Hewlett Packard Enterprise

Technical white paper

Oracle Database Docker Containers on HPE 3PAR Storage

Example configuration and settings for deploying Oracle Database Enterprise Edition Docker containers

Contents

Executive summary

A container image is a lightweight, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, and settings.[1] Unlike a virtual machine, a container does not require its own operating system. Such consistent packaging and low overhead allows development teams to move quickly through development lifecycles—deploying software at the speed of business.

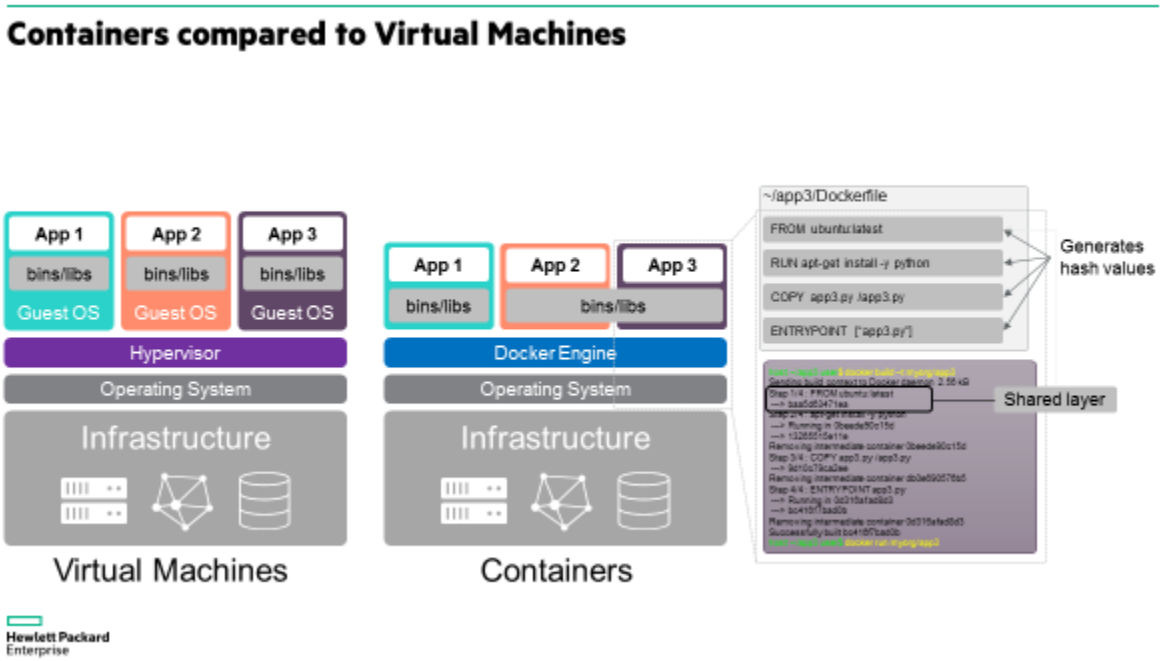


Figure 1. Virtual Machines versus Containers

Oracle has partnered with Docker, the world's leading containerization platform, to make a range of containerized Oracle products available through the Docker store. Deploying an Oracle Database Enterprise Edition container from the Docker store is quick and easy. Just download and run the Oracle Docker images.

After your Oracle database is containerized, how will you manage the data that goes with it? Because storing data in containers is not a best practice, you need enterprise-class persistent storage to match.

Whether you deploy Oracle containers on virtual machines or on bare metal, a unified, flash–optimized HPE 3PAR Storage provides the reliability, manageability, and flexibility this enterprise database requires. With the HPE 3PAR Volume Plug-in for Docker, enterprises can easily automate data management across the entire DevOps cycle.

[1] What is a container? By Docker

Solution overview

This paper shows you how to install and use Docker and how to run a containerized Oracle using persistent volumes provisioned with the HPE 3PAR Volume Plug-in for Docker. The focus is on demonstrating real-life use cases using Oracle Docker containers with HPE 3PAR storage.

Solution components

An HPE 3PAR 20800 storage array was the storage platform used for testing in this solution. Red Hat® Enterprise Linux® 7.4 running on an HPE ProLiant DL380 Generation10 (Gen10) server was used to host Docker Enterprise Edition with Oracle 12c1 and 12c2 database containers and the HPE 3PAR Volume Plug-in for Docker version 2.0.2 from the Docker store[2].

Hardware

Server

HPE ProLiant DL380 Gen10 server with 256 GB RAM.

- Intel® Xeon® Gold 5115 CPU @ 2.40 GHz, 10/10 cores; 20 threads
- 256 GB RAM

Figure 2. HPE ProLiant DL380 Gen10 server (front view)

Storage array

HPE 3PAR 20800 storage array (release version 3.3.1, EMU1 plus patch P21) with the following:

- Eight storage controller nodes
- SSDs: 32
  - 12 @ 1.92 TB, 100K
  - 20 @ 480 GB, 150K
- FC drives: 128
  - 64@300 GB, 15K rpm
  - 64@900 GB, 10K rpm
- CPGs employed:
  - SSD_r6_150K

[2] Note that the server for this lab setup is relatively small in relation to the storage array that was used. The HPE 3PAR 20800 described below could support a larger enterprise server or multiple servers.

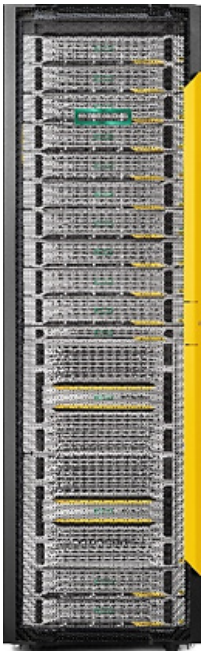


Figure 3. HPE 3PAR 20800 storage

Fibre Channel switch

Two HPE SN6000B 16 GB switches

- Kernel: 2.6.14.2
- Fabric OS: v7.4.1d

Fibre Channel Zoning

- Two FC ports/initiators on Oracle host
- Two FC ports/targets per HPE 3PAR storage node
- One initiator to multiple targets per zone
- Oracle host mirrored to storage node pairs

Refer to HPE 3PAR Storage: A reference and best practices guide for detailed recommendations on FC Host Zoning.

Figure 4. HPE SN6000B Fibre Channel switch

Software

Server OS

- Red Hat Enterprise Linux Server (release 7.4, kernel 3.10.0-693.el7.x86_64)

Application software

- Docker Enterprise Edition 17.06.2-ee-7
- Oracle 12.1.0.2 Enterprise Edition Docker Image
  - Oracle Database 12c1
- Oracle 12.2.0.1 Enterprise Edition Docker Image
  - Oracle Database 12c2
- Load generators (for Oracle database)
  - Swingbench 2.5.0.971 (OLTP)
  - Swingbench 2.6.0.1076 (DSS)
- Docker Plug-ins
  - HPE 3PAR Volume Plug-in for Docker, version 2.0.2

Figure 4 illustrates the installation that was used to perform testing for this solution. Docker Enterprise Edition was installed on the HPE ProLiant DL380 Gen10 server. Oracle Database Enterprise Edition 12.1.0.2 and 12.2.0.1 images from the Docker Store were pulled for the creation of Oracle database containers. Storage for the solution was provided by Docker volumes created by means of the HPE 3PAR Volume Plug-in for Docker, on flash devices from an HPE 3PAR 20800 storage array. OLTP and DSS workloads were run on the Oracle container, with a remote Oracle client serving as a load generator.

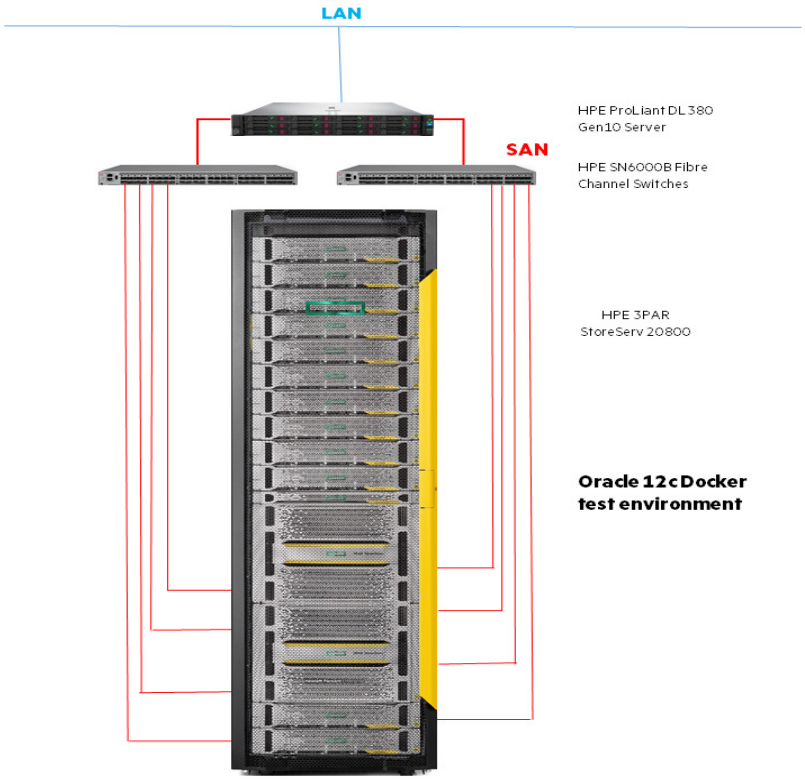


Figure 5. Test environment for Oracle 12c Docker Containers on HPE 3PAR Storage

Deploying Oracle Database Docker Containers on HPE 3PAR storage

Installing Docker on Linux

Docker is available in a free Community Edition and a licensed Enterprise Edition. For this project, HPE engineers used Docker Enterprise Edition installed on Red Hat Enterprise Linux 7.4. A free one-month evaluation license for Docker Enterprise Edition can be downloaded from the Docker Store and installed through the following process.

1. Select 'Start 1 Month Trial' on the Docker Enterprise Edition Trial page of the Docker Store:

   https://store.docker.com/editions/enterprise/docker-ee-trial

2. Create a Docker account, if not already a subscriber.
3. Log in and subscribe to the 'Docker Enterprise Edition Trial Start 1 Month Trial'.
4. On the Setup Instructions page, use the URL provided to download the Docker EE rpms, and follow the instructions to install the Docker Enterprise Edition container engine on Red Hat Linux.

