



# Quick Guide to Test InfoScale 7.1 Docker Plug-in using VMware WorkStation

For those who want to be familiar on how InfoScale 7.1 and Docker Plug-in integrate and works but only have access to their own laptop

Carlos Carrero  
Product Management

## Contents

<b>Introduction .....</b>	<b>3</b>
<b>Software .....</b>	<b>3</b>
<b>VMware WorkStation Configuration .....</b>	<b>4</b>
<b>Veritas InfoScale 7.1 Install and Configuration .....</b>	<b>4</b>
<b>Software Deployment.....</b>	<b>5</b>
<b>Configuration.....</b>	<b>7</b>
<b>Storage Configuration .....</b>	<b>8</b>
<b>Docker Engine &amp; InfoScale Plug-in Installation .....</b>	<b>10</b>
<b>InfoScale Docker Plug-in Installation .....</b>	<b>10</b>
<b>Docker Engine Installation .....</b>	<b>11</b>
<b>Plug-in Utilization.....</b>	<b>12</b>
<b>Testing data persistence .....</b>	<b>14</b>

# Quick Guide to Test InfoScale 7.1 and Docker Plug-in

## Introduction

If you have this document in your hands you probably are familiar with the [Veritas InfoScale Containers Group](#). This is a repository of demos, white papers and binaries that will help you adopt Docker technology without any fear when having to manage storage.

Containers are clearly the new wave in virtualization and they will create more agile data centers. They will allow better resources consolidation and will reduce development cycles. But all these benefits will go away if the storage is not managed accordingly. Veritas InfoScale, which is an evolution of the legacy Veritas Storage Foundation embraces Software Defined Storage to adapt to the new needs and requirements containers brings into the data center. This truly software solution provides scalability, data persistency, copy management, resiliency, IO acceleration, quality of service, encryption and disaster recovery for the modern containerized data center.

In order to keep the agile environment, Veritas has created a Docker Plug-in that integrates Docker and InfoScale to allow the Docker user to get all that functionality without having to manage a storage platform, and without having to call Storage Administrators (and wait long times) to perform typical storage management chores.

This paper is a quick guide for those who just want to get familiar with the solution without having to make any big investment, so VMware WorkStation is used to create a simple 2 node configuration.

## Software

This paper will be using the new version of InfoScale 7.1 that introduces among other capabilities features like Quality of Service, Encryption and support up to 64 nodes when using local storage (Flexible Storage Sharing).

InfoScale 7.1 can be downloaded in this link:

<https://www.veritas.com/content/trial/en/us/veritas-infoscale-solutions.html>

In that link download either the ISO or the tar image for Red Hat Enterprise Linux:

## Veritas InfoScale Solutions Download

### Multiple File Download

File	Size	
<a href="#">Veritas_InfoScale_7.1_RHEL.iso</a>	1.3 GB	<a href="#">Download Now</a> 
<a href="#">Veritas_InfoScale_7.1_RHEL.tar.gz</a>	1 GB	<a href="#">Download Now</a> 

The Docker Plug-in is so far a controlled release package that is currently in tech-preview mode. The plug-in is continually updated so make sure you get access to the latest version. In order to download it you have to be a member of the Veritas community and join the [Veritas InfoScale Containers](#)

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

[Group](#) because legal reasons. Once you have been approved as a member, this is the link to download the plug-in:

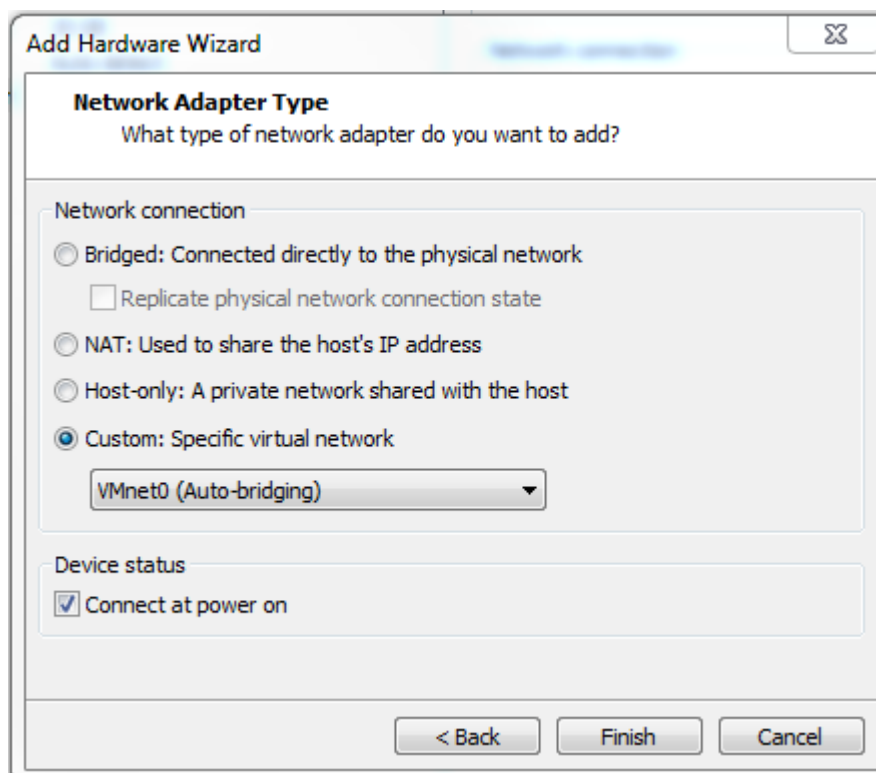
<https://www.veritas.com/community/downloads/5-plug-trialware-download>

### VMware WorkStation Configuration

When creating Virtual Machines using VMware WorkStation, use these options:

- **Virtual Machine:** RHEL 7 64-bit
- **Cores:** Create the VM with 2 processors, 1 core per processor
- **Memory:** Minimum of 2GB memory per VM
- **Network Type:** Use network address translation (NAT)
- **SCSI Controller TYPE :** LSI Logic SAS
- **Disk Type:** SCSI
- **Select Disk:** Create a new virtual disk

The VMs will need two private networks that are used for the cluster inter-communication. You can create those two networks using a custom virtual network as specified in the following figure:



### Veritas InfoScale 7.1 Install and Configuration

# Quick Guide to Test InfoScale 7.1 and Docker Plug-in

## Software Deployment

Once you have downloaded the Veritas\_InfoScale\_7.1\_RHEL package, either using the ISO or tar image, make it available to any of the VMs and from there execute the installer script (this only needs to be executed in one of the servers and will deploy and configure in all of them):

```
/root/dvd1-redhatlinux/rhel7_x86_64  
[root@docker1 rhel7_x86_64]# ./installer
```

Choose “I” to Install a Product:

Task Menu:

```
P) Perform a Pre-Installation Check      I) Install a Product  
C) Configure a Product Component       G) Upgrade a Product  
O) Perform a Post-Installation Check    U) Uninstall a Product  
L) License a Product                   S) Start a Product  
D) View Product Descriptions           X) Stop a Product  
R) View Product Requirements           ?) Help
```

Enter a Task: [P,I,C,G,O,U,L,S,D,X,R,?]

Select **option 4 for Veritas InfoScale Enterprise** and select “y” to configure it after installing the packages:

```
1) Veritas InfoScale Foundation  
2) Veritas InfoScale Availability  
3) Veritas InfoScale Storage  
4) Veritas InfoScale Enterprise  
b) Back to previous menu
```

Select a product to install: [1-4,b,q,?] **4**

Would you like to configure InfoScale Enterprise after installation? [y,n,q] (n) **y**

Select **option 4 for SFCFSHA**:

```
1) Cluster Server (VCS)  
2) Storage Foundation (SF)  
3) Storage Foundation and High Availability (SFHA)  
4) Storage Foundation Cluster File System HA (SFCFSHA)  
5) Storage Foundation for Oracle RAC (SF Oracle RAC)
```

Select a component to configure: [1-5,q] **4**

Enter the system names. In our case, the name of the VMs are *docker1* and *docker2*. Make sure they can reach each other using an IP address and host name:

Enter the system names separated by spaces: [q,?] **docker1 docker2**

Let the installer to set ssh password-less temporarily for you:

```
Would you like the installer to setup ssh or rsh communication automatically  
between the systems?  
Super user passwords for the systems will be asked. [y,n,q,?] (y) y
```

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

The installer will check if all rpms required are installed. If not, the installer can automatically use yum to install them (option 1):

```
The following required OS rpms (or higher version) were not found on docker1:
  net-tools-2.0-0.17.20131004git.el7.x86_64 bc-1.06.95-13.el7.x86_64 perl-
Socket-2.010-3.el7.x86_64
ed-1.9-4.el7.x86_64 perl-Exporter-5.68-3.el7.noarch ksh-20120801-22.el7.x86_64
perl-5.16.3-285.el7.x86_64
```

```
The following required OS rpms (or higher version) were not found on docker2:
  net-tools-2.0-0.17.20131004git.el7.x86_64 bc-1.06.95-13.el7.x86_64 perl-
Socket-2.010-3.el7.x86_64
ed-1.9-4.el7.x86_64 perl-Exporter-5.68-3.el7.noarch ksh-20120801-22.el7.x86_64
perl-5.16.3-285.el7.x86_64
```

The installer provides some guidance about how to install OS rpms using native methods, like yum, or how to manually install the required OS rpms.

- 1) **Install the missing required OS rpms with yum, if yum is configured on the systems**
- 2) Install the missing required OS rpms manually, (detailed steps are provided)
- 3) Do not install the missing required OS rpms

How would you like to install the missing required OS rpms? [1-3,q,?] **(1)**

**Enable keyless (option 2) for the demo purposes:**

- 1) Enter a valid license key
- 2) **Enable keyless licensing and complete system licensing later**

How would you like to license the systems? [1-2,q] **(2)**

**And option 4 to temporarily license Veritas InfoScale Enterprise:**

- 1) Veritas InfoScale Foundation
- 2) Veritas InfoScale Availability
- 3) Veritas InfoScale Storage
- 4) **Veritas InfoScale Enterprise**
- b) Back to previous menu

Which product would you like to register? [1-4,b,q] **(4)**

**For this testing there is no need of I/O fencing at this point:**

Do you want to configure I/O Fencing in enabled mode? [y,n,q,?] (y) **n**

**Choose a name for your cluster:**

Enter the unique cluster name: [q,?] **docker**

**Let the installer to auto-detect the “Low Latency Transport” devices. These will be the two private network interfaces that we configured previously during the VM setup:**

- 1) Configure the heartbeat links using LLT over Ethernet
- 2) Configure the heartbeat links using LLT over UDP

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

- 3) Configure the heartbeat links using LLT over RDMA
- 4) **Automatically detect configuration for LLT over Ethernet**
- b) Back to previous menu

How would you like to configure heartbeat links? [1-4,b,q,?] **(4)**

If the Virtual Machines have been configured properly with the two private networks, the installer should detect it automatically:

```
Configuring LLT links: 100%

Estimated time remaining: (mm:ss) 0:00
4 of 4

Checking system NICs on docker1 ..... 3 NICs found
Checking system NICs on docker2 ..... 3 NICs found
Checking network links ..... 3 links found
Setting link priority ..... Done

Enter a unique cluster ID number between 0-65535: [b,q,?] (58487)
```

**Use the proposed ID number.** There is no need here to check whether it is in use as you are using WorkStation and we will not expect another cluster on the same network.

You can finally review the configuration before going ahead:

```
Cluster information verification:

Cluster Name:      docker
Cluster ID Number: 58487

Private Heartbeat NICs for docker1:
  link1=en033554960
  link2=en050332184
Low-Priority Heartbeat NIC for docker1:
  link-lowpril=en016777736

Private Heartbeat NICs for docker2:
  link1=en033554960
  link2=en050332184
Low-Priority Heartbeat NIC for docker2:
  link-lowpril=en016777736

Is this information correct? [y,n,q,?] (y)
```

## Configuration

If we previously answered “y” when asked about configuring the software, the installer just go ahead and now keep asking configuration questions in order to finish the deployment. Most of the default answers will be good enough.

