



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

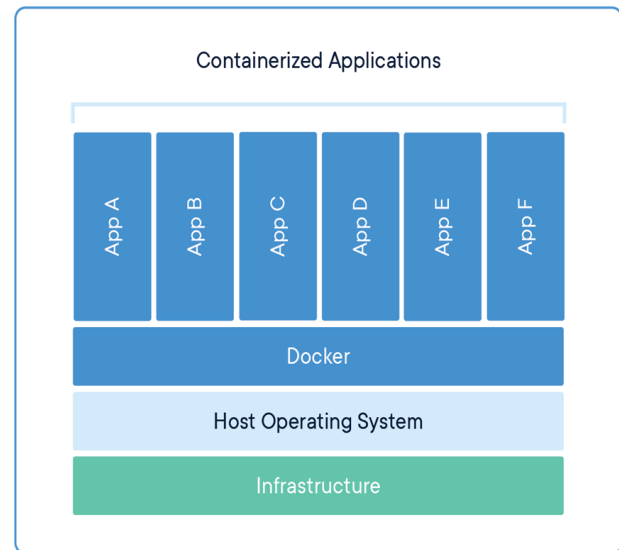
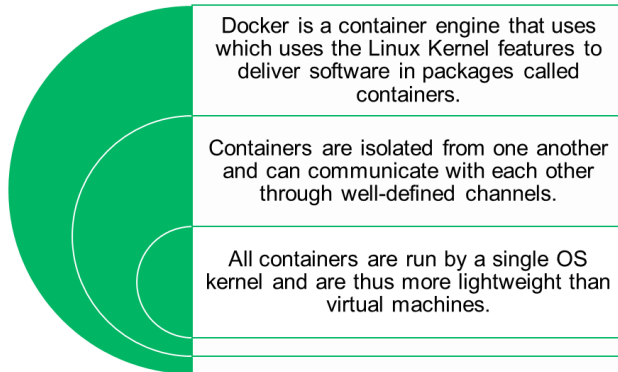


Table of Contents

Table of Contents	1
Introduction	2
Chapter 1 – Docker Desktop and CLI	2
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	6
LAB 4 – OpenEdge Container with oepas1 and deploy services	7
LAB 5 – OpenEdge Container with oepas1 and APSV	8
LAB 6 – Save your OpenEdge Container to an image	9
LAB 7 – Use the PASOE image from Progress ESD	10
LAB 8 – How to define multi container : docker-compose	10

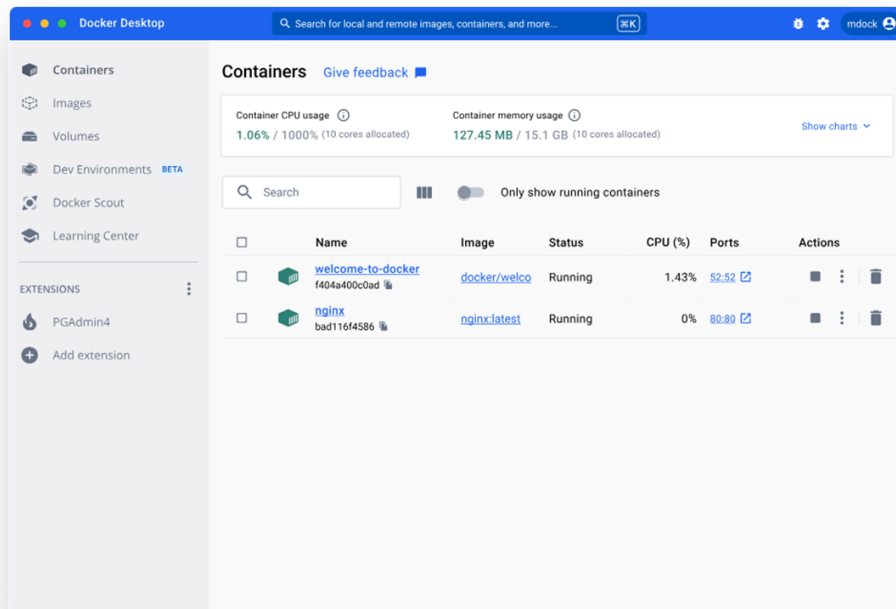
Introduction

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



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

On the command line : docker images

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
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 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

 **Run a new container**
image128:latest

Optional settings ^

Container name

A random name is generated if you do not provide one.