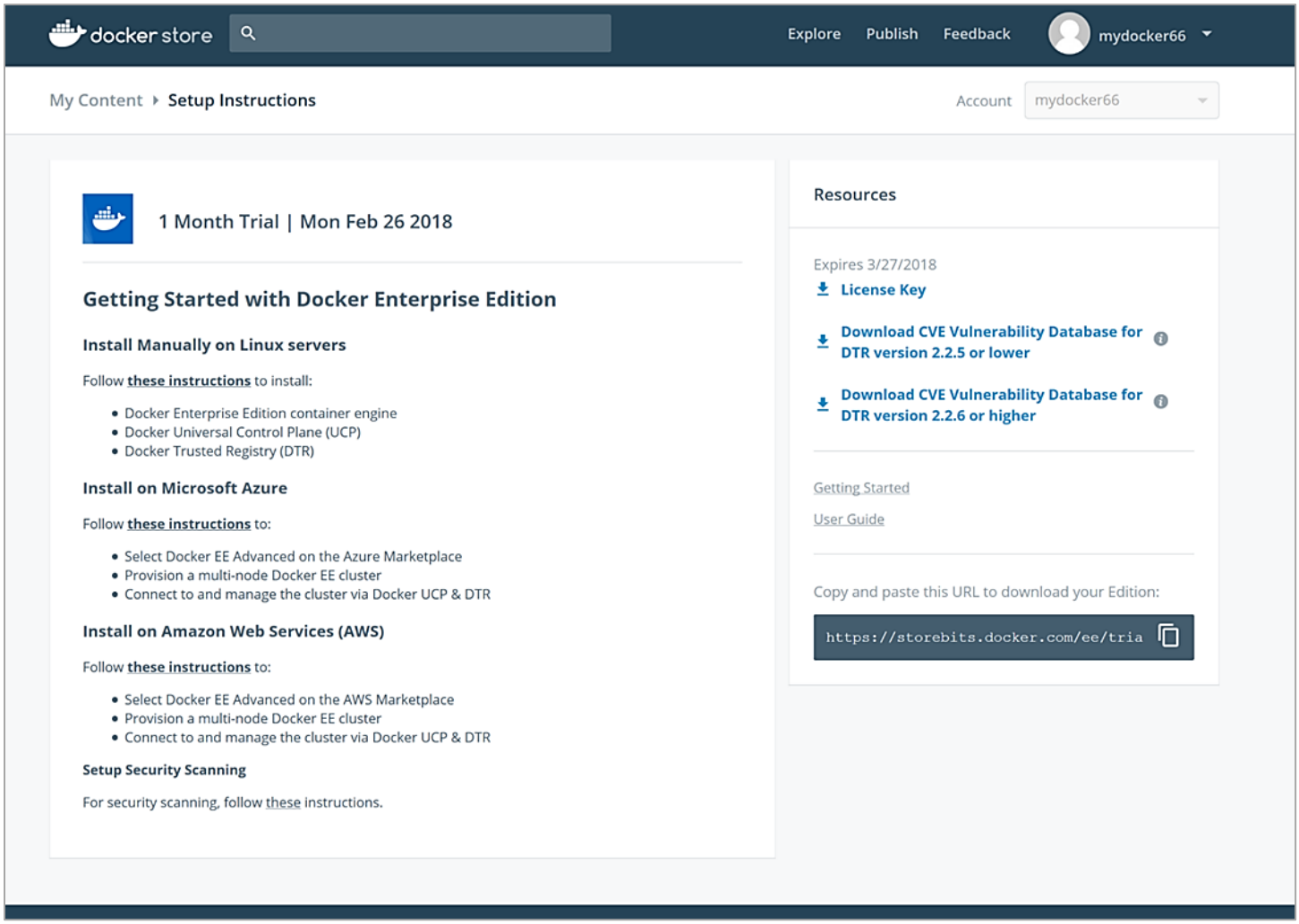


Figure 6. Docker Setup Instructions page

5. Navigate to the page corresponding to the appropriate Red Hat version. (For this project the Red Hat 7 branch provided the complete set of rpms required for installation of Docker Enterprise Edition, under `x86_64/stable-17.06`.)
6. Download the Docker Enterprise Edition rpm packages. For some installations, it might be possible to accomplish Docker installation with a single `yum` command. See Get Docker EE for Red Hat Enterprise Linux for further information.

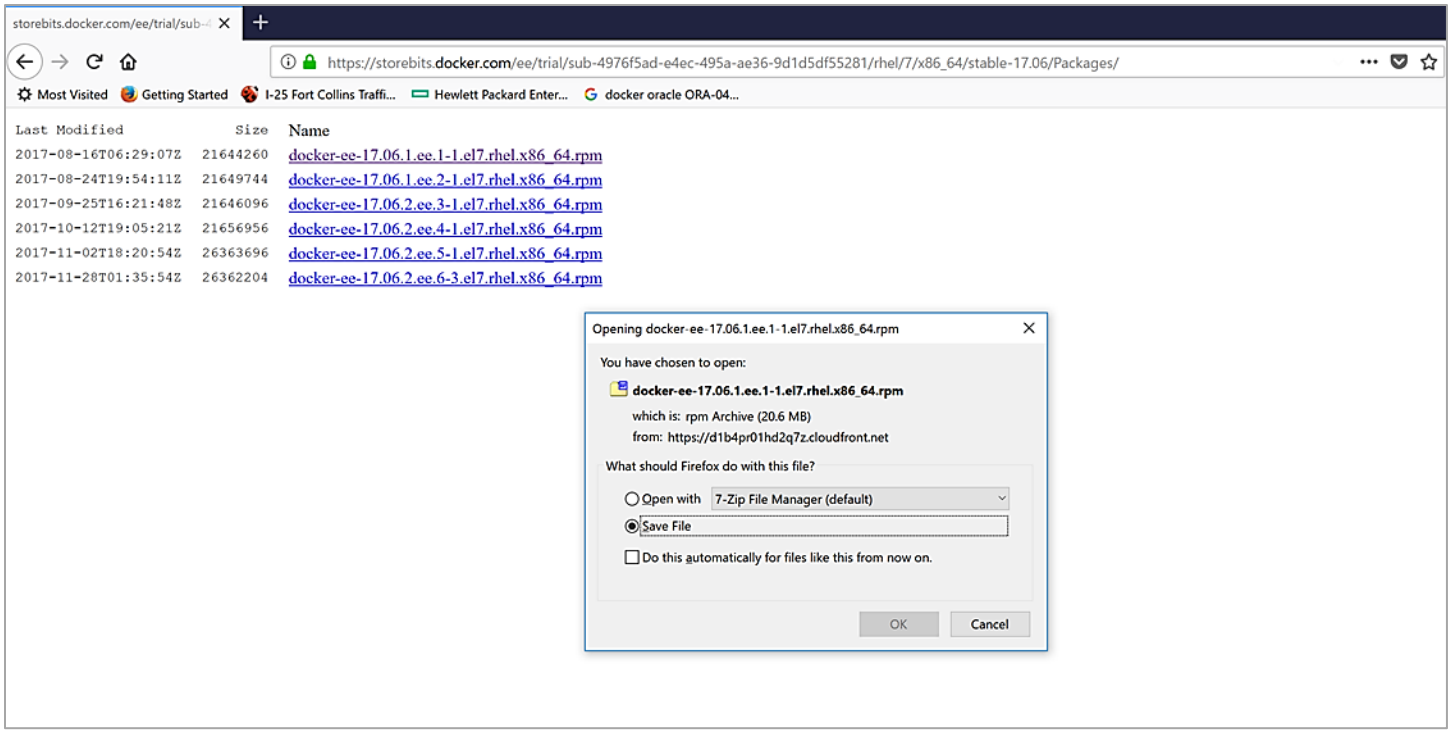


Figure 7. Docker EE download page for RHEL 7, x86-74, stable 17.06 release

7. Install the Docker packages. (Either log in as `root` for the installation or use `sudo` to run as `root`.)

   Note that the `container-selinux` package is a dependency that might have to be resolved before installing the Docker rpms. The `container-selinux` package can be obtained for installation from a public repository.

   For this project, `container-selinux-2.33-1.git86f33cd.el7.noarch` was downloaded from RPMFind.

```
[soldb06 Docker]# sudo yum install ./docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64.rpm
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
This system is not registered with an entitlement server. You can use subscription-manager to register.
Examining ./docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64.rpm: docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64
Marking ./docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package docker-ee.x86_64 0:17.06.1.ee.1-1.el7.rhel will be installed
--> Processing Dependency: container-selinux >= 2.9 for package: docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64
--> Finished Dependency Resolution
Error: Package: docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64 (/docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64)
           Requires: container-selinux >= 2.9
 You could try using --skip-broken to work around the problem
 You could try running: rpm -Va --nofiles --nodigest
[soldb06 Docker]#
```

Figure 8. The container-selinux package is required for installation of Docker EE

8. Install the Docker Enterprise Edition packages.

```
[soldb06 Docker]# sudo yum install ./docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64.rpm
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
This system is not registered with an entitlement server. You can use subscription-manager to register.
Examining ./docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64.rpm: docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64
Marking ./docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package docker-ee.x86_64 0:17.06.1.ee.1-1.el7.rhel will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================================================================
 Package                    Arch                    Version                                  Repository                                                  Size
================================================================================================================================================================
Installing:
 docker-ee                  x86_64                  17.06.1.ee.1-1.el7.rhel                  /docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64                   74 M

Transaction Summary
================================================================================================================================================================
Install  1 Package

Total size: 74 M
Installed size: 74 M
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64                                                                                                   1/1
  Verifying  : docker-ee-17.06.1.ee.1-1.el7.rhel.x86_64                                                                                                   1/1

Installed:
  docker-ee.x86_64 0:17.06.1.ee.1-1.el7.rhel

Complete!
[soldb06 Docker]#
```

Figure 9. Installation of the first docker-ee package

Note – using a proxy server with Docker

If using a proxy server, configure Docker to use the proxy. Depending on the Docker version, this can be done either on the Docker client or by using environment variables.[3]

When using `systemctl` to start Docker with an `http` or `https` proxy server, add the appropriate configuration files for the Docker `systemd` service, in the directory `/etc/systemd/system/docker.service.d` as shown in the figure below.[4]

```
[soldb06 docker.service.d]# pwd
/etc/systemd/system/docker.service.d
[soldb06 docker.service.d]# cat http-proxy.conf
[Service]
Environment="HTTP_PROXY=http://99.99.99.10:8080"
[soldb06 docker.service.d]# cat https-proxy
[Service]
Environment="HTTPS_PROXY=https://autocache.mycorp.net/"
[soldb06 docker.service.d]#
```

Figure 10. Docker service configuration files for HTTP/HTTPS

9. Enable and start Docker with `systemctl`.
   a. To start Docker in the current session:

      ```
      # systemctl start docker
      ```

[3] See Configure Docker to use a proxy server.
[4] See also Control Docker with systemd - HTTP/HTTPs proxy for details about proxy configuration with systemd for Docker.

b. To enable Docker to start automatically on boot, use `systemctl` enable

```
# systemctl enable docker
```