No need to use any virtual IP address:

The following data is required to configure the Virtual IP of the Cluster:

A public NIC used by each system in the cluster

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

A Virtual IP address and netmask

Do you want to configure the Virtual IP? [y,n,q,?] (n) **n**

### Accept secure mode by default:

Would you like to configure the VCS cluster in secure mode? [y,n,q,?] (**y**)

Do you want to grant read access to everyone? [y,n,q,?] (**n**)

Do you want to provide any usergroups that you would like to grant read access?  
[y,n,q,?] (y) **n**

- 1) Configure the cluster in secure mode
- 2) Configure the cluster in secure mode with FIPS
- b) Back to previous menu

Select the option you would like to perform [1-2,b,q,?] (1) **1**

### Other questions that will be asked and where defaults are good for this testing purpose:

Do you want to configure SMTP notification? [y,n,q,?] (**n**)

Do you want to configure SNMP notification? [y,n,q,?] (**n**)

Do you want to configure the Global Cluster Option? [y,n,q,?] (**n**)

### The next step will restart the processes and will perform the configuration:

Do you want to stop InfoScale Enterprise processes now? [y,n,q,?] (**y**)

### The configuration process will start:

Logs are being written to /var/tmp/installer-201605031034Xhi while installer is in progress

Starting SFCFSHA: 76%

Estimated time remaining: (mm:ss) 1:35  
23 of 30

```
Performing SFCFSHA configuration ..... Done
Starting veki ..... Done
Starting vxdmp ..... Done
Starting vxio ..... Done
Starting vxspec ..... Done
...
...
...
```

Once completed, the cluster will be ready to be configured with storage and to be used with Docker once the plug-in is installed.

## Storage Configuration

Before adding storage to each of the VMs, it is a good practice to set the flag `disk.EnableUUID = "TRUE"` in the VM configuration file. When using VMware WorkStation, power off the VM and edit the file to add that entry:



## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
flag disk.EnableUUID = "TRUE"
```

Now we need to add some local storage to each of the VMs. For testing purposes, we can add just one local VMDK to each of the VMs. For each of the VMs, go to “Settings” and click on the “Add” button and select “Hard Disk”. You can store the virtual disk with the VM.

Once the disk is created, you can go to each VM shell and run this command:

```
[root@docker1 ~]# vxdisk list
DEVICE          TYPE          DISK          GROUP          STATUS
docker1_disk_1  auto:LVM      -             -              LVM
docker1_disk_2 auto:none    -             -              online invalid
```

At the second node:

```
[root@docker2 ~]# vxdisk list
DEVICE          TYPE          DISK          GROUP          STATUS
docker2_disk_1  auto:LVM      -             -              LVM
docker2_disk_2 auto:none    -             -              online invalid
```

Note that in our configuration, the disk listed as “online invalid” is the new one added. The name of the disk is <hostname>\_<disk>\_<number>, so we can clearly identify each disk within the cluster.

In each of the nodes, initialize the local disk to be used. On docker1:

```
[root@docker1 ~]# /opt/VRTS/bin/vxdisksetup -i docker1_disk_2
```

On docker2:

```
[root@docker2 ~]# /opt/VRTS/bin/vxdisksetup -i docker2_disk_2
```

In any of the nodes you can see how now the disks are “online” after they have been initialized:

```
[root@docker1 ~]# vxdisk -o cluster list
DEVICE          MEDIA          SIZE (MB)     GROUP          NODES          STATE
docker1_disk_2  hdd            20480         -              1              online
docker2_disk_2  hdd            20480         -              1              online
```

The final step is to create a pool of storage (disk group) that will contain all the disks that will be used across the cluster. That includes, in this example, the local storage from docker1 and docker2 nodes.

From the list of disks we got from the previous command, execute the `vxdg` command to create a disk group named `dockerdg`:

```
[root@docker1 ~]# vxdg -o fss -s init dockerdg docker1_disk_2 docker2_disk_2
```

We can see how now those disks are included in the `dockerdg` disk group:

# Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
[root@docker1 ~]# vxdisk -o cluster list
DEVICE                MEDIA      SIZE (MB)  GROUP      NODES      STATE
docker1_disk_2        hdd       20480     dockerdg   1          online
docker2_disk_2        hdd       20480     dockerdg   1          online
```

From this point, we will install the Docker Plug-in and it will take care of storage provisioning.

