
- It can be a range : example 4000-5000:4000-5000

After the container is running click on the Container Name then on the Terminal

```
On the prompt type
cat /etc/os-release
```

It should present

```
bin dev fcs.tab lib lib64 media opt psc run srv tmp var
boot etc home lib32 libx32 mnt proc root sbin sys usr
```

```
cd psc
```

You will find it familiar
/psc/dlc/bin/proenv to set OpenEdge environment variables

LAB 3 – OpenEdge Container access to OpenEdge Database

As you have a first OpenEdge container running, you will access your database hosted on the host machine .

On your host machine find your ip address with ipconfig
On your host machine in a proenv session create a sports2000 database and start Prodb sports2000 sports2000 Proserve sports2000 -S 4567
From your container in a proenv session Mpro sports2000 -H host ip -S 4567
Access some data

LAB 4 – OpenEdge Container with oepas1 and deploy services

In this lab we will see how to deploy some REST services to the oepas1 instance.

Copy the following files from LabsDocker directory to your docker directory

- Advcustomer.r
- Openedge.properties
- SportsInc.war

Create an oelogs directory under your docker directory to get all oepas1 logs from the PASOE instance running in the container
Create a text file named DockerFile128Services or DockerFile128Services.txt
Add the following lines in the file
FROM image128 COPY AdvCustomer.r /psc/wrk/oepas1/openedge COPY SportsInc.war /psc/wrk COPY openedge.properties /psc/wrk/oepas1/conf/openedge.properties RUN /psc/wrk/oepas1/bin/tcman.sh deploy -l oepas1 /psc/wrk/SportsInc.war EXPOSE 8810 EXPOSE 8811 EXPOSE 8820

EXPOSE 9090
EXPOSE 4567

`docker build -t image128services -f DockerFile128Services.txt .`

Run the container from the new image128services

```
docker run -dt --name container128services -v  
C:\OpenEdge128\WRK\docker\oelogs:/psc/wrk/oepas1/logs -p 4567:4567 -p 8810:8810 -p  
8811:8811 -p 9090:9090 -d image128services
```

-v allows to mount the host log directory with the oepas1 log directory in the container

Check on the running container in /psc/wrk/oepas1/conf.openedge.properties the connection to the database .

It should be like

```
agentStartupParam=-T "${catalina.base}/temp" -db sports2000 -H 10.248.65.119 -S 4567
```

Change the -H to your localhost ip address

Open a Terminal in your running container

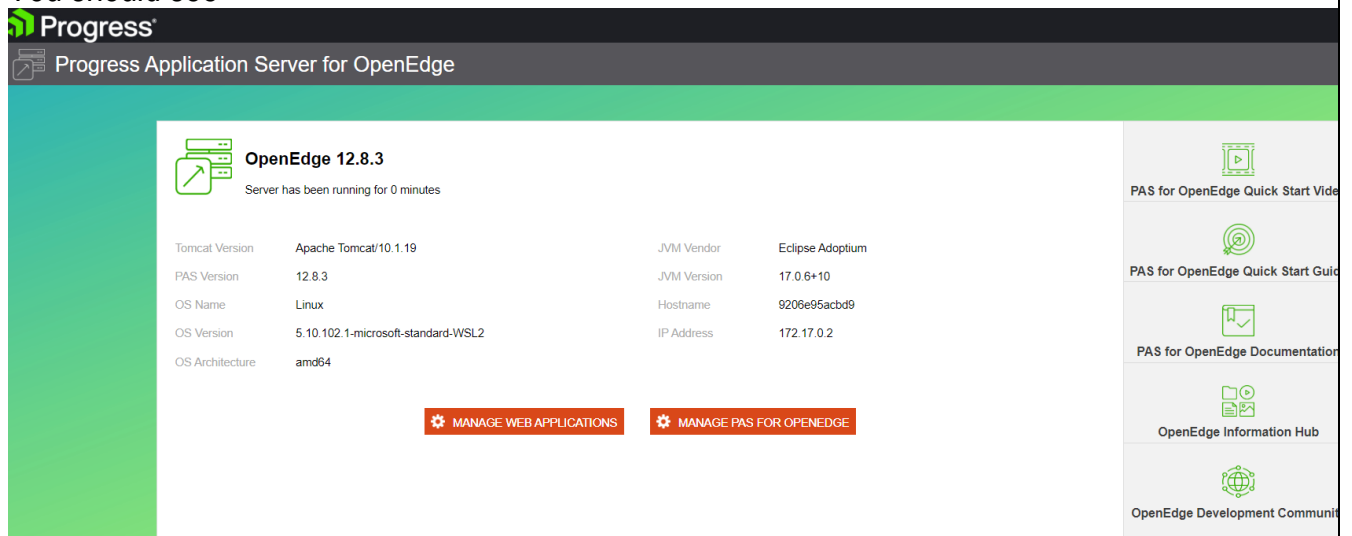
Start the oepas1 instance

```
cd /psc/wrk/oepas1/bin  
sh tcman.sh start
```