10. Using `systemctl`, verify Docker is running.

```
[soldb06 ~]# sudo systemctl start docker
[soldb06 ~]# sudo systemctl enable docker
[soldb06 ~]# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/docker.service.d
           └─http-proxy.conf
   Active: active (running) since Thu 2018-03-01 11:21:16 MST; 6 days ago
     Docs: https://docs.docker.com
 Main PID: 3681 (dockerd)
   CGroup: /system.slice/docker.service
           ├─ 3681 /usr/bin/dockerd
           ├─ 3695 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --metrics-interval=0 --start-timeout 2m --state-dir /var/run/docker/l...
           ├─20957 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 4001 -container-ip 172.17.0.2 -container-port 4001
           ├─20969 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 2380 -container-ip 172.17.0.2 -container-port 2380
           ├─20982 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 2379 -container-ip 172.17.0.2 -container-port 2379
           ├─20988 docker-containerd-shim d0584beb2862194479563fa83ed7df49a51f1fb2335154272754680e8b7f6048 /var/run/docker/libcontainerd/d0584beb2862194479563fa83ed7df49...
           ├─22397 docker-containerd-shim 294bdb2185b7267e13d65bc06be36b54b3d95a7d2e35487104d2c20fb2cd7fcb /var/run/docker/libcontainerd/294bdb2185b7267e13d65bc06be36b54...
           └─294bdb2185b7267e13d65bc06be36b54b3d95a7d2e35487104d2c20fb2cd7fcb
             ├─22414 /bin/sh -c ./plugin-start
             ├─22425 /bin/sh ./plugin-start
             ├─22429 /sbin/udevd
             └─22433 /usr/bin/python /usr/bin/twistd --nodaemon hpe_plugin_service

Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="/usr/lib/python2.7/site-packages/urllib3/connectionpool.py:858: InsecureRequestWa...
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="  InsecureRequestWarning)" plugin=294bdb2185b7267e13d65bc06be36b54b3d9...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02 22:55:14.161 14 INFO hpedockerplugin.etcdutil [-] ETCDUTIL ...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02 22:55:14.162 14 INFO hpedockerplugin.etcdutil [-] ETCDUTIL ...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02 22:55:14.162 14 INFO hpedockerplugin.etcdutil [-] ETCDUTIL ...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02 22:55:14.164 14 INFO hpedockerplugin.hpe_storage_api [-] Ov...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02T22:55:14+0000 [twisted.scripts._twistd_unix.UnixAppLogger#i...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02T22:55:14+0000 [twisted.scripts._twistd_unix.UnixAppLogger#i...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02T22:55:14+0000 [-] Site starting on '/run/docker/plugins/hpe...20fb2cd7fcb
Mar 02 15:55:14 soldb06 dockerd[3681]: time="2018-03-02T15:55:14-07:00" level=info msg="2018-03-02T22:55:14+0000 [twisted.web.server.Site#info] Starting facto...20fb2cd7fcb
Hint: Some lines were ellipsized, use -l to show in full.
[soldb06 ~]#
```

Figure 11. Using `systemctl` to verify Docker is enabled and active

11. Log in to Docker with the Docker account.

```
# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over
to https://hub.docker.com to create one.
Username: mydocker66
Password:
Login Succeeded
#
```

12. Verify Docker functionality by running 'hello-world'. If no local image for "hello-world" is found, the image is pulled from the Docker hub, as shown in the figure below.

```
[soldb06 ~]# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:083de497cff944f969d8499ab94f07134c50bcf5e6b9559b27182d3fa80ce3f7
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

[soldb06 ~]#
```

Figure 12. Running Docker 'hello-world'

13. By default, the `docker` daemon binds to a UNIX® port owned by the `root` user. To avoid the necessity of logging in as a superuser or using 'sudo' to run Docker as a non-root user, optionally create a Docker user belonging to the `docker` group.

```
[soldb06 ~]# whoami
root
[soldb06 ~]# useradd dockuser
[soldb06 ~]# su - dockuser
[soldb06 ~]$ whoami
dockuser
[soldb06 ~]$ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:
Post http://%2Fvar%2Frun%2Fdocker.socknix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
[soldb06 ~]$ exit
[soldb06 ~]# whoami
root
[soldb06 ~]# usermod -aG docker dockuser
[soldb06 ~]# su - dockuser
[soldb06 ~]$ whoami
dockuser
[soldb06 ~]$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

Figure 13. Create a Docker user and add the user to the docker group to run Docker without `sudo`

Deploying the Fibre Channel HPE 3PAR Volume Plug-in for Docker

The HPE 3PAR Volume Plug-in for Docker is available from the Docker Store, as well as from GitHub. The plug-in provides the ability to create Docker storage volumes on HPE 3PAR storage via Fibre Channel (FC) or iSCSI. For this project, the FC plug-in was employed for Docker volume management.

Install the etcd container

The `etcd` key-value store container is required for the FC plug-in. Install `etcd` as follows:

1. Export the IP address of the Docker host.

```
# export HostIP=10.120.19.116
```

2. Run the following command to setup the etcd container:

```
# docker run -d -v /usr/share/ca-certificates/:/etc/ssl/certs -p 4001:4001 \
 -p 2380:2380 -p 2379:2379 \
 --name etcd quay.io/coreos/etcd:v2.2.0 \
 -name etcd0 \
 -advertise-client-urls http://${HostIP}:2379,http://${HostIP}:4001 \
 -listen-client-urls http://0.0.0.0:2379,http://0.0.0.0:4001 \
 -initial-advertise-peer-urls http://${HostIP}:2380 \
 -listen-peer-urls http://0.0.0.0:2380 \
 -initial-cluster-token etcd-cluster-1 \
 -initial-cluster etcd0=http://${HostIP}:2380 \
 -initial-cluster-state new
```

```
[soldb06 docker]# export HostIP=10.120.19.116
[soldb06 docker]# docker run -d -v /usr/share/ca-certificates/:/etc/ssl/certs -p 4001:4001 \
> -p 2380:2380 -p 2379:2379 \
> --name etcd quay.io/coreos/etcd:v2.2.0 \
> -name etcd0 \
> -advertise-client-urls http://${HostIP}:2379,http://${HostIP}:4001 \
> -listen-client-urls http://0.0.0.0:2379,http://0.0.0.0:4001 \
> -initial-advertise-peer-urls http://${HostIP}:2380 \
> -listen-peer-urls http://0.0.0.0:2380 \
> -initial-cluster-token etcd-cluster-1 \
> -initial-cluster etcd0=http://${HostIP}:2380 \
> -initial-cluster-state new
Unable to find image 'quay.io/coreos/etcd:v2.2.0' locally
v2.2.0: Pulling from coreos/etcd
ebf0080d8f12: Pull complete
ed06b3b63e9d: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:608bb546c4fea694e58ca8414890e662557e36bdce4c1413cf9ad468ef3bf711
Status: Downloaded newer image for quay.io/coreos/etcd:v2.2.0
ac4e457de4dda4398c77680b0bf91a78a16ddcbdd26a5e2e33c7fc4d66911952
[soldb06 docker]#

[soldb06 docker]# docker images
REPOSITORY            TAG                 IMAGE ID            CREATED             SIZE
quay.io/coreos/etcd   v2.2.0              ee946ee864ee        2 years ago         27.4MB
[soldb06 docker]#
```

Figure 14. Installation of etcd container

```
[soldb06 docker]# docker ps
CONTAINER ID        IMAGE                         COMMAND                  CREATED             STATUS              PORTS
                                                    NAMES
13af6bed2604        quay.io/coreos/etcd:v2.2.0    "/etcd -name etcd0..."   4 weeks ago         Up 4 seconds        0.0.0.0:2379-2380
->2379-2380/tcp, 0.0.0.0:4001->4001/tcp, 7001/tcp   etcd
[soldb06 docker]#
```

Figure 15. `docker ps` shows `etcd` is running

Create hpe.conf file for the HPE 3PAR Volume Plug-in for Docker

Make a directory for the FC plug-in `/etc/hpedockerplugin` and create the `hpe.conf` file in the directory. A sample template for the FC plug-in `hpe.conf` file can be found at https://github.com/hpe-storage/python-hpedockerplugin/blob/master/config/hpe.conf.sample.3parFC. Modify the conf file values as needed for the host and the HPE 3PAR storage.[5] For example, the following values were used in testing for this project:

```
ssh_hosts_key_file = /root/.ssh/known_hosts

host_etcd_ip_address = 10.120.19.116

host_etcd_port_number = 2379

hpe3par_api_url = https://10.120.14.220:8080/api/v1

hpe3par_username = 3paradm

hpe3par_password = 3pardata

san_ip = 10.120.14.220

san_login = 3paradm
```

[5] Note that this file should be made secure, because it contains login credentials and other sensitive information in plain text.

```
san_password = 3pardata
hpe3par_cpg = SSD_r6_150K
```

Configure multipath.conf

Create or modify `/etc/multipath.conf`. The following default values were used for this project:

```
# cat /etc/multipath.conf
defaults {
                polling_interval 10
                max_fds 8192
        }
        devices {
                device {
                        vendor                  "3PARdata"
                        product                 "VV"
                        no_path_retry           18
                        features                "0"
                        hardware_handler        "0"
                        path_grouping_policy    multibus
                        #getuid_callout         "/lib/udev/scsi_id --whitelisted --device=/dev/%n"
                        path_selector           "round-robin 0"
                        rr_weight               uniform
                        rr_min_io_rq            1
                        path_checker            tur
                        failback                immediate
                }
        }
#
```

See Multipath Support – HPE 3PAR Volume Plug-in for Docker for further information on multipathing setup.

Prerequisite packages

Use the `rpm –q` command to make sure the prerequisite device-mapper-multipath and iscsi-initiator-utils packages are installed.

```
# rpm -q iscsi-initiator-utils
iscsi-initiator-utils-6.2.0.874-4.el7.x86_64
# rpm -q device-mapper-multipath
device-mapper-multipath-0.4.9-111.el7.x86_64
#
```

If these are not found, install the `iscsi-initiator-utils` and `device-mapper-multipath` packages. Then configure `/etc/multipath.conf` and run the following commands:

```
systemctl daemon-reload
systemctl enable iscsid multipathd
systemctl start iscsid multipathd
```

See Steps for Deploying the Managed Plugin for additional information on prerequisite packages for the HPE 3PAR Volume Plug-in for Docker.

Install the FC plug-in

Run the following command to install the FC plug-in:

```
# docker plugin install store/hpestorage/hpedockervolumeplugin:<version> --disable --alias hpe
```

```
[soldb06 ~]# docker plugin install store/hpestorage/hpedockervolumeplugin:2.0.2 --disable --alias hpe
Plugin "store/hpestorage/hpedockervolumeplugin:2.0.2" is requesting the following privileges:
 - network: [host]
 - mount: [/dev]
 - mount: [/run/lock]
 - mount: [/var/lib]
 - mount: [/etc]
 - mount: [/var/run/docker.sock]
 - mount: [/root/.ssh]
 - mount: [/sys]
 - mount: [/root/plugin/certs]
 - mount: [/sbin/iscsiadm]
 - mount: [/lib/modules]
 - mount: [/lib/x86_64-linux-gnu]
 - allow-all-devices: [true]
 - capabilities: [CAP_SYS_ADMIN CAP_SYS_RESOURCE CAP_MKNOD CAP_SYS_MODULE]
Do you grant the above permissions? [y/N] y
2.0.2: Pulling from store/hpestorage/hpedockervolumeplugin
63c35b4da7ed: Download complete
Digest: sha256:6a4653fbc373e26d5dbe5379fb3a6ccf91c2f7af90e04e5436de28000f7aa972
Status: Downloaded newer image for store/hpestorage/hpedockervolumeplugin:2.0.2
Installed plugin store/hpestorage/hpedockervolumeplugin:2.0.2
[soldb06 ~]#

[soldb06 ~]# docker plugin ls
ID                  NAME                                      DESCRIPTION                   ENABLED
7540c11c061b        hpe:latest                                HPE Docker Volume Plugin      false
cf158e4098be        docker/telemetry:1.0.0.linux-x86_64-stable   Docker Inc. metrics exporter   false
[soldb06 ~]#
```

Figure 16. Installation of HPE 3PAR Volume Plug-in for Docker version 2.0.2

Configure client certificates for secure etcd support

Execute the following command to point to the correct folder for secure `etcd` certificates. Example from testing:

```
# docker plugin set hpe glibc_libs.source=/lib64 certs.source=/etc/ssl/certs
```

Note that in this example, `/etc/ssl/certs` is the same certificate directory that was used to run the `etcd` container. (See Install the etcd container above.)

Make sure that the HPE 3PAR array is a known ssh host on the Docker server

The HPE 3PAR array must have an entry in the `.ssh/known_hosts` file for the Docker user on the Docker server. If the Docker user has not previously logged in to the HPE 3PAR array, a first login will create the required entry. The `ssh_host_key_file` entry in the `hpe.conf` file points to the `known_hosts` file for the Docker user. See the Create hpe.conf file for the HPE 3PAR Volume Plug-in for Docker section above

Enable the FC plug-in

The FC plug-in should now be enabled with “docker plugin enable”.

