VersionVault and Docker Containers

Author: Nancee Buckle
Date: December 17, 2020

Contents

1 Introduction

VersionVault 2.0.1 supports the ability to deploy a Linux container capable of using dynamic views. HCL VersionVault users often have the requirement to support versions of their products for a specific time frame. These users might have environments dependent on a specific tool set and/or system libraries. Some of these environments might be required for everyday use while others are required only periodically. For example, consider a chip designer who uses VersionVault to version their chip layouts. If the chip designer is required to view a chip layout using an older version of a tool, they might have to find or create a new system supporting that tool. Rather than maintaining different systems supporting different tool sets, a VersionVault user might prefer to create a Docker container that could access data stored in VersionVault.

The following sections outline the requirements for deploying a Docker container either as a dynamic view client or with view-extended access. Details for the Docker host as well as the Docker container in each of these use cases are provided in sections 2 and 3, along with detailed examples in section 4.

2 Docker Container with Dynamic View Client Access

VersionVault provides the ability to access VOB elements dynamically through the multiversion file system (MVFS), which is installed in the underlying host system (Docker host). Docker containers do not allow you to install file systems or drivers but do allow the containers to share file systems and drivers with the Docker host. To share the MVFS and MVFS device, the Docker host should minimally have VersionVault 2.0.1 installed and configured to run as a VersionVault dynamic view client. In this configuration, VersionVault in the Docker host will share the MVFS device, along with volumes to provide the container access to VOB and view storage, the viewroot, and VOB mount points.

VersionVault can either be installed in a Docker container or share VersionVault binaries with the Docker host. If the Docker container shares binaries with the Docker host, certain VersionVault configuration files will need to be copied into the container and/or customized to the environment for the container. Scripts can be created that run in the container after it is deployed to complete the pre-requisite configuration steps for VersionVault usage.

It is important to note that to facilitate access from the containers, views and VOBs should be configured with fully specified host/gpath/hpath information, so that both hpath and gpath are global pathnames to the database storage directories.

The following sections discuss requirements for deploying a Linux container capable of using VersionVault dynamic views.

2.1 Docker Host

2.1.1 VersionVault Installation

To share the MVFS and MVFS device, the Docker host should minimally have VersionVault 2.0.1 installed and configured to run as a VersionVault dynamic view client.

For more information on how to deploy VersionVault, refer to the VersionVault documentation.

2.1.2 NFS and Linux Automounter

VersionVault uses NFS and the Linux automounter to access view or VOB storage. The Docker host system must be configured for NFS and the Linux automounter.

2.1.3 Views

When creating views to be used within Docker containers, you should specify the options: -host, -hpath, and -gpath. Both hpath and gpath should be the global pathname of the view storage directory.

For example, suppose you would like to create a view residing on a machine whose hostname is build1. Your Docker host and view server host, **build1**, uses the automounter. To create a view to be used with a Docker container, you would do the following:

```
cleartool mkview -tag bld_view1 -host build1 \
-hpath /net/build1/user/viewstorage/bld_view1.vws
-gpath /net/build1/user/viewstorage/bld_view1.vws \
/net/build1/user/viewstorage/bld_view1.vws
```

This is required if the view resides on the Docker host or Docker container.

It is recommended that Docker containers use views external to the container since depending on how the container is configured, both storage and hostname can be temporary.

2.1.4 VOBs

When creating VOBs to be used with Docker containers, you should specify the options: -host, -hpath, and -gpath. Both hpath and gpath should be the global pathname of the VOB storage directory.

For example, suppose you would like to create a VOB residing on a system whose hostname is vvault_vserver. Your Docker host and vob server host, **vvault_vserver**, uses the automounter. To create a VOB to be used with a Docker container, you would do the following:

```
cleartool mkvob  -tag /vobs/vob1  -host vvault_vserver \
-hpath /net/vvault_vserver/user/vobstorage/vob1.vbs
```

```
        -gpath /net/vvault_vserver/user/vobstorage/vob1.vws \
        /net/vvault_vserver/user/vobstorage/vob1.vws
```

This is required if the VOB resides on the Docker host or Docker container.

It is recommended that Docker containers use VOBs external to the Docker container since depending on how the container is configured, both storage and hostname can be temporary.

2.2 Docker Container

The following sections describe how to build and deploy a Docker container capable of setting into a dynamic view and executing cleartool commands.

2.2.1 Docker Linux Base Image

The Docker Linux image must be supported by the VersionVault release running on the Docker host and should be a Base Linux Image that supports the init process. Both RedHat and SUSE provide Base images within their registries.