See the host log directory

Open a browser and type : <http://localhost:8810>

You should see



Manage Pas for OpenEdge

tomcat/tomcat to access the PASOE Manage APIs (swagger)

Try : <http://localhost:8810/SportsInc/static/SportsIncService.json>

You will see a catalog of available services (all are not active)

Try : <http://localhost:8810/SportsInc/rest/SportsIncService/Customer>
you should see

```
{
  - dsCustomer: {
    prods:hasChanges: true,
    - ttCustomer: [
      - {
        id: "0x0000000000000061",
        seq: 1118,
        CustNum: 1,
        Name: "Lift Tours",
        Address: "276 North Drive",
        Address2: "",
        Balance: 903.64,
        City: "Burlington",
        Comments: "This customer is on credit hold.",
        Contact: "Gloria Shepley",
        Country: "USA",
        CreditLimit: 66700,
        Discount: 35,
        EmailAddress: "",
        Fax: "",
        Phone: "(617) 450-0086",
        PostalCode: "01730",
        SalesRep: "HXM",
        State: "MA",
        Terms: "Net30"
      },
      - {
        id: "0x0000000000000062",
```

LAB 5 – OpenEdge Container with oepas1 and APSV

In this lab you will access your oepas1 server running in the container through the APSV transport.
This will show how to use some docker commands to copy from a host folder to a container folder.

Run the docker command : docker ps and look at the container id					
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					
9206e95acbd9	image128services	"/bin/bash"	29 minutes ago	Up 14 minutes	0.0.0.0:4567->4567/tcp, 0.0.0.0:8811->8810-8811/tcp, 0.0.0.0:9090->9090/tcp, 8820/tcp
container128services					
In OpenEdge development on your host create a procedure named : listprograms.p					
OS-COMMAND SILENT "ls /psc/wrk/oepas1/openedge" >> /psc/wrk/oepas1/logs/myprograms.txt.					
Copy the procedure in the docker container					
docker cp C:\OpenEdge128\WRK\LabsDocker\listprograms.p					
9206e95acbd9:/psc/wrk/oepas1/openedge					
In OpenEdge development on your host create a procedure to invoke the listprograms.p on oepas1					
DEFINE VARIABLE happ AS HANDLE.					

DEFINE VARIABLE retok AS LOGICAL.

CREATE SERVER happ.

retok = happ:CONNECT("-URL http://localhost:8810/apsv" , "" , "").

MESSAGE retok

VIEW-AS ALERT-BOX INFORMATION BUTTONS OK.

RUN listprograms.p ON happ.

Look in the oelogs directory : myprograms.txt

Extra activity : Develop your own server program and run it from the host program

LAB 6 – Save your OpenEdge Container to an image

You can save a container after any modifications to a new image

Run the command

```
docker commit container128services saved128
```

See the images in Docker Desktop, or run docker images

You should see a new image you can use

Run a new container from this saved image

What should the next command do ?

```
docker save -o saved128.tar image128services
```

LAB 7 – Use the Database image from Progress ESD

Progress provides PASOE and Database as docker image on the download site ESD.

