NetIQ / Microfocus / OpenText Identity Manager Designer docker installation

23.02.25

Pegasi Knowledge
https://ghost.pegasi.fi/wiki/

Table of Contents

NetIQ / Microfocus / OpenText Identity Manager Designer docker installation

Updates

- 22.07.2024 updated to use Rocky Linux 9 (RHEL9 clone) due to Designer 4.9. Mitigated a bug with Designer 4.9 and display usage. Added guide to Designer sudoers setup to make passwordless docker start. Added desktop shortcut file option.
- 18.07.2024 moved sudo inside wrapper script, moved container option below preferred wrapper option, updated wrapper script
- 12.07.2024 added sysctl max file limit raise. designer with larger projects eats files.
- 15.07.2024 wrapper script to run with sudo: you may create the container but you can actually do better by just creating image and running instance with wrapper script

General

Designer for OpenText NetIQ Identity Manager is an excellent Eclipse based IDM design and simulation tool which enables design, testing, documenting and deploying in a very convenient way. This guide shows how to install and use this tool from Docker container using any Linux distribution with docker support.

Problem

Designer is a beautiful tool for identity management work but it is certified only to be used in SLES or RHEL environments only which may not be the first choices of Linux for every expert as many prefer Ubuntu, Fedora, Arc and OpenSuse environments. Especially if one works with laptop.

My problem rose out when I decided to switch my laptop strategy from enterprise laptops to those sexy bleeding edge consumer ones and got me 2024 Asus Zenbook oled which requires 6.8 kernel to even boot up. Kernel 6.8 is light years ahead of any RHEL or SLES distro so I ended up putting Fedora 40 to my sleek laptop. Of course Designer did not work there so I could try match libraries and hack it to work somehow but I wanted to make more permanent solution.

Solution

Docker can cater any Linux environment in a more portable way and we can do just that by setting up a container using Rocky Linux 8 which is a clone of RHEL8 without that one branding package. In

addition to that we get a very convenient image versioning, upgradeability and **a very easy** rollback ability.

I write this using Fedora 40 as host but the effect of host distro is quite minimal.

Multiple Designer instances

As a bonus feature we have capability to run multiple designer instances using same image with multiple docker configurations. Also the base OS install is better protected to malicious attacks do to image read only property and our designer app is easy to transfer to other hosts.

Preparations

Make sure you are on GUI desktop to be able to run Designer. Not text console without display redirect.

```
sudo su
```

Install docker

Base install stuff.

```
dnf in docker
systemctl enable docker --now
sudo usermod -aG docker <user>
xauth add ${DISPLAY} . $(mcookie) #generate .Xauthority file so docker
volume mount will not make it a directory
```

Setup Designer medias and folders

Summary

- Acquire Designer install package from OpenText
- Make necessary directories for container
  - actual install destination
  - workspace
  - installation source
- Extract installer to be available to the container

Actual actions - excluding designer package which you must download from OpenText

```
mkdir /opt/designer
mkdir /opt/designer/designer
mkdir /opt/designer/designer_workspace
cd /opt/designer
tar -xzf /xxx/Identity_Manager_4.9.0_Designer_Linux.tar.gz
```

If you are using SELinux set file contexts to the directories with

```
semanage fcontext -a -t container_file_t '/opt/designer/designer'
semanage fcontext -a -t container_file_t '/opt/designer/designer_workspace'
semanage fcontext -a -t container_file_t '/opt/designer/designer_install'
restorecon -R /opt/designer/designer*
```

We now have /opt/designer with empty sub directory designer*workspace, designer*install and Designer installer at designer_install

Docker build

This is just workstation install and we could optimize this a lot by just installing minimal libraries needed by Designer. Create /opt/designer/Dockerfile:

```
FROM rockylinux:9
RUN dnf -y groupinstall "Workstation" --allowerasing && \
    dnf -y install libcanberra libcanberra-gtk2 glibc-*.i686 libgcc-*.i686
libXtst-*.i686 libXrender-*.i686 libXi-*.i686 unzip bc lsof net-tools
ncurses-libs && \
    dnf clean all
```