2.2.2 Linux Packages

Linux Base Images might not contain all the packages required to run VersionVault. The user must update the container to install these packages. Refer to Technotes 535653, 887639 and 718343.

2.2.3 View and VOB Storage

VersionVault requires access to both the view and VOB storage directories. Containers can access these storage directories by using shared volumes.

Consider the scenario where a container would like to access a VOB element in `/vobs/vob1` using the view, **build1**. The storage directory for the VOB is on `/net/vvault_vserver/user/vobstorage/vob1.vbs`. The storage directory for the view is on `/net/build1/user/viewstorage/bld_view1.vws`. Access to both the VOB and view storage location can be granted by starting up the container with the following option:

```
        --volume /net:/net:shared
```

2.2.4 MVFS file system

The MVFS device, MVFS viewroot (`/view`), and MVFS VOB mount points (`/vobs/vob1`) must be shared with the Docker container. If all VOBs reside under the directory `/vobs`, it is sufficient to share the `/vobs` directory.

Access to the MVFS device, viewroot, and VOB mount points can be granted by starting up the container with the following options:

```
        --device /view/.specdev:/dev/mvfs
        --volume /view:/view
        --volume /vobs:/vobs:shared
```

2.2.5 Privilege Mode

The Docker container requires being started in privileged mode. This is provided by the option:

```
        --privileged
```

2.2.6 Capabilities

The Docker container requires being started with SYS_ADMIN capabilities. This is provided by the option:

```
--cap-add “SYS_ADMIN”
```

2.2.7 VersionVault Installation

2.2.7.1 Installation of VersionVault

VersionVault can be installed within a container. The installation should be a server installation (not a minimal or full developer installation) since containers should not install and start up the MVFS file system.

2.2.7.2 Shared Docker Host VersionVault Installation

Rather than installing VersionVault within a container, the container can use the VersionVault binaries available on the Docker host and update configuration files residing in the Docker container.

2.2.7.2.1 Shared Binaries

The Docker container can share the Docker host VersionVault binaries by sharing volumes.

For example, if VersionVault is installed in the directory `/opt/hcl` and the Java JDK is installed on `/opt/jdk`, to share binaries with the Docker host, the following directories must be shared:

- `/opt/hcl:/opt/hcl`
- `/opt/rational:/opt/rational`
- `/usr/atria:/usr/atria`
- `/opt/jdk:/opt/jdk`

2.2.7.2.2 VersionVault Configuration Files

The directory, `/var/adm/hcl/versionvault/config`, contains the following files:

- `albd_rt_params.conf`
- `admin.conf`
- `cacert.pem`
- `snapshot.conf`
- `albd.conf`
- `cacert.pem.template`
- `vob_scrubber_params`
- `albd_lad.conf`

These files should be copied into the Docker container from the `config` directory of the Docker host.

2.2.7.2.3 Registry Host Configuration File

The registry configuration file, `/var/adm/hcl/versionvault/config/rgy_hosts.conf`, must be created in the Docker container to provide the registry hostname. If the Docker container is using the same registry host as the Docker host, this can be just a copy of the Docker host file.

2.2.7.2.4 Registry Region Configuration File

The registry configuration file, `/var/adm/hcl/versionvault/config/rgy_region.conf`, must be created in the Docker container to provide the registry region information. If the Docker container is using the same registry region as the Docker host, this can be just a copy of the Docker host file.

2.2.7.2.5 License File Configuration File

The license configuration file, `/var/adm/hcl/versionvault/config/fne_config`, must be created in the Docker container with the License URL and License Server ID. If the Docker container is using the same license as the Docker host, this can be just a copy of the Docker host file.

2.2.7.2.6 VersionVault Startup Script

VersionVault can be started within the container by using the following command:

```
/opt/hcl/ccm/versionvault/etc/versionvault start
```

3 Docker Container with View-Extended Access

For Docker containers, whose Linux Base image is not supported by the VersionVault release running on the Docker host, the user can access VOB elements using view-extended access.

The Docker host has minimally VersionVault 2.0.1 installed and configured to run as a VersionVault dynamic view client. A container is configured to allow access to views and VOBs on the Docker host system. The Docker host must share its viewroot, views storage and VOBs storage. The VersionVault binaries are not shared, and no VersionVault configuration files need to be copied or customized. Software in the container will be limited to accessing only VOB elements. No other VersionVault operations are possible in the container.

The following sections discuss the requirements for deploying Linux Docker containers capable of accessing VOB elements with view-extended pathnames.