+ OpenEdge Enterprise & Advanced Enterprise Relational Database Container Images	708.3 MB	↓ PROGRESS_OE_DATABASE_CONTAINER_IMAGES_12.8.3_LNX_64.zip
+ Progress Application Server for OpenEdge Container Image	427.1 MB	↓ PROGRESS_PASOE_CONTAINER_IMAGE_12.8.3_LNX_64.zip

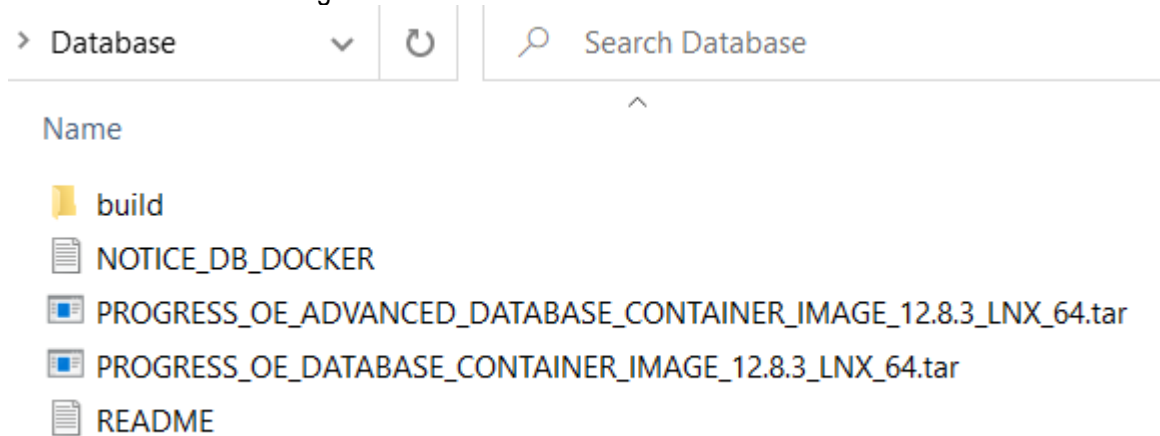
This image does not come with a progress.cfg file. This configuration file should be referenced (or copied) in an image

For this lab we will follow instructions coming from the OpenEdge documentation with some additional informations

<https://docs.progress.com/bundle/openedge-database-docker-container-122/page/Why-use-an-OpenEdge-database-in-a-Docker-container.html>

From the LabsDocker folder extract the Progress_Database_Image. You will see a PROGRESS_OE_DATABASE_CONTAINER_IMAGES_12.8.3_LNX_64.zip file

Under your working directory (for example : c:\openedge\wrk\docker), create a subfolder **Database**.
Copye the PROGRESS_OE_DATABASE_CONTAINER_IMAGES_12.8.3_LNX_64.zip file in this directory and extract all
You should have something like

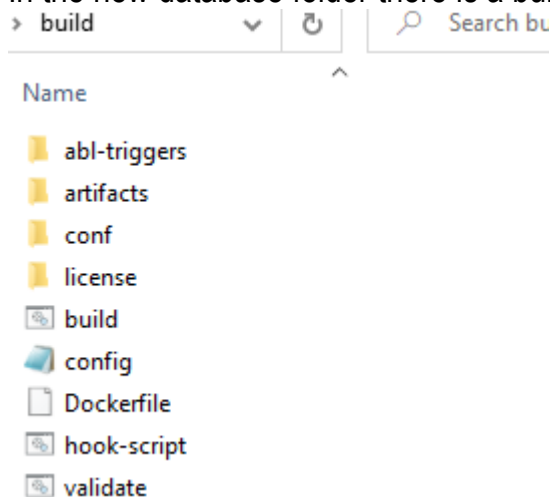


Run the docker command using this file
docker load -i PROGRESS_OE_DATABASE_CONTAINER_IMAGE_12.8.3_LNX_64.tar.gz

Loaded image: progresssoftware/prgs-oedb:12.8.3

Docker images or see in docker desktop the images

In the new database folder there is a build directory



Copy a progress.cfg provided from the labsdocker directory in the license directory.