```
# docker plugin enable hpe
```

Confirm the FC plug-in is enabled

The `docker plugin ls` command confirms the FC plug-in is installed and enabled.

```
[soldb06 etc]# docker plugin ls
ID                  NAME                                      DESCRIPTION                   ENABLED
7540c11c061b        hpe:latest                                HPE Docker Volume Plugin      true
cf158e4098be        docker/telemetry:1.0.0.linux-x86_64-stable   Docker Inc. metrics exporter   false
[soldb06 etc]#
```

Figure 17. The `docker plugin ls` command shows the HPE 3PAR Volume Plug-in for Docker is enabled

Creating Docker volumes with the HPE 3PAR Volume Plug-in for Docker

The "docker volume" commands are used to create, list, and remove Docker volumes. The `docker volume –help` command provides usage information.

```
[soldb06 etc]# docker volume --help

Usage:  docker volume COMMAND

Manage volumes

Options:
      --help   Print usage

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
[soldb06 etc]#
```

Figure 18. Usage for "docker volume" commands

```
[soldb06 etc]# docker volume create --help

Usage:  docker volume create [OPTIONS] [VOLUME]

Create a volume

Options:
  -d, --driver string   Specify volume driver name (default "local")
      --help            Print usage
      --label list      Set metadata for a volume
  -o, --opt map         Set driver specific options (default map[])
[soldb06 etc]#
```

Figure 19. Usage for `docker volume create` command

To create a Docker volume on the HPE 3PAR storage, pass the plug-in name "hpe" as the driver name, and for driver-specific options provide the number of gigabytes to be allocated.

```
[soldb06 etc]# docker volume create -d hpe --name test_vol1 -o size=10
test_vol1
[soldb06 etc]# docker volume ls
DRIVER              VOLUME NAME
hpe:latest          test_vol1
[soldb06 etc]#
```

Figure 20. Creating a 10 GB Docker volume on HPE 3PAR storage

On the HPE 3PAR, the Docker volume will have a name prefixed by `dcv-` (Docker container volume). Volume information can be viewed through the HPE 3PAR SSMC or the HPE 3PAR CLI.

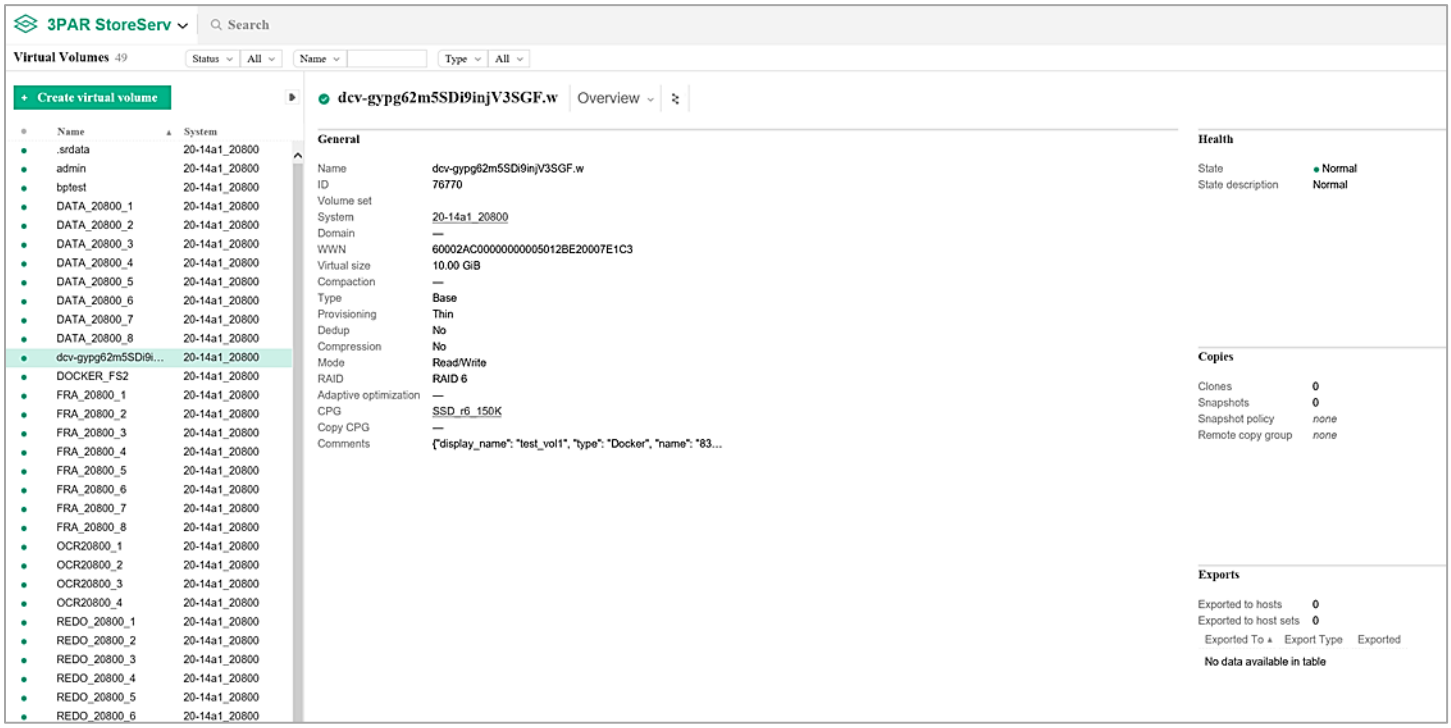


Figure 21. View of the Docker volume on HPE 3PAR SSMC

```
20-14a1_20800 cli% showvv -s dcv*
                                                  ---------Snp---------- ---------Usr---------- --------Total---------
                                                  --(MiB)-- -(% VSize)-- --(MiB)-- -(% VSize)-- --------(MiB)--------- ---Efficiency---
  Id Name                          Prov Compr Dedup Type Rsvd Used Used Wrn Lim Rsvd Used Used Wrn Lim Rsvd Used HostWr VSize Compact Compress
76770 dcv-gypg62m5SDi9injV3SGF.w tpvv No    No    base    0    0  0.0  --  -- 1024    0  0.0   0   0 1024    0      0 10240    0.00       --
------------------------------------------------------------------------------------------------------------------------------------------
    1 total                                           0    0             1024    0             1024    0      0 10240
20-14a1_20800 cli%
```

Figure 22. View of the Docker volume on HPE 3PAR CLI

Configuring and running an Oracle database container from the Docker store with Docker 3PAR volumes

Oracle provides ready-made containerized images of Oracle Database Enterprise Edition 12.1.0.2 and 12.2.0.1, which can be downloaded from the Docker Store. By default, the images provide a multitenant container database (CDB) with a single pluggable database (PDB). The 12.1.0.2 image was used for this testing. Note that the 12.2.0.1 version of the Oracle Docker image only supports the use of a single data volume, so the

use of multiple volumes illustrated below will not work with an Oracle 12.2.0.1 container. However, the single volume used by the 12.2.0.1 container can be reused by a different container, as described in Reusing an existing database in an Oracle Database Docker Container.

Download and run the Oracle database image

When logged in to Docker, the Oracle database image can be downloaded and run with the following command:

```
# docker run –d -it --name <container> store/oracle/database-enterprise:<version>
```

This will produce a running Docker container with the specified container name, having an Oracle container database with a single pluggable database. The *cdb* and the *pdb* will have default names and a default password for the `sys` user. The database is instantiated on the container's file system. Or, the Oracle image can be pulled from the Docker Store, and containers can be run as needed from the local copy.

```
[soldb06 ~]# docker  pull store/oracle/database-enterprise:12.1.0.2
12.1.0.2: Pulling from store/oracle/database-enterprise
e1ef1eb9fcad: Pull complete
7b36741b57dd: Pull complete
183fb4717901: Pull complete
19e8aae86959: Pull complete
5264789218d2: Pull complete
Digest: sha256:4a6ce8480d5e0aa0dd798564a5340fbfc45033a26468745482457d92008195a1
Status: Downloaded newer image for store/oracle/database-enterprise:12.1.0.2
[soldb06 ~]#
```

Figure 23. Pulling the Oracle database 12.1.0.2 image from the Docker Store

```
[soldb06 ~]# docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
hello-world                         latest              f2a91732366c        4 months ago        1.85kB
store/oracle/database-enterprise    12.1.0.2            9410a79c6531        10 months ago       5.27GB
quay.io/coreos/etcd                 v2.2.0              ee946ee864ee        2 years ago         27.4MB
[soldb06 ~]#
```

Figure 24. The `docker images` command is used to verify the image now has a local copy

Customizing the database

To customize the database, the following parameters can be provided in a configuration file:

```
DB_PASSWD – Password for the sys user.  (Default: Oradoc_db1)

DB_MEMORY – Server memory to be allocated for database SGA and PGA.  (Default: 2GB)

DB_DOMAIN – Domain for the database server. (Default: localdomain)

DB_SID –  The Oracle SID of the container database (Default: ORCLCDB)

DB_PDB – The name of the pluggable database.  (Default: ORCLPDB1)
```

```
[soldb06 ~]# cat /etc/ora.conf
DB_PASSWD=%Oradata1
DB_MEMORY=8g
DB_DOMAIN=us.rdlabs.hpecorp.net
DB_SID=MYCDB1
DB_PDB=MYPDB1
[soldb06 ~]#
```

Figure 25. Example of Oracle database configuration file

Using HPE 3PAR as external storage for the database

Additionally, the Oracle container can be instantiated on external storage by passing in Docker volumes with appropriate metadata labels for Oracle binaries (bits), Oracle data (data), Oracle fast recovery (fra), and Oracle redo (redo). For this container, four Docker volumes were created with the HPE 3PAR Volume Plug-in for Docker:

Table 1. Docker volumes created with the HPE 3PAR Volume Plug-in for Docker

Contents	Volume Name	Size in GB	Label
Oracle product	OracleProd	50	bits
Oracle data	OracleData	400	data
Oracle fast recovery area	OracleFRA	800	fra
Oracle Redo	OracleRedo	100	redo

```
[soldb06 ~]# docker volume create -d hpe -o size=50 --name=OracleProd --label=bits
OracleProd
[soldb06 ~]# docker volume create -d hpe -o size=400 --name=OracleData --label=data
OracleData
[soldb06 ~]# docker volume create -d hpe -o size=800 --name=OracleFRA --label=fra
OracleFRA
[soldb06 ~]# docker volume create -d hpe -o size=100 --name=OracleRedo --label=redo
OracleRedo
[soldb06 ~]#
```

Figure 26. Creating the Docker volumes for Oracle

Use the `docker volume ls` command to list all Docker volumes.

```
[soldb06 ~]# docker volume ls
DRIVER              VOLUME NAME
hpe:latest          OracleData
hpe:latest          OracleFRA
hpe:latest          OracleProd
hpe:latest          OracleRedo
[soldb06 ~]#
```

Figure 27. Listing the Docker volumes

The `showvv` command in the HPE 3PAR CLI lists the Docker `dcv-` volumes, and with the `-showcols Name,Comment` parameters, will list the Docker volume names.

