How to create your own Docker images for Denodo Platform Containers

Revision 20230623

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 INTRODUCTION

This article's aim is to explain how to build your own Docker images. This is not an introductory document to Containers so we assume that the reader has some knowledge of Docker.
This document applies to Denodo 8.0 update 20220815 and later.

2 BUILDING THE DOCKER IMAGES

In the following sections, we are going to describe how to build your own Docker images with the Denodo Platform installed.

Building an image for Denodo Platform involves:

1. Generating a Denodo Platform auto-installation XML file. You will need to download a Denodo Platform installer compatible with your OS.
2. Preparing a build folder with the auto-installation XML file, an zipped Denodo Platform installer for Linux, and the Dockerfile.
3. Creating a customized Dockerfile with the instructions to build your image using the Denodo Platform auto-installation file.
4. Executing the docker build command in the build folder.

2.1 GENERATING AN AUTO-INSTALLATION FILE

To generate an auto-installation file you must download and unzip the Denodo Platform installer for the OS (Linux, Windows) that you have installed on your local machine, independently of the OS file system you will use as the base in the image you intend to build. Follow the instructions to generate the auto-installation file depending on your installed OS. The generated file is named `install.xml` in the subsequent sections.

The installer sets the Denodo Platform installation path in the file according to the OS it was generated on. Update the installation path in the file with the syntax of the Linux base image that we will use to generate the Container image.
We assume in the rest of the document that the Denodo Platform will be installed in the path `/opt/denodo`.

2.2 PREPARING THE BUILD DIRECTORY

We need to create the following folder structure:

```
/denodo-docker-build/
│
├── builderfs/
│      ├── denodo-install/
│      │    ├── denodo-install-xxx.zip
│      │    └── install.xml
│      ├── denodo-update/
│      │     └── denodo-xxx-update-xxx.jar
│      └── container-scripts/
```

```
│          └── denodo-container-scripts-xxx.zip
│
└── Dockerfile
```

Fill the structure with the specific files:

- `/builderfs/denodo-install/`:
  - The zipped Denodo installer: denodo-install-xxx.zip
    - Must be the version compatible with your Linux distribution.
  - The auto-installation file, named install.xml
- `/builderfs/denodo-update/`:
  - Optionally include the jar corresponding to any Denodo update or hotfix to be installed
- `/builderfs/container-scripts/`:
  - The zipped scripts denodo-container-scripts.zip to be included into the container image. More details on these scripts in the section below
- The customized build file, named Dockerfile.

2.3 DENODO CONTAINER SCRIPTS

These scripts are specific for containers to allow an initial configuration before starting the executable process. The `ENTRYPOINT` instruction in the Dockerfile uses it to configure and launch the Denodo process. Furthermore, it ensures that the executable receives the Unix signals properly, to do some extra cleanup on shutdown.
The container configurations can be made by providing environment variables and mounting specific volumes. See the Denodo Docker container configuration for more information.

2.4 CREATING A DOCKERFILE

The following Dockerfile is a skeleton that may serve you to build your own customized one. Using the Debian:stable image as the base, it copies the Denodo Platform Linux installer that you have already downloaded and the generated auto-installation file from your host to the image file system to install Denodo Platform on a container. It uses a multi-stage build to optimize the Dockerfile and reduce the size of the final image.

```
### Stage 1 - install ###
# Review latest debian version here https://hub.docker.com/_/debian/
FROM debian:stable as builder-installer

RUN apt-get update \
 && apt-get install -y \
    zip \
 && rm -rf /var/lib/apt/lists/*

# The Denodo installation requires the denodo installer, the update
(optional) and the install.xml file for unattended installation.
COPY builderfs /opt
```

```
WORKDIR /opt/denodo-install

RUN unzip \*.zip \
 && mv */* .\
 && chmod +x installer_cli.sh \
 && ./installer_cli.sh install --autoinstaller install.xml \
 && rm -r /opt/denodo-install


### Stage 2 - update ###
FROM builder-installer as builder-updater

WORKDIR /opt/denodo-update
# PATH of JRE. This will be the default JRE used to run Denodo Platform.
ENV PATH="/opt/denodo/jre/bin:${PATH}"
#  || : will force a zero exit status
RUN find /opt/denodo-update/ -name "denodo*jar" | sort | while read file;
do echo "Applying ${file##*/}"; yes | java -jar ${file} /opt/denodo -c;
done \
 && cd /opt/denodo \
 && rm -r /opt/denodo-update \
 && find . -name \*.back.* -type f -delete


### Stage 3 - builder-scripts ###
FROM builder-updater as builder-scripts

WORKDIR /opt/container-scripts
RUN mkdir -p /opt/denodo/tools/container \
 && unzip \*.zip -d /opt/denodo/tools/container \
 && sed -i "s/\r$//" /opt/denodo/tools/container/*.sh \
 && chmod +x /opt/denodo/tools/container/*.sh \
 && rm -r /opt/container-scripts


### Stage 4 - final ###
FROM debian:stable

ENV DENODO_HOME=/opt/denodo

# Install required libs for some components like denodo-monitor, ...
RUN apt-get update \
 && apt-get install -y \
    net-tools \
    procps \
    curl \
    jq \
 && rm -rf /var/lib/apt/lists/*

# Create the system denodo user and group
RUN groupadd -r denodo && useradd --no-log-init -m -r -g denodo -c
"Denodo user" denodo

COPY --from=builder-scripts --chown=denodo /opt/denodo/ /opt/denodo/

RUN /opt/denodo/tools/container/postunpack.sh
```

```
# Sets the workdir
WORKDIR $DENODO_HOME

VOLUME   [   "/denodo",   "/container-entrypoint-preinit",   "/container-
entrypoint-init", "/container-entrypoint-prestop" ]

USER denodo
ENTRYPOINT [ "/opt/denodo/tools/container/entrypoint.sh" ]
CMD ["--help"]
```