## Docker Engine & InfoScale Plug-in Installation

### InfoScale Docker Plug-in Installation

Make sure you have downloaded the latest version of the plug-in from:

<https://www.veritas.com/community/downloads/5-plugin-trialware-download>

Remember that you have to become a member of the [Veritas InfoScale Containers Group](#).

You will get the file: VRTSdocker-plugin-1.1-Linux.x86\_64.rpm. Install the package in each of the nodes:

```
[root@docker1 ~]# yum install VRTSdocker-plugin-1.1-Linux.x86_64.rpm
Loaded plugins: product-id, rhnplugin, subscription-manager
This system is receiving updates from RHN Classic or Red Hat Satellite.
Examining VRTSdocker-plugin-1.1-Linux.x86_64.rpm: VRTSdocker-plugin-1.1-
Linux.x86_64
Marking VRTSdocker-plugin-1.1-Linux.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package VRTSdocker-plugin.x86_64 0:1.1-Linux will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package                Arch      Version
Repository              Size
=====
Installing:
  VRTSdocker-plugin      x86_64    1.1-Linux
/VRTSdocker-plugin-1.1-Linux.x86_64 6.7 M
```

Transaction Summary

```
=====
Install 1 Package

Total size: 6.7 M
Installed size: 6.7 M
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
```

# Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
Warning: RPMDB altered outside of yum.
Installing : VRTSdocker-plugin-1.1-Linux.x86_64
1/1
ln -s '/usr/lib/systemd/system/vxinfoscale-docker.service'
'/etc/systemd/system/multi-user.target.wants/vxinfoscale-docker.service'
Redirecting to /bin/systemctl start vxinfoscale-docker.service
Verifying : VRTSdocker-plugin-1.1-Linux.x86_64
1/1

Installed:
  VRTSdocker-plugin.x86_64 0:1.1-Linux

Complete!
[root@docker1 ~]#
```

## Docker Engine Installation

To install docker-engine you can follow Docker's instructions:

<https://docs.docker.com/engine/installation/linux/rhel/>

In each of the nodes:

Add the yum repo:

```
$ sudo tee /etc/yum.repos.d/docker.repo <<-EOF
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF
```

Install the Docker package in each node:

```
[root@docker1 ~]# yum install docker-engine
...
...
Installed:
  docker-engine.x86_64 0:1.11.1-1.e17.centos

Dependency Installed:
  bzip2-libs.i686 0:1.0.6-13.e17      elfutils-libelf.i686 0:0.163-3.e17      elfutils-
  libs.i686 0:0.163-3.e17

Updated:
  dracut.x86_64 0:033-359.e17          initscripts.x86_64 0:9.49.30-1.e17

Dependency Updated:
  bzip2-libs.x86_64 0:1.0.6-13.e17      device-mapper.x86_64
7:1.02.107-5.e17
  device-mapper-event.x86_64 7:1.02.107-5.e17      device-mapper-event-
libs.x86_64 7:1.02.107-5.e17
  device-mapper-libs.i686 7:1.02.107-5.e17      device-mapper-libs.x86_64
7:1.02.107-5.e17
  device-mapper-persistent-data.x86_64 0:0.5.5-1.e17      dracut-config-rescue.x86_64
0:033-359.e17
  dracut-network.x86_64 0:033-359.e17      elfutils-libelf.x86_64
0:0.163-3.e17
```

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
elfutils-libs.x86_64 0:0.163-3.e17
libgudev1.x86_64 0:219-19.e17
lvm2-libs.x86_64 7:2.02.130-5.e17
systemd-devel.x86_64 0:219-19.e17
systemd-libs.x86_64 0:219-19.e17
19.e17
kmod.x86_64 0:20-5.e17
lvm2.x86_64 7:2.02.130-5.e17
systemd.x86_64 0:219-19.e17
systemd-libs.i686 0:219-19.e17
systemd-sysv.x86_64 0:219-
```

Complete!