In the conf directory change the startup.pf and add :

```
# Provide DB parameters in this file
-S myservice -minport 4000 -maxport 4500
```

Open the Dockerfile and copy the below content (replace all)... look in bold

Explanations :

We want to use the loaded Database image :

ARG DB_DOCKER_IMAGE_NAME=**progresssoftware/prgs-oedb**

ARG DB_DOCKER_IMAGE_TAG=**12.8.3_ent**

We want to reuse the previously image created including a correct JDK:

ARG JDK_DOCKER_IMAGE_NAME=**image128**

ARG JDK_DOCKER_IMAGE_TAG=**latest**

We want to use the JDK provided by the image including JDK (image128)

COPY --from=builder-jdk --chown=pscadmin:pscadmin **/psc/jdk/jdk-17.0.6+10** /usr/java

We want to create a sports2020 database as SampleDB

Arguments used for images

ARG DB_DOCKER_IMAGE_NAME=**progresssoftware/prgs-oedb**

ARG DB_DOCKER_IMAGE_TAG=**12.8.3_ent**

ARG JDK_DOCKER_IMAGE_NAME=**image128**

ARG JDK_DOCKER_IMAGE_TAG=**latest**

Use JDK image as a staging image

FROM \${JDK_DOCKER_IMAGE_NAME}:\${JDK_DOCKER_IMAGE_TAG} AS builder-jdk

Build the new OpenEdge Database image

FROM \${DB_DOCKER_IMAGE_NAME}:\${DB_DOCKER_IMAGE_TAG}

USER pscadmin

Copy license file

COPY --chown=pscadmin:pscadmin ./license/progress.cfg /psc/dlc/progress.cfg

Copy JAVA

ARG JDK_DOCKER_IMAGE_JAVA_LOCATION

#COPY --from=builder-jdk --chown=pscadmin:pscadmin

\${JDK_DOCKER_IMAGE_JAVA_LOCATION} /usr/java

COPY --from=builder-jdk --chown=pscadmin:pscadmin **/psc/jdk/jdk-17.0.6+10** /usr/java

Copy DB related artifacts

COPY --chown=pscadmin:pscadmin ./artifacts /deploy/artifacts/

COPY --chown=pscadmin:pscadmin ./abl-triggers /deploy/abl-triggers/

COPY --chown=pscadmin:pscadmin ./conf /deploy/scripts/config/

COPY --chown=pscadmin:pscadmin ./hook-script.sh /deploy/scripts/

default values in case they are not provided in config.properties

```

ARG DB_CREATE_METHOD=sampleDB
ARG DB_NAME=sports2020
ARG SAMPLE_DB_NAME=sports2020
ARG EXTERNAL_DATABASE_PATH=/usr/wrk

# set environment variables
ENV DB_CREATE_METHOD=${DB_CREATE_METHOD} \
  DB_NAME=${DB_NAME} \
  SAMPLE_DB_NAME=${SAMPLE_DB_NAME} \
  EXTERNAL_DATABASE_PATH=${EXTERNAL_DATABASE_PATH} \
  LD_LIBRARY_PATH=/usr/java/jre/lib/amd64/server:/usr/java/jre/lib/amd64:/usr/java/lib/server \
  DB_BROKER_PORT=4567 \
  DB_MINPORT=4000 \
  DB_MAXPORT=4500

# add a service myservice
RUN echo 'myservice 4567/tcp' >> /etc/services

# creates the relevant OpenEdge Database
RUN /deploy/scripts/create-db.sh

CMD ["/bin/sh", "-c", "/deploy/scripts/start-db-server.sh"]

```

Run the docker command to create the new image

```
docker build -t sportsdb -f Dockerfile .
```

To run a new container based on the sportsdb image

```
docker run --hostname=localhost --name sportsdbcontainer -p 4567:4567 -p 4000:4500 -e DB_BROKER_PORT=4567 -e DB_MINPORT=4000 -e DB_MAXPORT=4500 sportsdb
```

From an OpenEdge desktop connect to the database using

LAB 8 – Use the PASOE image from Progress ESD

Progress provides PASOE and Database as docker image on the download site ESD.