Ports

Host port

Container port
8810/tcp

Host port

8811/tcp

Host port

8820/tcp

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

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

Progress
Progress Application Server for OpenEdge

OpenEdge 12.8.3
Server has been running for 0 minutes

Tomcat Version	Apache Tomcat/10.1.19	JVM Vendor	Eclipse Adoptium
PAS Version	12.8.3	JVM Version	17.0.6+10
OS Name	Linux	Hostname	9206e95acbd9
OS Version	5.10.102.1-microsoft-standard-WSL2	IP Address	172.17.0.2
OS Architecture	amd64		

[MANAGE WEB APPLICATIONS](#) [MANAGE PAS FOR OPENEDGE](#)

- PAS for OpenEdge Quick Start Video
- PAS for OpenEdge Quick Start Guide
- PAS for OpenEdge Documentation
- OpenEdge Information Hub
- OpenEdge Development Community

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: "0x00000000000000061",
        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: "0x00000000000000062",
```

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 : <code>docker ps</code> 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, 0.0.0.0:8810-8811->8810-8811/tcp, 0.0.0.0:9090->9090/tcp, 8820/tcp container128services
In OpenEdge development on your host create a procedure named : <code>listprograms.p</code>					
OS-COMMAND SILENT "ls /psc/wrk/oepas1/openedge" >> /psc/wrk/oepas1/logs/myprograms.txt.					
Copy the procedure in the docker container <code>docker cp C:\OpenEdge128\WRK\LabsDocker\listprograms.p 9206e95acbd9:/psc/wrk/oepas1/openedge</code>					
In OpenEdge development on your host create a procedure to invoke the <code>listprograms.p</code> on <code>oepas1</code>					
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 : <code>myprograms.txt</code>					
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 <code>docker commit container128services saved128</code>
See the images in Docker Desktop, or run <code>docker images</code> You should see a new image you can use
Run a new container from this saved image
What should the next command do ? <code>docker save -o saved128.tar image128services</code>

LAB 7 – 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

From the LabsDocker folder extract the Progress_PASOE_Container. You will see a PROGRESS_PASOE_CONTAINER_IMAGE_12.8.3_LNX_64.tar.gz
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 8 – 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

version: '3.8'

services:

mysports2020-db:

image: \${DB_IMAGE}

volumes:

- \${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

Explanations :

mysports2020-db : will be a container to run a sports2020 database

mysports2020-pas : will be a container hosting a PASOE instance . This container will depend on mysports2020-db

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















Copy a provided src directory in the dockercomposetest directory

Copy a provided conf directory in the dockercomposetest directory

After everything is set in the dockercomposetest directory run

Docker-compose up -d

Look in docker desktop. You should see something like

<input type="checkbox"/>		dockercomposetest	-	Running (2/2)				
<input type="checkbox"/>		mysports2020-db-1 120cf3719f4b 	devbfvio/sports2020-db:12.8.3	Running	1 hour ago			
<input type="checkbox"/>		mysports2020-pas-1 6c8f4996c94a 	devbfvio/sports2020-pas:12.8.3	Running	1 hour ago			

Click on the mysports2020-db-1 container and review the logs

Do the same for mysports2020-pas-1

Open a browser and run :

<http://localhost:8810/web/api/data/customers>

<http://localhost:8810/web/api/data/customers/3000>

To connect to the running sports2020 database you can also launch a openedge editor and connect using
Connect sports2020 -S 10000

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

**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.