Build the image with

```
docker build . -t designer-base-rocky-9
```

Designer install with SELinux

You can skip this if not using SELinux. If using SELinux we need to do following

- setenforce 0
- install and run designer
- create rule to selinux
- setenforce 1 and run again

Set SELinux to permissive.

```
setenforce 0
```

Create new container with display redirection and correct folder mounts:

```
docker run --name designer-app-temp -it --rm --net=host --env
DISPLAY=$DISPLAY --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" --
volume="/home/<username>/.Xauthority:/root/.Xauthority:rw" --
volume="/opt/designer/designer_workspace:/root/designer_workspace" --
volume="/opt/designer/designer:/root/designer" --
volume="/opt/designer/designer_install:/opt/designer_install" designer-base-
rocky-9 /bin/bash
```

Now in container prompt raise max files limit for Designer, install Designer and run it. Update it all the way and DO NOT EXIT the container:

```
echo "root soft nofile 1024000" > /etc/security/limits.d/designer.conf
echo "root hard nofile 1024000" >> /etc/security/limits.d/designer.conf
setenforce 0 #you can disable selinux inside container at least when
installing
#also maybe you want to set /etc/selinux/config SELINUX=permissive. perhaps
not needed.
dnf up
/opt/designer_install/install
/root/designer/StartDesigner.sh
```

Then we catch SELinux errors, create module and add it:

```
journalctl -e -t audit | grep denied | audit2allow -M designer-docker
cat designer-docker.te #verify we got containerd rules
semodule -i designer-docker.pp
setenforce 1
```

If the above somehow fails you can try following:

```
journalctl -t audit -e
ausearch -m avc -ts recent > selinux.log
vim selinux.log #delete irrelevant lines
cat selinux.log | audit2allow -M designer-docker
semodule -i designer-docker.pp
```

Continue to finalizing Designer installation and DO NOT EXIT the container.

Designer install without SELinux

Create new container with display redirection and correct folder mounts:

```
docker run --name designer-app-rocky-9 -it --rm --net=host --env
DISPLAY=$DISPLAY --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" --
volume="$HOME/.Xauthority:/root/.Xauthority:rw" --
volume="/opt/designer/designer_workspace:/root/designer_workspace" --
volume="/opt/designer/designer:/root/designer" --
volume="/opt/designer/designer_install:/opt/designer_install" designer-base-
rocky-9
```

Now in container prompt start installer, run designer and let it update itself and packages and DO NOT exit the container yet:

```
/opt/designer_install/install
/root/designer/StartDesigner.sh
```

Finalizing Designer installation

Lets commit changes to new image and set container to run Designer on start by opening another terminal with root access and commit changes as new image.

```
docker commit --change='CMD /root/designer/StartDesigner.sh' [CONTAINER_ID]
designer-rocky-9-image-v1
```

Passwordless start

To start Designer container without sudo password you can add your logins or group to /etc/sudoers:

```
<user> ALL=(ALL) NOPASSWD: /usr/bin/docker
```

Then remove “sudo” from the beginning of row “sudo docker run…” and you can start Designer with ease.

Run with wrapper script

This seems to be the most usable way to run Designer: from image without creating permanent container keeping all data in designer volumes. This enables easier use of updated images with multiple users and will be my goto approach since ready made containers want to bind themselves to one user display.

Wrapper script rundesigner.sh, run with “./rundesigner.sh”:

```
#!/bin/bash

if [ -z "$1" ]; then
  display_var=$DISPLAY
else
  display_var=$1
fi

container_name="designer-$USER-49-removable"

if [ $(docker ps -q -f name=^/${container_name}$) ]; then
  echo "Container already running. Use different name if this instance needs
to run in parallel."
  exit
fi

xhost +local:docker

sudo docker run --name $container_name -it --rm --net=host \
        --env DISPLAY=$DISPLAY \
        --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" \
        --volume="/home/$USER/.Xauthority:/root/.Xauthority:rw" \
        --volume="/opt/designer/designer_workspace:/root/designer_workspace"
\
        --volume="/opt/designer/designer:/root/designer" \
        --volume="/opt/designer/designer_install:/opt/designer_install" \
        designer-base-rocky-9-designer-v1 /root/designer/Designer.sh

xhost -local:docker
```

Desktop shortcut

To access easy with search you can make a desktop shortcut. Be aware that you must have xterm installed and if you do not do the sudoers change above do not use the geometry options so you can type the sudo password to the terminal. Also the designer logo must be copied manually.