2.5 EXECUTING THE DOCKER BUILD COMMAND

To generate your new Denodo Platform image you need to open a command line on the recently created build directory and run the following command:

```
$ docker build -t denodo-image:latest .
```

Please note the space period at the end of the build command. Check that the execution of this command has occurred without errors.

2.6 DEPLOYING IN RED HAT OPENSHIFT

The Dockerfile template provided before uses a Debian image as the base system. However, if you want to deploy the container in Red Hat OpenShift you must base your container image on Red Hat Universal Base Image (UBI).

It is possible to create the Denodo container based on a different image base by just replacing the text *debian:stable* in the Dockerfile with your preferred UBI image.

For example, if you want to use ubi8 you can replace the line in the template that indicates:

```
FROM debian:stable
```

with:

```
FROM registry.access.redhat.com/ubi8/ubi:latest
```

Replace `apt-get` with `yum` and:

```
rm -rf /var/lib/apt/lists/*
```

with:

```
yum clean all -y
```

OpenShift Container Platform runs containers using an arbitrarily assigned user ID. For an image to support running as an arbitrary user, directories and files that may be written to by processes in the image should be owned by the root group and be read/writable by that group. Files to be executed should also have group execute permissions.

In order to run Denodo using an arbitrarily assigned user ID you need to add the following to your Dockerfile to allow users in the root group to access the Denodo files:

```
RUN chgrp -R 0 /opt/denodo && chmod -R g=u /opt/denodo
```

A complete example:

```
### Stage 1 - install ###
# Review latest ubi version
FROM    registry.access.redhat.com/ubi8/ubi:latest    as    builder-
installer

RUN yum -y update \
 && yum -y install \
    zip \
 && yum clean all -y

# The Denodo installation requires the denodo installer, the update
(optional) and the install.xml file for unattended installation.
COPY builderfs /opt

WORKDIR /opt/denodo-install

RUN unzip \*.zip \
 && mv */* .\
 && chmod +x installer_cli.sh \
 && ./installer_cli.sh install --autoinstaller install.xml \
 && rm -r /opt/denodo-install


### Stage 2 - update ###
FROM builder-installer as builder-updater

WORKDIR /opt/denodo-update
# PATH of JRE. This will be the default JRE used to run Denodo
Platform.
ENV PATH="/opt/denodo/jre/bin:${PATH}"
#  || : will force a zero exit status
RUN find /opt/denodo-update/ -name "denodo*jar" | sort | while read
file; do echo "Applying ${file##*/}"; yes | java -jar ${file}
/opt/denodo -c; done \
 && cd /opt/denodo \
 && rm -r /opt/denodo-update \
 && find . -name \*.back.* -type f -delete


### Stage 3 - builder-scripts ###
FROM builder-updater as builder-scripts

WORKDIR /opt/container-scripts
RUN mkdir -p /opt/denodo/tools/container \
```

```
 && unzip \*.zip -d /opt/denodo/tools/container \
 && sed -i "s/\r$//" /opt/denodo/tools/container/*.sh \
 && chmod +x /opt/denodo/tools/container/*.sh \
 && rm -r /opt/container-scripts \
 && chgrp -R 0 /opt/denodo && chmod -R g=u /opt/denodo

### Stage 4 - final ###
FROM registry.access.redhat.com/ubi8/ubi:latest

ENV UID=1001 DENODO_HOME=/opt/denodo

# Install required libs for some components like denodo-
monitor, ...
RUN yum -y update \
 && yum -y install \
    net-tools \
    procps \
    curl \
    jq \
 && yum clean all -y

# Create the system denodo user and group
RUN groupadd -r denodo && useradd --no-log-init -m -r -g denodo -G
root -c "Denodo user" -u $UID denodo \
     && mkdir -p /opt/denodo && chown $UID:0 /opt/denodo && chmod
g=u /opt/denodo

COPY     --from=builder-scripts     --chown=$UID:0     /opt/denodo/
/opt/denodo/


# Sets the workdir
WORKDIR $DENODO_HOME

VOLUME [ "/denodo", "/container-entrypoint-preinit", "/container-
entrypoint-init", "/container-entrypoint-prestop" ]

## Specify the user with UID
USER $UID
ENTRYPOINT [ "/opt/denodo/tools/container/entrypoint.sh" ]
CMD ["--help"]
```