```
20-14a1_20800 cli% showvv dcv*
                                                                                -Rsvd(MiB)- -(MiB)-
  Id Name                      Prov Compr Dedup Type CopyOf  BsId Rd -Detailed_State-  Snp    Usr   VSize
76787 dcv-9RlokkU0T6S7WQ5IiYUhhw tpvv No    No    base ---    76787 RW normal              0   1024  102400
76785 dcv-EDnZkloVRx6m2OOmz2Khzw tpvv No    No    base ---    76785 RW normal              0   1024  409600
76786 dcv-LiFa6WziRninvTkqaJzM.w tpvv No    No    base ---    76786 RW normal              0   1024  819200
76784 dcv-P3KOiUXbTDSwb02RloIOFQ tpvv No    No    base ---    76784 RW normal              0   1024   51200
------------------------------------------------------------------------------------------------------------
   4 total                                                                                  0   4096 1382400
20-14a1_20800 cli% showvv -showcols Name,Comment dcv*
Name                    Comment
dcv-9RlokkU0T6S7WQ5IiYUhhw {"display_name": "OracleRedo", "type": "Docker", "name": "f5196892-4534-4fa4-bb59-0e4889852187", "volume_id": "f5196892-4534-4fa4-bb59-0e4889852187"}
dcv-EDnZkloVRx6m2OOmz2Khzw {"display_name": "OracleData", "type": "Docker", "name": "1039d992-5a15-471e-a6d8-e3a6cf62a1cf", "volume_id": "1039d992-5a15-471e-a6d8-e3a6cf62a1cf"}
dcv-LiFa6WziRninvTkqaJzM.w {"display_name": "OracleFRA", "type": "Docker", "name": "2e215ae9-6ce2-4678-a7bd-392a689cccfb", "volume_id": "2e215ae9-6ce2-4678-a7bd-392a689cccfb"}
dcv-P3KOiUXbTDSwb02RloIOFQ {"display_name": "OracleProd", "type": "Docker", "name": "3f728e89-45db-4c34-b06f-4d9196820e15", "volume_id": "3f728e89-45db-4c34-b06f-4d9196820e15"}
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
total
20-14a1_20800 cli%
```

Figure 28. Listing the Docker volumes with the HPE 3PAR CLI

Running an Oracle database in a container

Some common parameters used with the `docker run` command:

`-d`	Detach. Run the container in the background
`--env-file`	Read in a file of environment variables for the container
`-p`	Publish container's ports to the host
`--name`	Name of the container
`-v`	Bind mount a volume
`--shm-size`	Size of /dev/shm in bytes

So, putting it all together, a database instance can be run with the configuration file and the four Docker 3PAR volumes, using these parameters:

```
docker run -d --env-file /etc/ora.conf \
                      -p 1521:1521 -p 5500:5500 \
                      -it --name testdb \
                      -v OracleProd : /u01 \
                      -v OracleData : /u02 \
                      -v OracleFRA : /u03 \
                      -v OracleRedo : /u04 \
                      --shm-size=16g \
                      store/oracle/database-enterprise:12.1.0.2
```

```
[soldb06 ~]# docker run -d --env-file /etc/ora.conf -p 1521:1521 -p 5500:5500 -it --name testdb  -v OracleProd:/u01 -v OracleData:/u02 -v OracleFRA:/u03 -v Or
acleRedo:/u04 --shm-size=8g store/oracle/database-enterprise:12.1.0.2
fb38ef0baf022506c61a9f550aafd43b44f0914bf589491b8564351702a0d806
[soldb06 ~]#
```

Figure 29. Running the Oracle container with the configuration file and persistent storage mounts

Creation of the database container can take a few minutes. To monitor the progress of the container, the `docker logs <container>` command can be used. After the database container is up and running, a log message will show "The database is ready for use."

```
[soldb06 ~]# docker logs testdb
User check : root.
Setup Oracle Database
Oracle Database 12.1.0.2 Setup
Thu Mar 29 22:23:55 UTC 2018

Check parameters ......
log file is : /home/oracle/setup/log/paramChk.log
paramChk.sh is done at 0 sec

untar DB bits ......
log file is : /home/oracle/setup/log/untarDB.log
untarDB.sh is done at 91 sec

config DB ......
log file is : /home/oracle/setup/log/configDB.log
configDB.sh is done at 261 sec

Done ! The database is ready for use .
Thu Mar 29 22:23:55 UTC 2018
User check : root.
Setup Oracle Database

[soldb06 ~]#
```

Figure 30. Running Docker logs on the container to verify database is ready to use

```
[soldb06 ~]# docker ps --format "table {{.Names}}\t{{.ID}}\t{{.Image}}\t{{.Status}}" --filter "name=testdb"
NAMES               CONTAINER ID        IMAGE                                      STATUS
testdb              fb38ef0baf02        store/oracle/database-enterprise:12.1.0.2   Up 20 hours
[soldb06 ~]#
```

Figure 31. `docker ps` with a format and filter confirms the testdb1 container is running and healthy

Running a Bash session in the Oracle container

Run the following command to enter a Bash shell session as the `oracle` user in the Oracle container, giving the `oracle` user and group and the name of the container:

```
# docker exec --user oracle:oinstall -it <container> bin/bash
```

The figure below shows the opening of a Bash session. SQLPlus is used to confirm that the database instance name agrees with the ORACLE_SID specified in the configuration file. It can also be verified, using “df –h”, that the mounted Docker 3PAR volumes have been mounted at /u01, /u02, /u03, /u04, and used for Oracle product, data, fast recovery area, and redo, as intended.

```
[soldb06 ~]# docker exec --user oracle:oinstall -it testdb bin/bash
[oracle@fb38ef0baf02 /]$ df -h
Filesystem                                      Size  Used Avail Use% Mounted on
overlay                                          50G   10G   41G  20% /
tmpfs                                           126G     0  126G   0% /dev
tmpfs                                           126G     0  126G   0% /sys/fs/cgroup
/dev/mapper/360002ac00000000005012c050007e1c3    50G   11G   37G  23% /u01
/dev/mapper/360002ac00000000005012c060007e1c3   394G  4.0G  370G   2% /u02
/dev/mapper/360002ac00000000005012c070007e1c3   788G  121M  748G   1% /u03
/dev/mapper/360002ac00000000005012c080007e1c3    99G  3.1G   91G   4% /u04
/dev/mapper/rhel-root                            50G   10G   41G  20% /etc/hosts
shm                                             8.0G  4.0K  8.0G   1% /dev/shm
tmpfs                                           126G     0  126G   0% /sys/firmware
tmpfs                                           126G     0  126G   0% /proc/scsi
[oracle@fb38ef0baf02 /]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Fri Mar 30 18:26:13 2018

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced Analytics
and Real Application Testing options

SQL> select instance_name from v$instance;

INSTANCE_NAME
----------------
MYCDB1

SQL>
```

Figure 32. Running a Bash shell in the Oracle container to verify the Docker volume mounts and container database instance name

The `sysdba` user can perform SQL operations within the container using SQL*Plus.

```
SQL> CREATE TABLE NewOrders (
  2  CustNumber NUMBER(3),
  3  OrderDate  DATE,
  4  Street     VARCHAR2(20),
  5  City       VARCHAR2(20),
  6  State      CHAR(2),
  7  Zip        VARCHAR2(10)
  8  );

Table created.

SQL> INSERT INTO NewOrders (CustNumber, OrderDate, Street, City, State, Zip)
  2    VALUES (416, TO_DATE('22/November/2018 8:30:00AM','DD/MON/YY HH:MI:SSAM'),
  3      'One Big Loop', 'Duckburg', 'WA', '98989-9876');

1 row created.

SQL> Select * from NewOrders;

CUSTNUMBER ORDERDATE STREET              CITY                ST ZIP
---------- --------- -------------------- -------------------- -- ----------
       416 22-NOV-18 One Big Loop         Duckburg            WA 98989-9876

SQL>
```

Figure 33. Simple SQL operations on the containerized database

Using the Docker `cp` command to copy files to and from an Oracle Container

The Docker `cp` command is used to copy files into and out of a Docker container. For example, a script containing SQL statements could be copied into the Oracle database container for use by the `sysdba`. Here is a simple query to confirm temporary and default tablespaces:

```
# more checktemptablespace.sql
COLUMN property_name FORMAT A30
COLUMN property_value FORMAT A30
COLUMN description FORMAT A50
SET LINESIZE 200

SELECT *
FROM database_properties
WHERE property_name like '%TABLESPACE';
#
```

The Docker `cp` command copies the SQL script into the Oracle database container, and with `docker exec`, a Bash shell is opened for the `oracle` user.

```
# docker cp checktemptablespace.sql testdb:/home/oracle/
# docker exec --user oracle:oinstall -it testdb bin/bash
[oracle@3d14669e5b0a /]$
```

In SQL*Plus the `oracle` user executes the command with spooled output:

```
SQL> spool "/home/oracle/checkdefaults";
SQL> @checktemptablespace.sql

PROPERTY_NAME                  PROPERTY_VALUE                 DESCRIPTION
------------------------------ ------------------------------ ------------------------------------
---------------
DEFAULT_TEMP_TABLESPACE        TEMP2                          Name of default temporary
tablespace
DEFAULT_PERMANENT_TABLESPACE   USERS                          Name of default permanent
tablespace
SQL> exit
```

Outside of the Oracle database container, the Docker `cp` command can be used to make a local copy of the spooled output.

```
# docker cp testdb:/home/oracle/checkdefaults.lst .
# cat checkdefaults.lst

SQL> @checktemptablespace.sql

PROPERTY_NAME                  PROPERTY_VALUE                 DESCRIPTION
------------------------------ ------------------------------ ------------------------------------------------
---
DEFAULT_TEMP_TABLESPACE        TEMP2                          Name of default temporary tablespace
DEFAULT_PERMANENT_TABLESPACE   USERS                          Name of default permanent tablespace

SQL> exit
#
```

Connecting to the containerized database from a remote Oracle host

The pluggable database within the Docker container can be accessed over a network with a mapped port designated when the container is run. For example, to connect as `sys` to the container created in the previous section, use the mapped port with the password, pdb name, and domain specified in the env–file:

```
[oracle@soldb01]$ sqlplus sys/%Oradata1@10.120.19.116:1521/mypdb1.us.rdlabs.hpecorp.net
```

```
[oracle@soldb01 ~]$ sqlplus sys/%Oradata1@10.120.19.116:1521/mypdb1.us.rdlabs.hpecorp.net as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Fri Mar 30 13:39:43 2018

Copyright (c) 1982, 2016, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL>
```

Figure 34. Connecting to the pdb from a remote Oracle host

Running an OLTP load on a Docker-containerized Oracle pluggable database with Docker 3PAR volumes

It is straightforward to run an OLTP load on a Docker-containerized Oracle pdb. For this testing, the Swingbench Sales Order Entry workload was used to exercise the Docker volumes for Oracle built on HPE 3PAR storage with an OLTP-like mix of small block reads and writes.

Size redo log files, temp, and SOE tablespaces

For this database, a bigfile SOE tablespace of size 40 GB was created before starting data generation. For example:

```
SQL> create bigfile tablespace SOE datafile '/u02/app/oracle/oradata/MYCDB1/MYPDB1/soe.dbf' size 40g;

Tablespace created.