Create ~/.local/share/applications/designer.desktop (replace <user> with your username):

```
[Desktop Entry]
Version=1.0
Type=Application
Name=Designer
Comment=Designer application Docker
Icon=/opt/designer/designer.png
Exec=xterm -geometry 1x1-0-0 -e '/home/<user>/bin/run_designer.sh'
```

```
Path=/home/<user>
Terminal=false
Categories=Development;Utility;
```

And run:

```
update-desktop-database ~/.local/share/applications/
```

Run with permanent container

To start Designer when container is started we can make it into container which can be started and stopped. This does not bring much advantage but lets you easily see which containers exist and maybe more manageable.

Add xhost always first:

```
xhost +local:docker
```

Make container:

```
sudo docker run --name designer-app -it --net=host \
    --env DISPLAY=$DISPLAY \
    --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" \
    --volume="$HOME/.Xauthority:/root/.Xauthority:rw" \
    --volume="/opt/designer/designer_workspace:/root/designer_workspace" \
    --volume="/opt/designer/designer:/root/designer" \
    --volume="/opt/designer/designer_install:/opt/designer_install" \
    designer-image-v3
```

Exit Designer and run again just by starting container:

```
docker start designer-app
```

Minor Designer updates

Designer updates should land normally since the files live under /opt/designer.

OS updates and minor Designer updates

Steps:

1. run the container with Designer, do not exit Designer

2. connect to shell with docker exec -it designer-app bash
3. dun dnf update
4. use another shell to commit container image to new version
5. remove current container
6. add new container using new image

```
#while container running update designer or enter container shell and do dnf
update
docker commit [CONTAINER_ID] designer-image-vN #replace N with version
number
docker rm designer-app
docker run --name designer-app -it --net=host --env DISPLAY=$DISPLAY --
volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" --
volume="/opt/designer/designer_workspace:/root/designer_workspace" --
volume="/opt/designer/designer:/root/designer" --
volume="/opt/designer/designer_install:/opt/designer_install" designer-
image-vN
```

Major Designer updates

- Start new container from image in interactive mode
- Do necessary uninstalls, installs, updates
- Set SELinux to permissive if problems occur
- Do not exit container
- Commit the image to new version
- Set container to use new image
- Tune SELinux policies using above help

Update software in docker container interactive session

```
setenfoce 0
docker run -it --name designer-app-vN -it --net=host --env DISPLAY=$DISPLAY
--volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" --
volume="/opt/designer/designer_workspace:/root/designer_workspace" --
volume="/opt/designer/designer:/root/designer" --
volume="/opt/designer/designer_install:/opt/designer_install" designer-
image-vN /bin/bash
dnf up
#remove old designer, install new designer, update everything
#do not exit container
```

Generate new image

Do not exit the container interactive session. Just exit Designer so that you are in bash session. Use another terminal to commit docker to another image.

```
docker commit --change='CMD /root/designer/StartDesigner.sh' [CONTAINER_ID]
designer-image-vN #replace N with version number
docker rm designer-app #remove previous container UNLESS you want access to
older version
```

Exit docker interactive session and generate new container

After committing changes to new image exit the interactive session, remove intermediate container and re-create it using new image.

```
docker rm designer-app-vN
docker run --name designer-app-vN -it --net=host --env DISPLAY=$DISPLAY --
volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" --
volume="/opt/designer/designer_workspace:/root/designer_workspace" --
volume="/opt/designer/designer:/root/designer" --
volume="/opt/designer/designer_install:/opt/designer_install" designer-
image-vN
```

Remember to set setenforce 1 if you had to disarm it.

Error situations

If you cannot start the container / image here are some basic commands to start with.