Start the service:

```
[root@docker1 ~]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@docker1 ~]#
```

And make sure Docker start with a system boot:

```
[root@docker1 ~]# chkconfig docker on
Note: Forwarding request to 'systemctl enable docker.service'.
ln -s '/usr/lib/systemd/system/docker.service' '/etc/systemd/system/multi-
user.target.wants/docker.service'
[root@docker1 ~]#
```

## Plug-in Utilization

Once the platform has been initialized, containers can be created and volumes will be automatically carved and attached to the containers as needed.

Initially we can check there are no volumes created:

```
[root@docker1 ~]# docker volume ls
DRIVER          VOLUME NAME
[root@docker1 ~]#
```

To create a container where persistent storage is needed, use the veritas volume driver:

```
[root@docker1 ~]# docker volume create -d veritas --name volumel -o size=300m
volumel
```

We can list it:

```
[root@docker1 ~]# docker volume ls
DRIVER          VOLUME NAME
veritas       volumel
veritas        volumel_dcl
[root@docker1 ~]#
```

Because the driver utilization, Docker used InfoScale to create a volume which has two plexes underneath, one residing in each of the servers, so resiliency for the persistent data is automatically achieved. There is no need for the Docker user to know these details as the Veritas driver hides all those tasks, but here we can see the `vxprint` command where the output shows two plexes, one in each of the disks that we created previously:

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
[root@docker1 ~]# vxprint
Disk group: dockerdg

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
dg dockerdg      dockerdg      -        -        -        -        -
dm docker1_disk_2 docker1_disk_2 -    41875568 -        -        -        -
dm docker2_disk_2 docker2_disk_2 -    41875568 -        REMOTE  -        -

v  volume1      fsgen         ENABLED  614400  -        ACTIVE  -        -
pl volume1-01    volume1       ENABLED  614400  -        ACTIVE  -        -
sd docker1_disk_2-01 volume1-01  ENABLED  614400  0        -        -        -
pl volume1-02    volume1       ENABLED  614400  -        ACTIVE  -        -
sd docker2_disk_2-01 volume1-02  ENABLED  614400  0        -        -        -
dc volume1_dco   volume1       -        -        -        -        -
v  volume1_dcl   gen          ENABLED  67840  -        ACTIVE  -        -
pl volume1_dcl-01 volume1_dcl  ENABLED  67840  -        ACTIVE  -        -
sd docker1_disk_2-02 volume1_dcl-01  ENABLED  67840  0        -        -        -
pl volume1_dcl-02 volume1_dcl  ENABLED  67840  -        ACTIVE  -        -
sd docker2_disk_2-02 volume1_dcl-02  ENABLED  67840  0        -        -        -
[root@docker1 ~]#
```

NOTE: The <volume>\_dcl is just used by InfoScale to provide Fast Resync.

We can inspect the volume:

```
[root@docker1 ~]# docker volume inspect volume1
[
  {
    "Name": "volume1",
    "Driver": "veritas",
    "Mountpoint": "/dockerdfs/volume1_dockerdg",
    "Labels": {}
  }
]
```

Also a volume can be created when the container is created:

```
[root@docker1 ~]# docker run --name mysql1 -v mysql1:/var/lib/mysql --volume-driver
veritas -e MYSQL_ROOT_PASSWORD=root -d mysql:5.6
4f725dd5a8f2c82af3bd85142a3fc248777d9bda1ffbbf0f77b6af8cfd68c034

[root@docker1 ~]# docker volume ls
DRIVER          VOLUME NAME
veritas        mysql1
veritas         mysql1_dcl
veritas         volume1
veritas         volume1_dcl
```

Or we can create the volume in advance using a desired size:

```
[root@docker1 ~]# docker volume create -d veritas --name mysql2 -o size=900m
mysql2
```

And then use it with a container:

```
[root@docker1 ~]# docker run --name mysql2 -v mysql2:/var/lib/mysql --volume-driver
veritas -e MYSQL_ROOT_PASSWORD=root -d mysql:5.6
e21adf0b33fe4960fdb4032092603b07ff974c1b842c145c109acce9715c0038
```

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
[root@docker1 ~]# docker volume ls
DRIVER          VOLUME NAME
veritas         mysql1
veritas         mysql1_dcl
veritas        mysql2
veritas         mysql2_dcl
veritas         volume1
veritas         volume1_dcl
```