SQL>
```

The default temporary tablespace for the container database will be too small for generating multiple gigabytes of data. For this testing, a new default tablespace was created with size 7 GB.

The default sizes for redo log files will be three redo groups, with a file size of about 1 GB each. For this testing, these were replaced with three new groups, with log file sizes of 20 GB each.

Load the database with Sales Order Entry data

Connect to the containerized pluggable database and load the SOE tablespace, using the Swingbench oewizard.

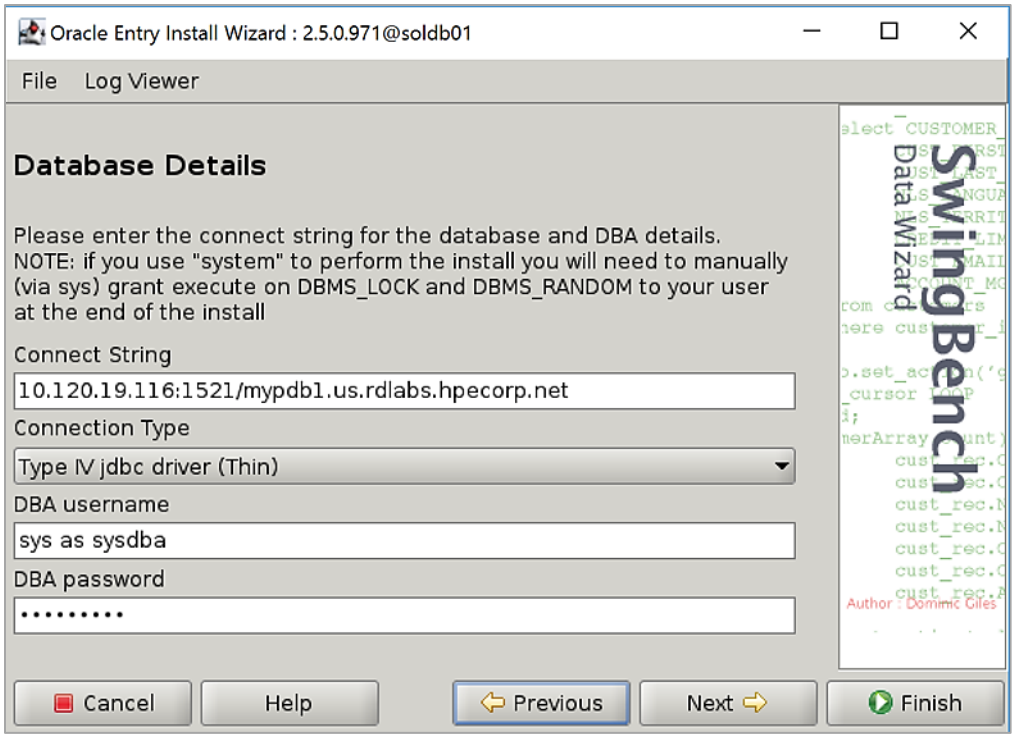


Figure 35. Connecting to the pluggable database

After connecting to the pluggable database, the oewizard will present the specified SOE tablespace for the creation of the schema.

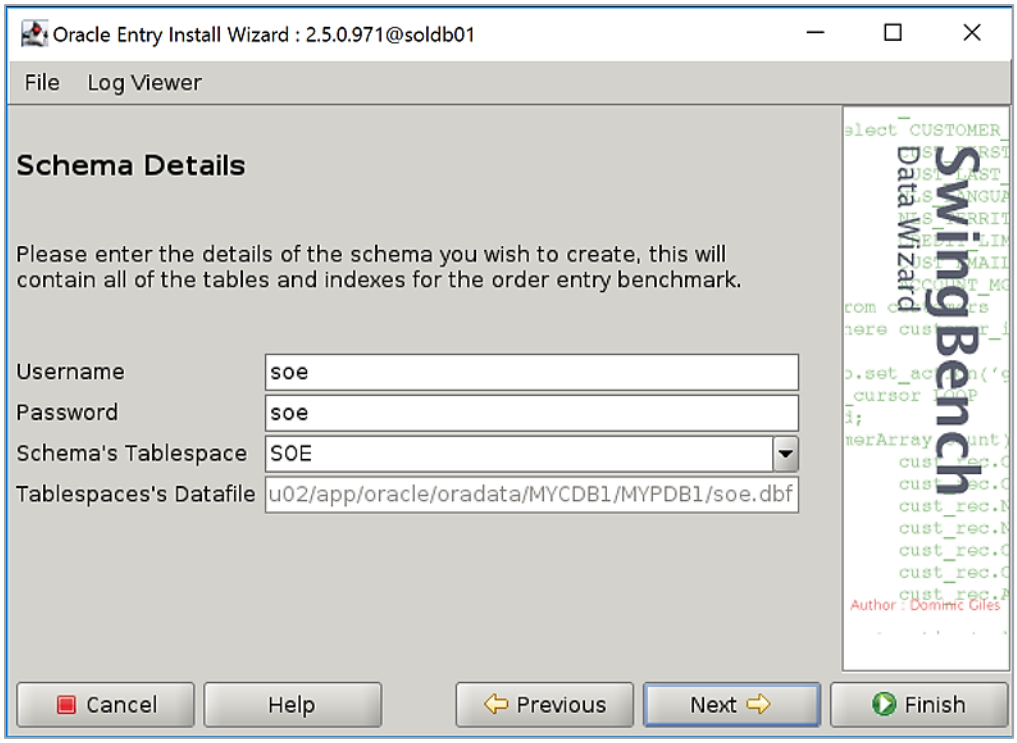


Figure 36. The oewizard tool displays the specified SOE tablespace

For this testing, the seed size was set at 10 GB, resulting in a total SOE tablespace data size of about 32 GB.

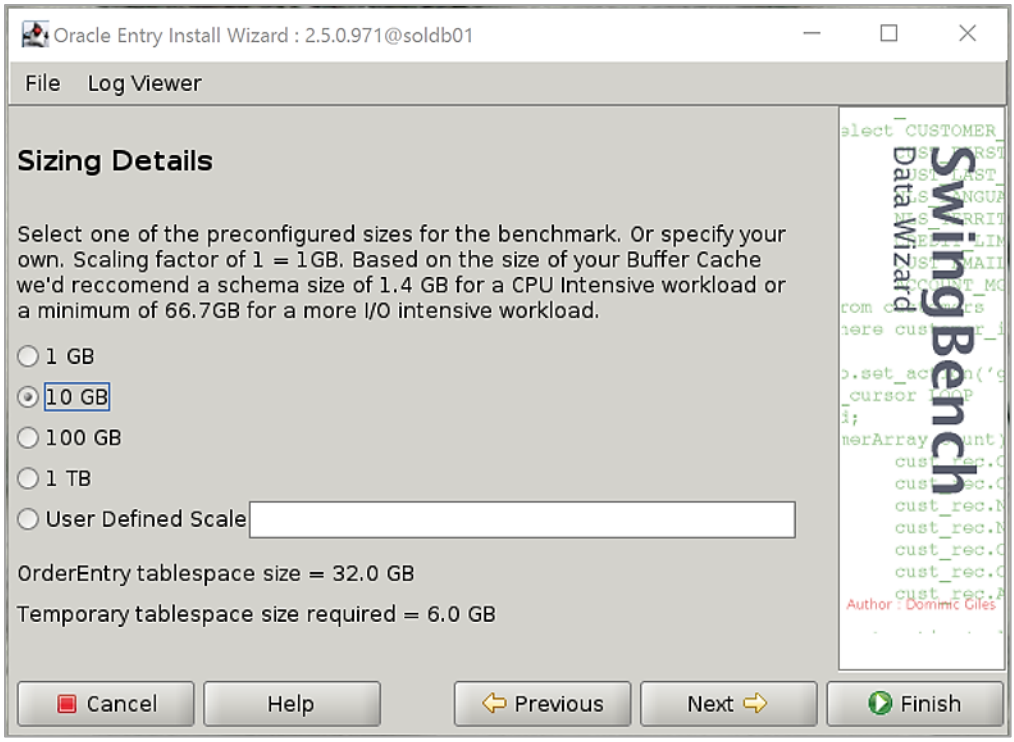


Figure 37. Setting the size of the benchmark

Run the SOE load

With a populated SOE schema, the Sales Order Entry workload can be run with Swingbench. For this testing, the SOE load was run from a remote host with 50 concurrent sessions.

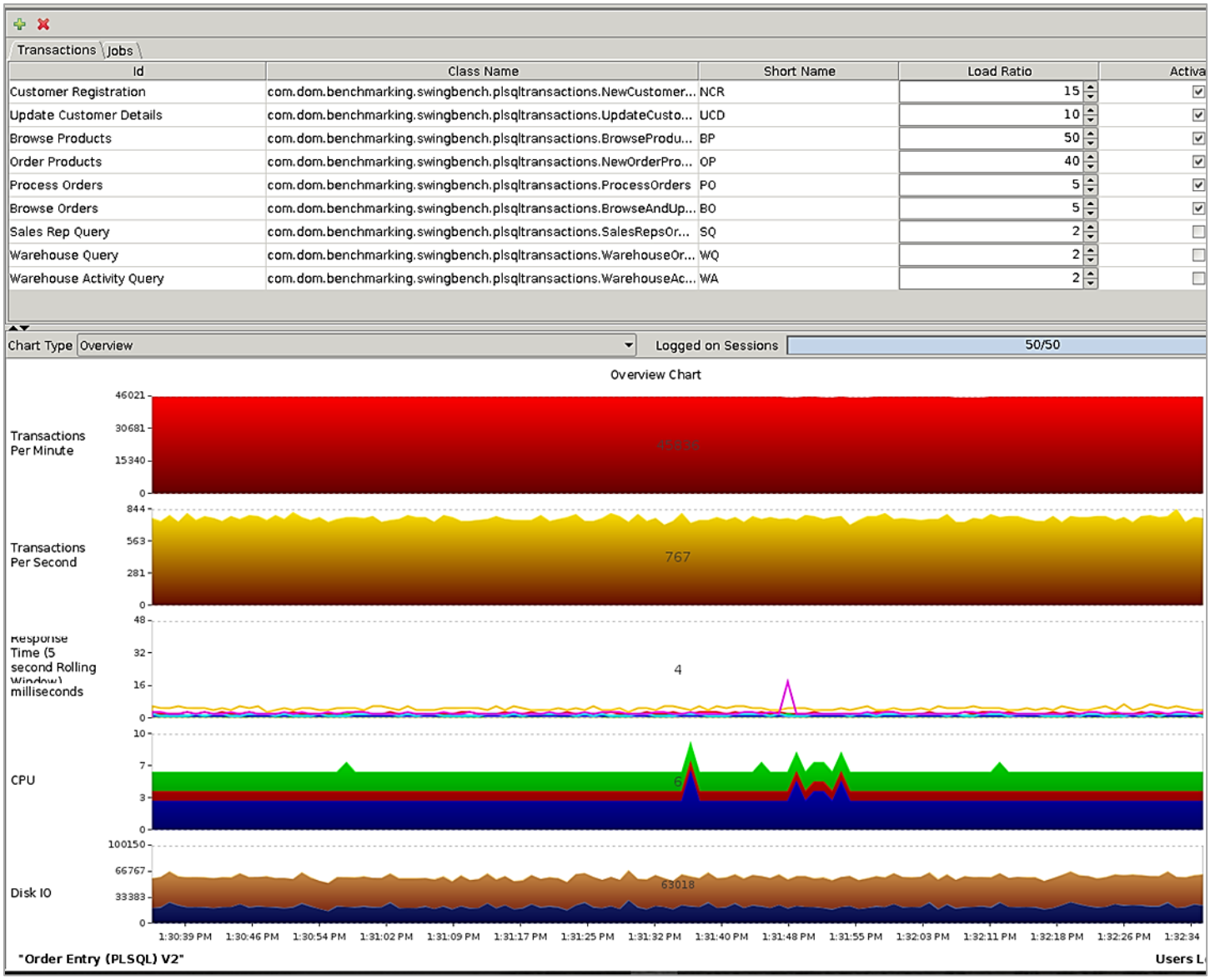


Figure 38. Swingbench running 50 "users" SOE load on Oracle Docker container pdb

Monitor performance of HPE 3PAR virtual volumes

Performance of the HPE 3PAR volumes under load from the container database can be monitored by using the `statvv` command in the HPE 3PAR CLI to stat the volumes with Docker volume names (`dcv-*`). As expected, IO activity is seen on the HPE 3PAR volumes labeled for Oracle data and Oracle redo logs.