+ OpenEdge Enterprise & Advanced Enterprise Relational Database Container Images	708.3 MB	↓ PROGRESS_OE_DATABASE_CONTAINER_IMAGES_12.8.3_LNX_64.zip
+ Progress Application Server for OpenEdge Container Image	427.1 MB	↓ PROGRESS_PASOE_CONTAINER_IMAGE_12.8.3_LNX_64.zip

This image does not come with a progress.cfg file. This configuration file should be referenced (or copied) in an image

For this lab we will follow instructions coming from the OpenEdge documentation with some additional informations

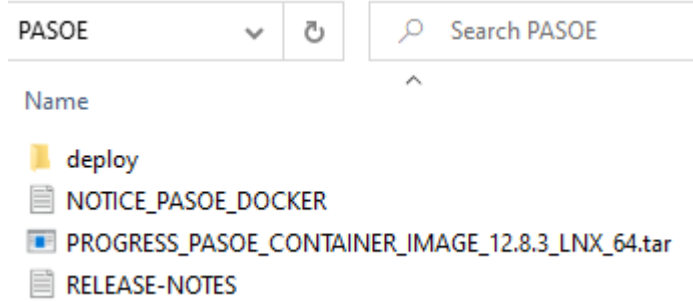
<https://docs.progress.com/bundle/openedge-database-docker-container-122/page/Why-use-an-OpenEdge-database-in-a-Docker-container.html>

From the LabsDocker folder extract the Progress_Database_Image. You will see a PROGRESS_OE_PASOE_CONTAINER_IMAGES_12.8.3_LNX_64.zip file

Under your working directory (for example : c:\openedge\wrk\docker), create a subfolder **PASOE**.

Copy the PROGRESS_OE_PASOE_CONTAINER_IMAGES_12.8.3_LNX_64.zip file in this directory and extract all

You should have something like



Run the docker command using this file

Docker load -i PROGRESS_PASOE_CONTAINER_IMAGE_12.8.3_LNX_64.tar.gz

Loaded image: progresssoftware/prgs-pasoe:12.8.3

Docker images or see in docker desktop the images

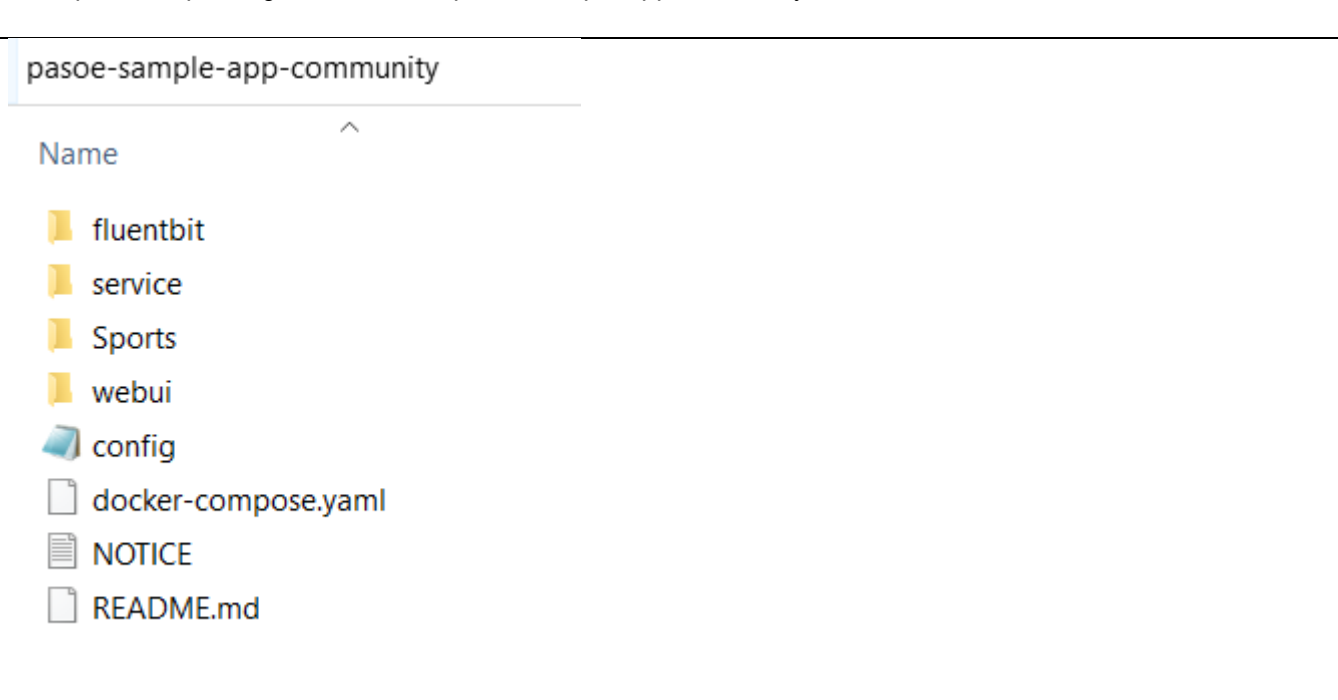
docker build -t pasoe128 -f DockerfilePasoe128.txt .