3.1 Docker Host

3.1.1 VersionVault Installation

The Docker host must have VersionVault installed and configured to run as a VersionVault dynamic view client.

3.1.2 Views

3.1.2.1 View Creation

Views accessed through view-extended mode from within a Docker container must be created using the options: -host, -hpath, and -gpath. Both hpath and gpath must be the global pathname of the view storage directory.

3.1.2.2 View Startup

Views accessed via view-extended mode within the Docker container must be started on the Docker host.

3.1.3 VOBs

3.1.3.1 VOB Creation

VOBs accessed within the Docker container should be created using the options: -host, -hpath, and -gpath. Both hpath and gpath must be the global pathname of the VOB storage directory.

3.1.3.2 VOB Mount

VOBs accessed from within the Docker container must be mounted on the Docker host.

3.2 Docker Container

3.2.1 Views and VOBs Storage

The Docker container must have access to both the view and VOB storage directories. Access can be granted by starting the container with shared volumes. For a Docker Host that uses the automounter to access view and VOB storage directories, the Docker container can be started up with the following option:

```
--volume /net:/net:share
```

3.2.2 Privileged Mode

The Docker container requires running in privileged mode. This is provided by the following option:

```
--privileged
```

3.2.3 Viewroot Mount

The Docker container must be started sharing the viewroot mount for the Docker host. This can be done using the following option:

```
-volume /view:/view
```

4 Sample Docker Containers

4.1 Docker Container with Dynamic View Client Access

A VersionVault user would like to ssh into a Docker container, set into a VersionVault view (setview) and update some data files (checkin/checkout).

The container to be built shares the VersionVault binaries on its Docker host.

4.1.1 Build Directory

The build directory consists of the following files and directories:

- `Dockerfile`
    - Used to build the container
- `config_ssh.sh`
    - Script to be run by the Docker container
- `config_dir`
    - Directory containing VersionVault configuration files
        - `albd_rt_params.conf`
        - `admin.conf`
        - `cacert.pem`
        - `snapshot.conf`

    - albd.conf
    - cacert.pem.template
    - vob_scrubber_params
    - albd_lad.conf
- docker_compose.yml
    - docker compose YAML file

4.1.2 Dockerfile

This Dockerfile creates an image that has the following entities:

- Redhat 7.6 Init Base images
- NIS
- SSH
- root user
- Script to configure the VersionVault License Server, Registry Server, Region, start VersionVault and exec the Linux init process.

This Dockerfile uses environment variables to set up NIS, VersionVault Registry, Region, and License. These environment variables are:

- VVRGYHOST
    - Registry Server hostname
- VVRGYREGION
    - Network region defining namespace of VOB and view tags
- FNECLOUD
    - VersionVault License Type
- FNESERVER
    - VersionVault License URL
- FNESERVERID
    - VersionVault License ServerID (Null string if local license type)
- ENV FNECERT
    - VersionVault License Certificate (Null string if not needed)
- NISDOMAIN
    - NIS Domain
- NISSERVER
    - NIS Domain Server

The Dockerfile is as follows:

```
FROM registry.redhat.io/ubi7-init:7.6
ENV VVDIR /opt/hcl/ccm/versionvault
ENV VVADMDIR /var/adm/hcl/versionvault/
ENV VVRGYHOST registry_host
ENV VVRGYREGION  registry_region
ENV FNETYPE cloud
ENV FNESERVER "https://hclsoftware-uat.compliance.flexnetoperations.com"
ENV FNESERVERID "A1B2C3D4E5"
ENV FNECERT “”
ENV NISDOMAIN "mydomain.com"
ENV NISSERVER "mydomain.nis.server"
```

```
ADD config_dir/* $VVADMDIR/config/
RUN chmod 755 $VVADMDIR/config/

RUN (cd /lib/systemd/system/sysinit.target.wants/; for i in *; do [ $i == \
systemd-tmpfiles-setup.service ] || rm -f $i; done); \
rm -f /lib/systemd/system/multi-user.target.wants/*;\
rm -f /etc/systemd/system/*.wants/*;\
rm -f /lib/systemd/system/local-fs.target.wants/*; \
rm -f /lib/systemd/system/sockets.target.wants/*udev*; \
rm -f /lib/systemd/system/sockets.target.wants/*initctl*; \rm -f
/lib/systemd/system/basic.target.wants/*;\
rm -f /lib/systemd/system/anaconda.target.wants/*;

RUN yum clean all
RUN yum install -y openssh-server \
          openssh-clients \
          iputils \
          hostname \
          vim \
          gcc \
          make \
          strace \
          psmisc \
          file \
          unzip \
          gedit \
          dbus-x11 \
          libacl \
          xorg-x11-xauth \
          gtk2.i686 \
          libXtst.i686 \
          libSM.i686 \
          libICE.i686 \
          motif.i686 \
          ncurses-libs.i686 \
          tcsh \
          lsof \
          sudo \
          ypbind \
          bind-utils

RUN echo domain $NISDOMAIN server $NISSERVER >> /etc/yp.conf
RUN sed 's@passwd:\s*files\s*sss@passwd: files nis@g' -i /etc/nsswitch.conf
RUN sed 's@shadow:\s*files\s*sss@shadow: files nis@g' -i /etc/nsswitch.conf
RUN sed 's@group:\s*files\s*sss@group: files nis@g' -i /etc/nsswitch.conf
RUN systemctl enable ypbind

RUN mkdir -p /etc/ssh
RUN echo 'root:vvault' | chpasswd

RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' \
/etc/ssh/sshd_config

RUN sed 's@session\s*required\s*pam_loginuid.so@session optional  \
pam_loginuid.so@g' -i /etc/pam.d/sshd

RUN echo "PermitTunnel yes" >> /etc/ssh/sshd_config
RUN echo "StrictModes no" >> /etc/ssh/sshd_config
RUN usr/bin/ssh-keygen -A >> /tmp/gen
ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile
COPY config_ssh.sh /tmp
RUN chmod 777 /tmp/config_ssh.sh
EXPOSE 22
RUN systemctl enable sshd
CMD ["/tmp/config_ssh.sh"]
```