Start image with shell

Set xhost to make sure we can use GUI from docker container.

```
xhost +local:docker
```

Start image in bash shell and test there.

```
sudo docker run --name designer-app -it --net=host \
    --env DISPLAY=$DISPLAY \
    --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" \
    --volume="$HOME/.Xauthority:/root/.Xauthority:rw" \
    --volume="/opt/designer/designer_workspace:/root/designer_workspace" \
    --volume="/opt/designer/designer:/root/designer" \
```

```
    --volume="/opt/designer/designer_install:/opt/designer_install" \
    designer-image-v3 /bin/bash
```

Delete and reinstall Designer.

```
xhost +local:docker
cp -a /opt/designer/designer /backuplocation/
rm -rf /opt/designer/designer /opt/designer/designer/.eclipse*
sudo docker run --name designer-app -it --net=host \
    --env DISPLAY=$DISPLAY \
    --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" \
    --volume="$HOME/.Xauthority:/root/.Xauthority:rw" \
    --volume="/opt/designer/designer_workspace:/root/designer_workspace" \
    --volume="/opt/designer/designer:/root/designer" \
    --volume="/opt/designer/designer_install:/opt/designer_install" \
    designer-image-v3 /bin/bash
/opt/designer_install/install
```

Designer will not start / throws errors

You may see this error:

```
(java:2683): Gdk-ERROR **: 12:28:44.579: The program 'java' received an X
Window System error.
This probably reflects a bug in the program.
The error was 'BadValue (integer parameter out of range for operation)'.
  (Details: serial 192 error_code 2 request_code 130 minor_code 3)
  (Note to programmers: normally, X errors are reported asynchronously;
   that is, you will receive the error a while after causing it.
   To debug your program, run it with the --sync command line
   option to change this behavior. You can then get a meaningful
   backtrace from your debugger if you break on the gdk_x_error() function.)
```

This is some sort of bug in how Designer binary, which executes after StartDesigner.sh, loads Designer.ini settings and executes java process. You can go around this by bypassing Designer binary and putting the java command directry to StartDesigner.sh or making your own wrapper. Here is example of my own wrapper Designer.sh. Please note the larger heap sizes:

```
#!/bin/sh
cd /root/designer
LD_LIBRARY_PATH=/root/designer/plugins/com.novell.core.iconeditor_4.0.0.2024
05241531/os/linux/x86_64:/root/designer/plugins/com.novell.core.jars_4.0.0.2
02405241531/os/linux/x86_64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

```
MAGICK_CONFIGURE_PATH=/root/designer/plugins/com.novell.core.iconeditor_4.0.
0.202405241531/os/linux/x86_64/ImageMagick-6.2.6/config
export MAGICK_CONFIGURE_PATH
GDK_NATIVE_WINDOWS=true
export GDK_NATIVE_WINDOWS
/root/designer/jre/bin/java -Xms10240m -Xmx20480m -Dfile.encoding=UTF-8 -
Dcom.sun.jndi.ldap.object.disableEndpointIdentification=true -
Dorg.eclipse.swt.browser.XULRunnerPath=/root/designer/plugins/com.novell.cor
e.jars_4.0.0.202405241531/os/linux/xulrunner -
Djava.util.Arrays.useLegacyMergeSort=true -
Dorg.eclipse.swt.internal.gtk.disablePrinting -jar
/root/designer/plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
-os linux -ws gtk -arch x86_64 -showsplash -launcher /root/designer/Designer
-name Designer --launcher.library
/root/designer/plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.200
.v20140603-1326/eclipse_1605.so -startup
/root/designer/plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
--launcher.overrideVmargs -exitdata 3 -clean -vm /root/designer/jre/bin/java
-vmargs -Xms10240m -Xmx20480m -Dfile.encoding=UTF-8 -
Dcom.sun.jndi.ldap.object.disableEndpointIdentification=true -
Dorg.eclipse.swt.browser.XULRunnerPath=/root/designer/plugins/com.novell.cor
e.jars_4.0.0.202405241531/os/linux/xulrunner -
Djava.util.Arrays.useLegacyMergeSort=true -
Dorg.eclipse.swt.internal.gtk.disablePrinting -jar
/root/designer/plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
```

You can then execute this command on docker run wrapper instead of StartDesigner.sh.