```
      12:46:38 04/03/2018 r/w I/O per second    KBytes per sec    Svt ms    IOSz KB
                  VVname      Cur Avg   Max   Cur  Avg    Max  Cur  Avg  Cur   Avg Qlen
dcv-V2qWNNWzTHC2L1ZPFMQ8FQ   t    1   2  1233     4   83 129876 0.05 0.11  4.2  50.2    0
dcv-qDoGdYdQQoSIm2EWQZ75gA   t 6421 512 20583 57693 4929 192552 0.05 0.05  9.0   9.6    0
dcv-oPkQee.gT02p.hOLgtolng   t    0   5  1824     8 1066 469778 0.05 0.42 16.6 236.6    0
dcv-sb81cap0Qg2n-e-FRhgHog   t  684  55  1742  2937 2225 886473 0.07 0.07  4.3  40.8    0
------------------------------------------------------------------------------------------
                       4   t 7106 573       60643 8304        0.05 0.05  8.5  14.5    0
Press the enter key to stop...

      12:46:40 04/03/2018 r/w I/O per second    KBytes per sec    Svt ms    IOSz KB
                  VVname      Cur Avg   Max   Cur  Avg    Max  Cur  Avg  Cur   Avg Qlen
dcv-V2qWNNWzTHC2L1ZPFMQ8FQ   t    0   2  1233     0   83 129876 0.00 0.11  0.0  50.2    0
dcv-qDoGdYdQQoSIm2EWQZ75gA   t 6383 513 20583 56789 4939 192552 0.04 0.05  8.9   9.6    1
dcv-oPkQee.gT02p.hOLgtolng   t    0   5  1824     8 1066 469778 0.05 0.42 16.6 236.6    0
dcv-sb81cap0Qg2n-e-FRhgHog   t  701  55  1742  3081 2225 886473 0.07 0.07  4.4  40.7    0
------------------------------------------------------------------------------------------
                       4   t 7084 574       59878 8314        0.05 0.05  8.5  14.5    1
Press the enter key to stop...

      12:46:42 04/03/2018 r/w I/O per second    KBytes per sec    Svt ms   IOSz KB
                  VVname      Cur Avg   Max   Cur  Avg    Max  Cur  Avg Cur   Avg Qlen
dcv-V2qWNNWzTHC2L1ZPFMQ8FQ   t    0   2  1233     0   83 129876 0.00 0.11 0.0  50.2    0
dcv-qDoGdYdQQoSIm2EWQZ75gA   t 6455 514 20583 57669 4949 192552 0.05 0.05 8.9   9.6    0
dcv-oPkQee.gT02p.hOLgtolng   t    0   5  1824     0 1066 469778 0.00 0.42 0.0 236.6    0
dcv-sb81cap0Qg2n-e-FRhgHog   t  679  55  1742  2984 2225 886473 0.06 0.07 4.4  40.6    0
------------------------------------------------------------------------------------------
                       4   t 7134 575       60653 8324        0.05 0.05 8.5  14.5    0
Press the enter key to stop...
```

Figure 39. Using `statvv dcv-*` in the HPE 3PAR CLI to monitor the performance of volumes under load from Oracle Docker container db

```
20-14a1_20800 cli% showvv -showcols Comment dcv-qDoGdYdQQoSIm2EWQZ75gA
Comment
{"display_name": "OracleData", "type": "Docker", "name": "a83a0675-8750-4284-889b-6116419ef980", "volume_id": "a83a0675-8750-4284-889b-6116419ef980"}
------------------------------------------------------------------------------------------------------------------------------------------------

20-14a1_20800 cli% showvv -showcols Comment dcv-sb81cap0Qg2n-e-FRhgHog
Comment
{"display_name": "OracleRedo", "type": "Docker", "name": "b1bf3571-aa74-420d-a7fd-efc5461807a2", "volume_id": "b1bf3571-aa74-420d-a7fd-efc5461807a2"}
------------------------------------------------------------------------------------------------------------------------------------------------

20-14a1_20800 cli%
```

Figure 40. Showing volume labels for the workload-active volumes, dedicated to Oracle data and redo logs

Run an Oracle AWR report and use the Docker copy command to copy the report from inside the container

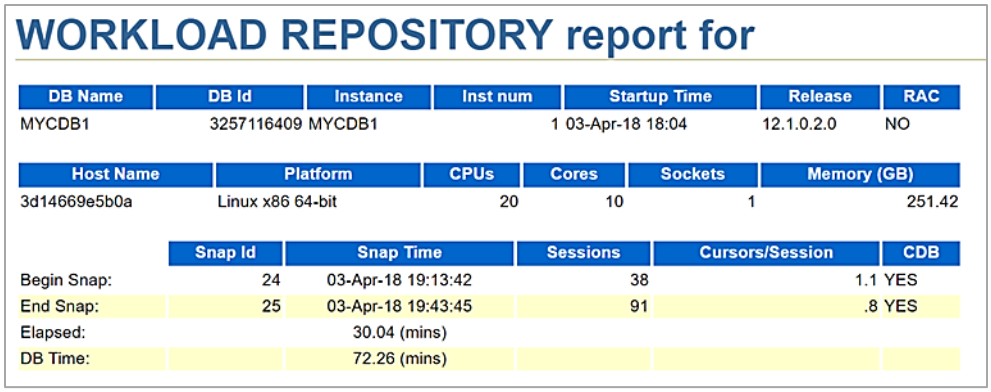
Automated Workload Repository reports and other Oracle diagnostics can be run from within the Docker container. Create an AWR report by running the SQL*Plus `awrrpt` command from `$ORACLE_HOME/rdbms/admin`.

```
SQL>/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/admin/rdbms/admin/awrrpt.sql
```

After the AWR report has been generated, Docker `cp` can be used to copy it from the container to another location for analysis.

```
# docker cp testdb:/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/admin/April3_50users.lst /tmp
```

The AWR report will show the container database for the database name, and service stats are broken down to the pluggable database level.



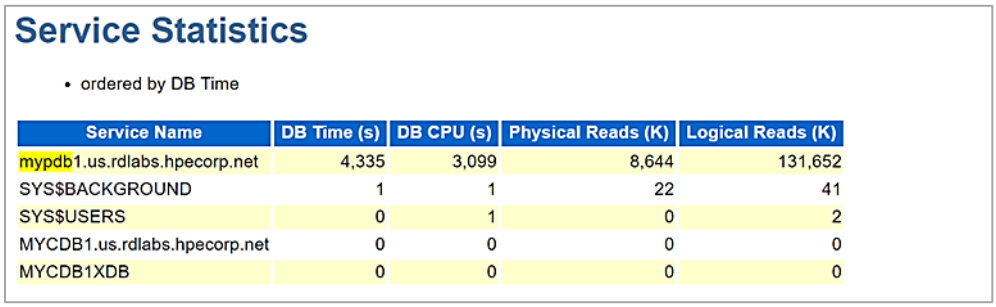
WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
MYCDB1	3257116409	MYCDB1	1	03-Apr-18 18:04	12.1.0.2.0	NO

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
3d14669e5b0a	Linux x86 64-bit	20	10	1	251.42

	Snap Id	Snap Time	Sessions	Cursors/Session	CDB
Begin Snap:	24	03-Apr-18 19:13:42	38	1.1	YES
End Snap:	25	03-Apr-18 19:43:45	91	.8	YES
Elapsed:		30.04 (mins)			
DB Time:		72.26 (mins)			

Figure 41. AWR report for the container database



Service Statistics

- ordered by DB Time

Service Name	DB Time (s)	DB CPU (s)	Physical Reads (K)	Logical Reads (K)
mypdb1.us.rdlabs.hpecorp.net	4,335	3,099	8,644	131,652
SYS$BACKGROUND	1	1	22	41
SYS$USERS	0	1	0	2
MYCDB1.us.rdlabs.hpecorp.net	0	0	0	0
MYCDB1XDB	0	0	0	0

Figure 42. Service statistics for the pluggable database

Running a DSS Load on a Docker-containerized Oracle pluggable database with Docker 3PAR volumes

Swingbench Sales History provides a DSS-style workload of large block random reads. The process of running Swingbench Sales History on an Oracle Docker container requires essentially the same steps as outlined above for the Swingbench Sales Order Entry load, using the Swingbench shwizard to load the Sales History schema, and then running the workload as the `sh` user. (Note that for this testing the Swingbench 2.5 shwizard was used for data loading and Swingbench 2.6 was used to run the workload with 24 threads/users.)

With 24 concurrent sessions, the large block read performance attained with this testing averaged ~2.6 GB per second Read IO from the database client perspective.

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	23.3	173.9	2.67	7.64
DB CPU(s):	4.3	32.0	0.49	1.41
Background CPU(s):	0.0	0.1	0.00	0.00
Redo size (bytes):	4,249.9	31,744.4		
Logical read (blocks):	449,580.3	3,358,086.7		
Block changes:	342.2	2,556.1		
Physical read (blocks):	339,309.9	2,534,434.9		
Physical write (blocks):	115.2	860.1		
Read IO requests:	183,166.6	1,368,141.2		
Write IO requests:	8.0	59.4		
Read IO (MB):	2,650.9	19,800.3		
Write IO (MB):	0.9	6.7		

Figure 43. Load profile detail from Sales History workload AWR report

Reusing an existing Oracle database on a Docker 3PAR volume

The Oracle Database 12.2.0.1 image in the Docker Store provides the ability to create persistent reusable data in a container database, using a specified mount point for the Docker data volume. By starting a database with a Docker data volume mounted within the container at /ORCL, this creates persistent data that can be reused by another container. For example, this command creates a database with all data, including binaries, fast recovery, and redo longs, on the data volume Oracle122.

```
# docker run -d -it --name testdb4 -v Oracle122:/ORCL store/oracle/database-enterprise:12.2.0.1
```

For a simple example,

1. Create the data volume “Oracle122” with the HPE 3PAR Volume Plug-in for Docker.
2. Run a new database container “testdb4” with volume “Oracle122” mounted at /ORCL.
3. Within the container create a bigfile tablespace “MYFOOT”.
4. After stopping and deleting the “testdb4” container, create a new container “testdb5” with the reused data volume.
5. A query shows that the “MYFOOT” tablespace has persisted into the new container.

Without this feature, attempts to reuse the volume with a new container would result in file permission errors.