3 TESTING THE IMAGE

3.1 DENODO LICENSES

Denodo Platform requires a license to run. If you intend to use a license managed by the License Manager you can run the container with the following environment variables:

- `DENODO_LM_PROTO`: To indicate the protocol scheme: http or https. If this environment variable is not present, it takes the default value http.
- `DENODO_LM_HOST`: To indicate the host of the License Manager.
- `DENODO_LM_PORT`: To indicate the port of the License Manager. If the environment variable is not present, it takes the default value 10091.

```
$ docker run -d -h denodo-vdpserver \
 -p 9999:9999 -p 9997:9997 -p 9996:9996 -p 9995:9995 -p 9090:9090 \
 -e DENODO_LM_PROTO=http \
 -e DENODO_LM_HOST=host.docker.internal \
 -e DENODO_LM_PORT=10091 \
 --name denodo-vdpserver denodo-platform:latest --vdpserver
```

Alternatively, you can map this configuration file or a Denodo standalone license (like the Evaluation or Denodo Express licenses) from your host to a container using the Docker volume mount parameters.

```
$ docker run -d -h denodo-vdpserver \
 -p 9999:9999 -p 9997:9997 -p 9996:9996 -p 9995:9995 -p 9090:9090 \
 -v <path>:/opt/denodo/conf/denodo.lic \
 --name denodo-vdpserver denodo-platform:latest --vdpserver
```

where:

`-v <path>:/opt/denodo/conf/denodo.lic`

Specifies the Standalone Denodo License file to use by the Denodo Server, where:

- `<path>` - Path for the license file in the host OS, for example: `/home/denodo/denodo.lic`
- `/opt/denodo/conf/denodo.lic` - Path for the license file in the container (Do not change this).

3.2 STARTING SEVERAL SERVICES

The container image is parametrized to indicate arguments for starting several services. For example:

```
$ docker run -d -h denodo-vdpserver -p 9999:9999 -p 9997:9997 -p
9996:9996 -p 9995:9995 -p 9090:9090 -v /mnt/c/tmp/conf:/denodo/conf
--name denodo-vdpserver denodo-platform:latest --vdpserver
--designstudio --datacatalog
```

These are the available arguments for the Denodo Platform image:

```
--vdpserver, --designstudio, --datacatalog, –dmt, –monitor, --schindex,
--schserver, --schadmin
```

As an alternative, you can manually start these servers with the `docker exec` command after the Denodo Platform container has been started. For example, you can start the data catalog in a running Denodo container named “`denodo`” with the command:

```
$ docker exec -t denodo ./bin/datacatalog_startup.sh
```

3.3 MOUNTING EXTERNAL VOLUMES

As this is a non-root container, the mounted files and directories must have the proper permissions for the user denodo (UID 999 by default).

4 INSTALLING A NEW UPDATE

Denodo delivers new updates regularly to improve the user experience of the Denodo Platform. When a new update is released, the Denodo container that is available to download from the Denodo Support Site gets updated to the new version, so there is always a container with the latest version of Denodo ready to download.

However, there might be situations where you would like to build your own container using a specific Denodo version that does not match that version available in the Support Site.

In those cases, you can use the previous builder folder structure to locate the corresponding jar file for updates or hotfixes at `/builderfs/denodo-update/` and build the new image. Take into account that if there are several .jar files, they will be applied by alphabetical order.

You can build and test the image by using the same commands already provided in the previous sections of this article.