LAB 9 – Sample App deployment with PASOE Docker image

This labs will be inspired from <https://community.progress.com/s/question/0D5Pb00000ZgWSLKA3/use-the-container-image-for-pas-for-openedge-1281-1282-1283-1284-12214-12215-12216-with-a-sample-application>

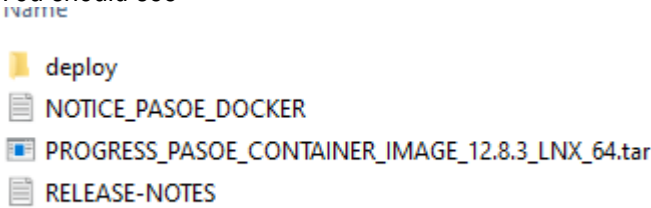
Download the SampleApp_12.8.2 zip file and uncompress it in some directory

Example : C:\OpenEdge\WRK\docker\pasoe-sample-app-community



The Sports directory is the application to deploy

<div>mp... > Sports</div> <div> <div>Name</div> <div> <div>.scripts</div> <div>.services</div> <div>.settings</div> <div>AppServer</div> <div>conf</div> <div>PASOEContent</div> <div>tlr</div> <div>.project</div> <div>.propath</div> <div>build.config</div> <div>build</div> <div>build</div> </div> </div>	
In the build.properties file position DLC to your OpenEdge directory (in the case below : c:\progress\openedge128)	
<pre># Properties file for build.xml # WORK location used in '.propath', similar to PDSOE '@{WORK}' # Default value is the project root location WORK = . ## Tailor these values if structure is changed #ROOT = <project root. Defaulted to current folder> #WEBAPP = <Location of webapp. Defaulted to \${ROOT}\PASOEContent> DLC = c:\progress\openedge128 #STARTUP-FILE-PATH = <location of startup.pf file. Defaulted to \${ROOT}/conf/startup.pf> #PROPATH-FILE-PATH = <location of .propath file. Defaulted to \${ROOT}/.propath> #PROJECT.NAME = <Name of the ablapp and webapp. Defaulted to the project folder name></pre>	
In the conf directory change the startup.pf to connection to a sports database on your host (below 192.168.1.32)	
<pre>-db sports -H 192.168.1.32 -S 1234</pre>	
<p>On your host laptop start a 12.8 proenv</p> <pre>Prodb sports sports2000 Proserver sports -S 1234</pre>	
<p>If everything is set</p> <pre>Cd C:\OpenEdge\WRK\docker\pasoe-sample-app-community\sports</pre> <p>ant package</p>	
<p>At the end you should see</p> <pre>[ABLWebAppPackage] Cleaning up temporary files... [ABLWebAppPackage] War file generation task completed successful. [zip] Building zip: C:\OpenEdge\WRK\docker\pasoe-sample-app-community\Sports\PASOEContent\output\package-output\Sports.war [delete] Deleting: C:\OpenEdge\WRK\docker\pasoe-sample-app-community\Sports\PASOEContent\output\package-output\Sports_tmp.war [zip] Building zip: C:\OpenEdge\WRK\docker\service\Sports.zip [copy] Copying 1 file to C:\OpenEdge128\WRK\docker\pasoe-sample-app-community\service</pre> <p>BUILD SUCCESSFUL Total time: 47 seconds</p> <p>Notice that we will use the sports.zip</p>	