4.1.3 config_ssh.sh

This script configures the Registry Server, Registry region, License Information for VersionVault, and starts VersionVault. It then execs the Linux init process.

The config_ssh.sh file is as follows:

```
#!/bin/bash

LOGFILE=/tmp/startup_log

#
# This function configures the region and registry server
#
configure_vvault_info()
{
    echo ${VVRGYHOST} > ${VVADMDIR}/config/rgy_hosts.conf
    echo ${VVRGYREGION} > ${VVADMDIR}/config/rgy_region.conf
}
#
# This function configures FNE license
#
configure_fne_license()
{
    FNELICDIR=${VVADMDIR}/../common/config/
    FNELICFILE=${FNELICDIR}/fne.properties
    VVFNELICFILE=${VVADMDIR}/config/fne_config

    mkdir -p  ${FNELICDIR}
    chmod 755 ${FNELICDIR}
    echo "CCM_FNE_CloudOrLocal=${FNETYPE}" >> ${FNELICFILE}
    echo "CCM_FNE_LicenseServerURL=${FNESERVER}" >> ${FNELICFILE}
    if [ ${FNETYPE} == "local" ]
    then
        FNESERVERID="~"
    fi
    echo "CCM_FNE_LicenseServerID=${FNESERVERID}" >> ${FNELICFILE}
    chmod 755 ${FNELICFILE}
    echo  ${FNESERVER} > ${VVFNELICFILE}
    echo  ${FNESERVERID}>> ${VVFNELICFILE}
    chmod 755 ${VVFNELICFILE}
    if [ -f "${FNECERT}" ]
    then
        cat ${FNECERT} >> ${VVADMDIR}/config/cacert.pem
    fi
}
configure_vvault_info
```

```
configure_fne_license
${VVDIR}/etc/versionvault start >> /tmp/startup_log 2>&1
mkdir -p /run/systemd/system
exec /usr/sbin/init
```

4.1.4 docker-compose.yml

The following is a docker-compose.yml file used to build and deploy the container. Configuration values for DNS, Registry, Region, and License should be filled in appropriately.

```
version: '3'
services:
  vvault_test:
    build: .
    image: vvault_test:v1
    privileged: true
    cap_add:
      - SYS_ADMIN
    volumes:
      - /net:/net:shared
      - /vobs:/vobs:shared
      - /opt/hcl:/opt/hcl
      - /opt/jdk:/opt/jdk
      - /opt/rational:/opt/rational
      - /home:/home
      - /view:/view
      - /usr/atria:/usr/atria
      - /sys/fs/cgroup:/sys/fs/cgroup:ro
    tmpfs:
       - /run
    devices:
       -  "/view/.specdev:/dev/mvfs"
    dns:
       - XX.XX.XX.XX
    dns_search:
       - XXXXX
       - YYYYYY
    stop_signal: SIGRTMIN+3
    environment:
      - VVRGYREGION=my_region
      - VVRGYHOST=my_registry_host
      - FNETYPE=cloud
      - FNESERVER=https://hclsoftware-uat.compliance.flexnetoperations.com
      - FNESERVERID=my_server_id
```

Notice the shared volumes and devices.