```
[soldb06 ~]# docker volume create -d hpe -o size=200 --name=Oracle122 --label=oradata
Oracle122
[soldb06 ~]#
```

Figure 44. Creating the Docker data volume “Oracle122” for reuse on HPE 3PAR storage

```
[soldb06 ~]# docker run -d --env-file /etc/ora3.conf -p 1529:1521 -p 5509:5500 -it --name testdb4 -v Oracle122:/ORCL --shm-size=8g store/oracle/database-enterprise:12.2.0.1
Unable to find image 'store/oracle/database-enterprise:12.2.0.1' locally
12.2.0.1: Pulling from store/oracle/database-enterprise
4ce27fe12c04: Pull complete
9d3556e8e792: Pull complete
fc60a1a28025: Pull complete
0c32e4ed872e: Pull complete
b465d9b6e399: Pull complete
Digest: sha256:40760ac70dba2c4c70d0c542e42e082e8b04d9040d91688d63f728af764a2f5d
Status: Downloaded newer image for store/oracle/database-enterprise:12.2.0.1
66b232790d307410f0957364664edc36c520aa4483fe0d6cfeeab5425083242f
[soldb06 ~]#
```

Figure 45. Run the container with data volume "Oracle122" mounted at /ORCL

```
SQL> create bigfile tablespace MYFOOT datafile '/ORCL/u02/app/oracle/oradata/MYCDB1/myfoot.dbf' size 40g;

Tablespace created.

SQL> select tablespace_name from user_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
MYFOOT

6 rows selected.

SQL> exit
```

Figure 46. Create a new tablespace "MYFOOT"

```
[soldb06 ~]# docker stop testdb4
testdb4
[soldb06 ~]# docker rm testdb4
testdb4
[soldb06 ~]# docker run -d --env-file /etc/ora3.conf -p 1529:1521 -p 5509:5500 -it --name testdb5 -v Oracle122:/ORCL --shm-size=8g store/oracle/database-enterprise:12.2.0.1
db3c95d1c9bfd16b0193b292958a8b7886d973dad5b11573044287e5e23dec74
[soldb06 ~]# docker logs testdb5
```

Figure 47. Delete the "testdb4" container and run a new container with the same data volume "Oracle122"

```
[soldb06 ~]# docker exec --user oracle:oinstall -it testdb5 bin/bash
[oracle@db3c95d1c9bf /]$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Wed Apr 11 22:02:26 2018

Copyright (c) 1982, 2016, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL> select tablespace_name from user_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS
MYFOOT

6 rows selected.

SQL>
```

Figure 48. Verify the "MYFOOT" tablespace persists in the new container

Building an Oracle database custom Docker Container from Oracle scripts

A set of Oracle build scripts for Docker are available for download from GitHub at https://github.com/oracle/docker-images/.

To use the build scripts, download the `docker-images-master.zip` file and unzip it on the Docker server.

This creates a directory tree under `docker-images-master`, which includes branches for all Docker-containerized Oracle products.

```
[root@soldb04 docker-images-master]# ls
CODEOWNERS       GraalVM   OracleCloudInfrastructure  OracleEDQ               OracleInstantClient     OracleTuxedo
ContainerCloud   NoSQL     OracleCoherence            OracleFMWInfrastructure OracleJava              OracleWebCenterSites
CONTRIBUTING.md  OpenJDK   OracleDatabase             OracleGoldenGate        OracleRestDataServices  OracleWebLogic
GlassFish        OracleBI  OracleDataIntegrator       OracleHTTPServer        OracleSOASuite          README.md
[root@soldb04 docker-images-master]#
```

Figure 49. Oracle Docker images directory tree

In order to build an Oracle database image for Docker, it is necessary to obtain the licensed installation binary for the desired version of the Oracle database (available from the Oracle Technology Network) and copy it into the corresponding subdirectory of `OracleDatabase/dockerfiles`. After the Oracle installation binary is in the proper location, the Oracle Docker image can be built.

```
[root@soldb04 docker-images-master]# cp /software/linuxx64_12201_database.zip OracleDatabase/dockerfiles/12.2.0.1/
[root@soldb04 docker-images-master]# cd OracleDatabase/dockerfiles/
[root@soldb04 dockerfiles]# ls
11.2.0.2  12.1.0.2  12.2.0.1  buildDockerImage.sh
[root@soldb04 dockerfiles]# ./buildDockerImage.sh

Usage: buildDockerImage.sh -v [version] [-e | -s | -x] [-i] [-o] [Docker build option]
Builds a Docker Image for Oracle Database.

Parameters:
   -v: version to build
       Choose one of: 11.2.0.2  12.1.0.2  12.2.0.1
   -e: creates image based on 'Enterprise Edition'
   -s: creates image based on 'Standard Edition 2'
   -x: creates image based on 'Express Edition'
   -i: ignores the MD5 checksums
   -o: passes on Docker build option

* select one edition only: -e, -s, or -x

LICENSE UPL 1.0

Copyright (c) 2014-2017 Oracle and/or its affiliates. All rights reserved.

[root@soldb04 dockerfiles]#
```

Figure 50. Copying Oracle database installation binary into the corresponding directory, and display usage for `buildDockerImage.sh`

For example, to build an Oracle Enterprise Edition 12.2.0.1 image, ignoring the MD5 sum check, run `buildDockerImage.sh` as follows:

```
./buildDockerImage.sh -v 12.2.0.1 -e –i
```

The build process requires `http` access to Oracle's public yum repo and will do a full installation of the Oracle software within a Docker image. After the build is complete, it can be seen that both the Oracle database and an Oracle Linux 7-slim image have been created.

```
# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
oracle/database     12.2.0.1-ee         f2960dc1b4f6        15 minutes ago      13.3GB
oraclelinux         7-slim              9870bebfb1d5        2 months ago        118MB
#
```

The built Oracle database image can be run with parameters similar to those used with the ready-made images, although parameter names are different.[6] One convenient feature of the Oracle Docker build scripts is the ability to designate setup and startup scripts for additional customization of the database at runtime. The container is built to recognize the locations `/opt/oracle/scripts/setup` and `/opt/oracle/scripts/startup` and will automatically execute shell scripts and SQL scripts in those directories at setup or startup. To utilize this feature, map a directory containing the required scripts to one of those locations. Scripts will be executed in alpha order, so the desired order can be imposed by prefixing the script file name with a number. For example, a setup SQL script can be developed to automatically create an SOE tablespace and user:

```
# cat 01_createSOE.sql
ALTER SESSION SET CONTAINER=PDBTEST1;
CREATE bigfile TABLESPACE SOE DATAFILE '/opt/oracle/oradata/CDBTEST1/PDBTEST1/soe.dbf' size 1G
REUSE AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL;
CREATE USER SOE IDENTIFIED BY soe;
GRANT CONNECT TO soe;
GRANT CREATE TABLE, CREATE VIEW, CREATE PROCEDURE TO soe;
ALTER USER soe QUOTA UNLIMITED ON soe;
ALTER USER soe DEFAULT TABLESPACE soe;
exit;
#
```

[6] Refer to the Oracle Database on Docker README on github for information on runtime parameters.

To enable automatic execution of the script as `sysdba` when the Oracle container is first run, simply map the volume containing the script to the setup location `/opt/oracle/scripts/setup` using the "-v" parameter. For example:

```
# docker run --name dbtest1 -v /docker_fs/myscripts:/opt/oracle/scripts/setup
```

Successful execution of the script can be confirmed by monitoring the Docker log for the container:

```
Executing user defined scripts
/opt/oracle/runUserScripts.sh: running /opt/oracle/scripts/setup/01_createSOE.sql
Session altered.
Tablespace created.
User created.
Grant succeeded.
Grant succeeded.
User altered.
User altered.
```

The Oracle build scripts do not offer the same rapid setup as ready-made Oracle containers available from the Docker Store. They do require Oracle installation images, and the resulting Docker images are much larger. But for those who require greater flexibility, the build scripts can be modified to suit the user environment, and the option to implement automatic setup and startup scripts could add a lot of value in the deployment of customized database containers. For either type of Oracle database container, the HPE 3PAR Volume Plug-in for Docker offers the full power of HPE 3PAR to fulfill all Oracle database storage needs.

Summary

HPE 3PAR storage offers a unique set of features to meet the challenge of implementing Oracle database applications with outstanding workload performance, efficient storage utilization, high availability, and data protection. With models ranging from a few terabytes to many petabytes, the HPE 3PAR product family includes primary storage to fit the business requirements of every Oracle installation. For Docker environments, HPE provides the HPE 3PAR Volume Plug-in for Docker to enable HPE 3PAR to deliver enterprise-class persistent volumes for Oracle database containers. With HPE 3PAR, there is no need to compromise on storage quality or performance in implementing a containerized Oracle database.

HPE proof-of-concept

HPE recommends implementing a proof-of-concept using an Oracle test environment that matches as closely as possible the planned production or development environment. In this way, appropriate performance and scalability characterizations can be obtained. For help with a proof-of-concept, contact an HPE Services representative (hpe.com/us/en/services/consulting.html) or your HPE partner.

List of acronyms

AWR	Oracle Automatic Workload Repository
CDB	Oracle Container Database
CLI	Command line interface
DBA	Oracle Database Administrator
DSS	Decision Support System
FC	Fibre Channel
CPG	HPE 3PAR Common Provisioning Group
SSMC	HPE 3PAR Management Console
OLTP	Online transaction processing
PDB	Oracle pluggable database
SH	Swingbench Sales History schema
SOE	Swingbench Sales Order Entry schema
RAID MP	RAID Multiple Parity (RAID 6)
SQL*Plus	Oracle database utility used by admins

Resources and additional links

HPE 3PAR RAID 6: Securing your data investment hpe.com/h20195/V2/Getdocument.aspx?docname=a00000244enw

HPE 3PAR Red Hat Enterprise Linux, CentOS Linux, and Oracle Linux Implementation Guide h20565.www2.hpe.com/hpsc/doc/public/display?docId=c04448818

HPE 3PAR Storage Concepts Guide h20564.www2.hpe.com/hpsc/doc/public/display?docId=c04204225

HPE 3PAR Storage: A reference and best practices guide for HPE 3PAR Storage hpe.com/h20195/v2/GetPDF.aspx/4AA4-4524ENW.pdf

HPE 3PAR Storage software: Family data sheet hpe.com/h20195/v2/GetPDF.aspx/4AA2-8396ENN.pdf

HPE 3PAR OS Command Line Interface Reference support.hpe.com/hpsc/doc/public/display?docId=c04204279

HPE 3PAR Architecture h20195.www2.hpe.com/V2/Getdocument.aspx?docname=4AA3-3516ENW

Oracle Database 12.1 – Introduction to the Multitenant Architecture docs.oracle.com/database/121/CNCPT/cdbovrvw.htm#CNCPT89234

Swingbench dominicgiles.com dominicgiles.com/swingbench.html

Get Docker EE for Red Hat Enterprise Linux docs.docker.com/install/linux/docker-ee/rhel/

Docker post-installation steps for Linux docs.docker.com/install/linux/linux-postinstall/

HPE 3PAR Volume Plug-in for Docker – Docker Store: store.docker.com/plugins/hpe-3par-docker-volume-plugin

HPE 3PAR Volume Plug-in for Docker GitHub: github.com/hpe-storage/python-hpedockerplugin

Oracle Database Server Docker Image Documentation store.docker.com/images/oracle-database-enterprise-edition/plans/08cf8677-bb8f-453c-b667-6b0c24a388d4?tab=instructions

Oracle Database on Docker – GitHub github.com/oracle/docker-images/tree/master/OracleDatabase

Creating an Oracle Database Docker Image – Gerald on IT geraldonit.com/2017/08/21/creating-an-oracle-database-docker-image/

Learn more at HPE 3PAR Storage

hpe.com/us/en/storage/3par.html

a00047635ENW, May 2018