Now for the PASOE image
<p>From the LabsDocker folder extract the Progress_Database_Image. You will see a PROGRESS_OE_PASOE_CONTAINER_IMAGES_12.8.3_LNX_64.zip file</p> <p>Unzip this file in a directory like c:\openedge\wrk\docker\pasoe-sample-app</p>
<p>You should see</p> 
<p>Now copy the necessary files from the previous build</p> <p>Cd c:\openedge\wrk\docker\pasoe-sample-app</p> <p>Copy C:\OpenEdge\WRK\docker\service\Sports.zip .\deploy\ablapps</p>
Update the config.properties file in c:\openedge\wrk\docker\pasoe-sample-app\deploy
<pre># Deployment mode can be one of: docker/docker-compose/minikube DEPLOYMENT.MODE=docker # Name and tag with which app container image will be built # Same name will be used as APP_NAME for fluentbit logging APP.DOCKER.IMAGE.NAME=sports APP.DOCKER.IMAGE.TAG=12.8.3 # Container image which contains JDK(compatible) in it JDK.DOCKER.IMAGE.NAME=jdkimage17 JDK.DOCKER.IMAGE.TAG=latest # Location/Path to JDK inside container JDK.DOCKER.IMAGE.JAVA.LOCATION=/opt/java/openjdk PAS.INSTANCE.NAME=oepas1 PASOE.DOCKER.IMAGE.NAME=progresssoftware/prgs-pasoe PASOE.DOCKER.IMAGE.TAG=12.8.3 # In case of kubernetes provide port should be in the default nodePort range: 30000-32767 PASOE.HTTPS.PORT=8811 # Flag to enable fluent-bit logging, defaults to 'true' FLUENTBIT.LOGGING=false</pre>
Copy a progress.cfg file in c:\openedge\wrk\docker\pasoe-sample-app\license
<p>If everything is set</p> <p>In a command line</p> <p>Cd c:\openedge\wrk\docker\pasoe-sample-app</p> <p>Ant -f ./deploy/build.xml deploy</p>
After some time you should see

deploy_app_start_pasoe:

[echo] PASOE instance named 'oepas1_pasoeinstance_d' will be available at 'https://localhost:8811'

deploy:

deploy:

BUILD SUCCESSFUL

Total time: 37 seconds

In your docker desktop you should see



[container128services](#)

72a91ba9fbb1



[dreamy_mendel](#)

f22c4486c9eb



[oepas1_pasoeinstance_d](#)

a50a5ae51e36



[dockercomposetest](#)



[genericservice_devcontainer](#)



[sampleapp](#)

Oepas1_pasoeinstance_d container running

Try

<https://localhost:8811>

<https://localhost:8811/Sports>

<https://localhost:8811/Sports/static/SportsService.json>

<https://localhost:8811/Sports/rest/SportsService/Customer>

Update the config.properties file in c:\openedge\wrk\docker\pasoe-sample-app\deploy using docker-compose

Deployment mode can be one of: docker/docker-compose/minikube

DEPLOYMENT.MODE=docker-compose

Name and tag with which app container image will be built

Same name will be used as APP_NAME for fluentbit logging

APP.DOCKER.IMAGE.NAME=sports

APP.DOCKER.IMAGE.TAG=12.8.3

Container image which contains JDK(compatible) in it

JDK.DOCKER.IMAGE.NAME=jdkimage17

JDK.DOCKER.IMAGE.TAG=latest

Location/Path to JDK inside container

JDK.DOCKER.IMAGE.JAVA.LOCATION=/opt/java/openjdk

PAS.INSTANCE.NAME=oepas1

PASOE.DOCKER.IMAGE.NAME=progresssoftware/prgs-pasoe

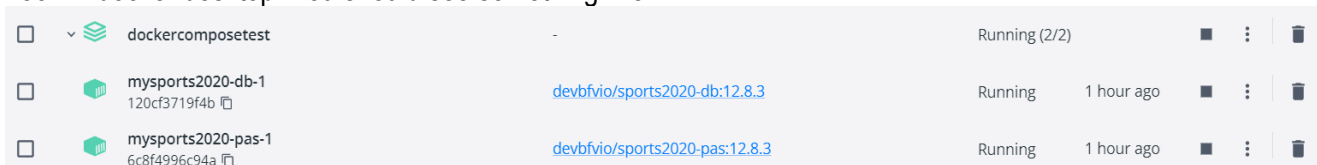
PASOE.DOCKER.IMAGE.TAG=12.8.3 # In case of kubernetes provide port should be in the default nodePort range: 30000-32767 PASOE.HTTPS.PORT=8811 # Flag to enable fluent-bit logging, defaults to 'true' FLUENTBIT.LOGGING=false
Ant -f ./deploy/build.xml deploy
In your docker desktop you should see <div> </div>