4.2 Docker Container with View-Extended Access

A VersionVault user would like to ssh into a Docker container whose base image is not supported by VersionVault release running on the Docker host.

4.2.1 Build Directory

The build directory consists of the following files and directories:

- `Dockerfile`

    - Used to build the container
- `config_ssh.sh`
    - Script to be run by Docker container
- `docker_compose.yml`
    - Docker compose YAML file

4.2.2 Dockerfile

The Dockerfile creates an image that uses a Redhat 7.6 Init Base image with

- NIS
- SSH
- Script to create the systemd directory to start the Linux init process.

This Dockerfile uses environment variables to set up NIS.

- NISDOMAIN
    - NIS Domain
- NISSERVER
    - NIS Domain Server

The Dockerfile is as follows:

```
FROM registry.redhat.io/ubi7-init:7.6
ENV NISDOMAIN "mydomain.com"
ENV NISSERVER "mydomain.nis.server"
RUN (cd /lib/systemd/system/sysinit.target.wants/; for i in *; do [ $i == \
systemd-tmpfiles-setup.service ] || rm -f $i; done); \
rm -f /lib/systemd/system/multi-user.target.wants/*;\
rm -f /etc/systemd/system/*.wants/*;\
rm -f /lib/systemd/system/local-fs.target.wants/*; \
rm -f /lib/systemd/system/sockets.target.wants/*udev*; \
rm -f /lib/systemd/system/sockets.target.wants/*initctl*; \
rm -f /lib/systemd/system/basic.target.wants/*;\
rm -f /lib/systemd/system/anaconda.target.wants/*;
RUN yum clean all
RUN yum install -y openssh-server \
          openssh-clients \
          iputils \
          hostname \
          vim \
          gcc \
          make \
          strace \
          psmisc \
          file \
          unzip \
          gedit \
          dbus-x11 \
          libacl \
          xorg-x11-xauth \
          gtk2.i686 \
          libXtst.i686 \
          libSM.i686 \
          libICE.i686 \
          motif.i686 \
          ncurses-libs.i686 \
          tcsh \
          lsof \
          sudo \
          rpcbind \
          ypbind \
          bind-utils

RUN echo domain $NISDOMAIN server $NISSERVER >> /etc/yp.conf
RUN sed 's@passwd:\s*files\s*sss@passwd: files nis@g' -i /etc/nsswitch.conf
RUN sed 's@shadow:\s*files\s*sss@shadow: files nis@g' -i /etc/nsswitch.conf
RUN sed 's@group:\s*files\s*sss@group: files nis@g' -i /etc/nsswitch.conf
RUN systemctl enable ypbind

RUN mkdir -p /etc/ssh
RUN echo 'root:vvault' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' \
/etc/ssh/sshd_config

RUN sed 's@session\s*required\s*pam_loginuid.so@session optional \
pam_loginuid.so@g' -i /etc/pam.d/sshd

RUN echo "PermitTunnel yes" >> /etc/ssh/sshd_config
RUN echo "StrictModes no" >> /etc/ssh/sshd_config

RUN usr/bin/ssh-keygen -A >> /tmp/gen
ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile
EXPOSE 22
RUN systemctl enable sshd
COPY config_ssh.sh /tmp
RUN chmod 777 /tmp/config_ssh.sh
CMD ["/tmp/config_ssh.sh"]
```

4.2.3 config_ssh.sh

This script creates the directory required to start up and exec the Linux init process.

The config_ssh.sh file is as follows:

```
#!/bin/bash

mkdir -p /run/systemd/system
exec /usr/sbin/init
```

4.2.4 docker-compose.yml

The following is a docker-compose.yml file used to build and deploy the container.

Configuration values for DNS should be filled in appropriately.

```
version: '3'
services:
  vvault_vextended:
    build: .
    image: vvault_vextended:v1
    privileged: true
    volumes:
      - /net:/net:shared
      - /home:/home
      - /view:/view
      - /sys/fs/cgroup:/sys/fs/cgroup:ro
    tmpfs:
      - /run
    dns:
      - XX.XX.XX.XX
    dns_search:
      - XXXX
      - YYYY
   stop_signal: SIGRTMIN+3
```

Notice the shared volumes.

5 Reference and More

Docker Website – The main website for Docker

Docker Documentation – The website for Docker documentation

Systemd in a Container – Blog for running systemd in a Docker container

VersionVault Documentation – The website for VersionVault documentation.