LAB 10 – How to define multi container : docker-compose

Docker compose allows to share multi container applications and define dependencies.
Docker compose uses a docker-compose.yaml fil as configuration

Steps to follow

Create a new folder name : dockercomposetest
In this folder create a file with name : .env with below content
DEVCONTAINER_IMAGE=docker.io/devbfvio/openedge-devcontainer:12.8.1-rc1 DB_IMAGE=docker.io/devbfvio/sports2020-db:12.8.3 PAS_IMAGE=docker.io/devbfvio/sports2020-pas:12.8.3 DEBUG_PORT=3099 PAS_PORT=8810 PROGRESS_CFG=./license/progress.cfg
Create a license folder in dockercomposetest
Copy some progress.cfg file (provided during the workshop) in this license folder
Create a docker-compose.yaml file in dockercomposetest directory and copy the below content
<pre>version: '3.8' services: mysports2020-db: image: \${DB_IMAGE} volumes:</pre>

<pre> - \${PROGRESS_CFG}:/usr/dlc/progress.cfg ports: - 10000-10010:10000-10010 environment: - DBNAME=sports2020 mysports2020-pas: image: \${PAS_IMAGE} volumes: - \${PROGRESS_CFG}:/usr/dlc/progress.cfg - ./src:/app/src - ./conf/as.pf:/app/config/as.pf ports: - \${PAS_PORT}:8810 environment: - PASWEBHANDLERS=/app/src/webhandlers/ROOT.handlers depends_on: - mysports2020-db </pre>
<p>Explanations :</p> <p>mysports2020-db : will be a container to run a sports2020 database</p> <p>mysports2020-pas : will be a container hosting a PASOE instance . This container will depend on mysports2020-db</p> <p>volumes: shows that there is a mapping between container host directory (.src) and container directory (/app/src). Same between ./conf/as.pf and /app/config/as.pf</p>
<p>Copy a provided src directory in the dockercomposetest directory</p> <p>Copy a provided conf directory in the dockercomposetest directory</p>
<p>After everything is set in the dockercomposetest directory run</p> <p>Docker-compose up -d</p>
<p>Look in docker desktop. You should see something like</p> 
<p>Click on the mysports2020-db-1 container and review the logs</p> <p>Do the same for mysports2020-pas-1</p>
<p>Open a browser and run :</p> <p>http://localhost:8810/web/api/data/customers</p> <p>http://localhost:8810/web/api/data/customers/3000</p>
<p>To connect to the running sports2020 database you can also launch a openedge editor and connect using</p> <p>Connect sports2020 -S 10000</p>

Appendix A : Useful Docker commands

Command	Description
---------	-------------

docker version	Displays docker version
docker images	Displays the images in the host machine
docker image inspect	Shows the layers of the docker image
docker ps	Shows the running containers
docker run	Runs a container
docker exec	Can be used to obtain terminal access to a running container
docker stop	Stops a container
docker rm	Removes the container
docker rmi	Removes the image
docker build	Builds an image from a Docker file
docker pull	Pulls image to the host machine from docker repository
docker push	Pushes image to the docker repository
docker system prune	Cleans up space by removing unused images

Appendix B : Docker , Docker Desktop , JDK

How to install Docker Desktop,: <https://docs.docker.com/desktop/install/windows-install/>

JDK : [Latest Releases | Adoptium](#) Linux x64 JDK 17 LTS



Progress Software

Progress Software Corporation (NASDAQ: PRGS) is a global software company that simplifies the development, deployment and management of business applications on premise or on any Cloud, on any platform and on any device with minimal IT complexity and low total cost of ownership.

Worldwide Headquarters

For regional international office locations and contact information, please refer to the Web page below: <https://www.progress.com/company/offices>

Progress and [ALL PRODUCT NAMES LISTED IN PUBLICATION] are trademarks or registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and other countries. Any other marks contained herein may be trademarks of their respective owners. Specifications subject to change without notice.

© 2024 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.