Also, the volume can be finally removed using Docker CLI:

```
[root@docker1 ~]# docker volume rm volume1
volume1
```

```
[root@docker1 ~]# docker volume ls
DRIVER          VOLUME NAME
veritas         mysql1
veritas         mysql1_dcl
veritas         mysql2
veritas         mysql2_dcl
```

## Testing data persistence

We can use one of the mysql containers created to put some data:

```
[root@docker1 ~]# docker exec -it mysql1 bash

root@4f725dd5a8f2:/# mysql -proot
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database ccarrero;
Query OK, 1 row affected (0.07 sec)

mysql> use ccarrero;
Database changed
mysql> create table kids (name text, age integer);
Query OK, 0 rows affected (0.16 sec)

mysql> insert into kids values ('Marina', 7);
Query OK, 1 row affected (0.02 sec)

mysql> insert into kids values ('Diego', 3);
Query OK, 1 row affected (0.07 sec)

mysql> select * from kids;
+-----+-----+
| name  | age  |
```

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
+-----+-----+
| Marina |    7 |
| Diego  |    3 |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

To test data persistency, power off the VM where we have that container running (docker1 in my case). Log in into the other server (docker2) and bring up a mysql container pointing to the volume that we had already created:

```
[root@docker2 ~]# docker run --name mysql1 -v mysql1:/var/lib/mysql --volume-driver
veritas -d mysql:5.6

aef961b96571250a37671dfc40e9f92630b7c4418275db3069b86daffb9d5f59
```

And we can connect to the database and check the data is there:

```
[root@docker2 ~]# docker exec -it mysql1 bash

root@aef961b96571:/# mysql -proot
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use ccarrero;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from kids;
+-----+-----+
| name   | age  |
+-----+-----+
| Marina |    7 |
| Diego  |    3 |
+-----+-----+
2 rows in set (0.05 sec)

mysql>
```

What did happen under the covers is that Veritas InfoScale took care of data resiliency. Given that the volume has been created with a mirror configuration by default, each node had a plex with a copy of the data. We can use the vxprint command to inspect the current configuration and verify that one of the plex has been disabled and the other one is still active:

## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

```
[root@docker2 ~]# vxprint
Disk group: dockerdg

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
dg dockerdg      dockerdg      -        -        -        -        -        -

dm docker1_disk_2 -          -          -        -        -        NODEVICE -        -
dm docker2_disk_2 docker2_disk_2 -        41875568 -        -        -        -        -

v  mysql1        fsgen         ENABLED  409600  -        ACTIVE  -        -
pl mysql1-01     mysql1        DISABLED 409600  -        NODEVICE -        -
sd docker1_disk_2-03 mysql1-01  DISABLED 409600  0        NODEVICE -        -
pl mysql1-02     mysql1        ENABLED 409600  -        ACTIVE  -        -
sd docker2_disk_2-03 mysql1-02  ENABLED 409600  0        -        -        -
dc mysql1_dco    mysql1        -        -        -        -        -        -
v  mysql1_dcl    gen           ENABLED  67840  -        ACTIVE  -        -
pl mysql1_dcl-01 mysql1_dcl    DISABLED 67840  -        NODEVICE -        -
sd docker1_disk_2-04 mysql1_dcl-01 DISABLED 67840  0        NODEVICE -        -
pl mysql1_dcl-02 mysql1_dcl    ENABLED  67840  -        ACTIVE  -        -
sd docker2_disk_2-04 mysql1_dcl-02 ENABLED 67840  0        -        -        -
```

Also, once the VM is active again, InfoScale resynchronize just the last changes occurred in the volume and the two plexes keep being used as normal:

```
[root@docker2 ~]# vxprint
Disk group: dockerdg

TY NAME          ASSOC          KSTATE  LENGTH  PLOFFS  STATE  TUTILO  PUTILO
dg dockerdg      dockerdg      -        -        -        -        -        -

dm docker1_disk_2 docker1_disk_2 -        41875568 -        REMOTE  -        -
dm docker2_disk_2 docker2_disk_2 -        41875568 -        -        -        -

v  mysql1        fsgen         ENABLED  409600  -        ACTIVE  -        -
pl mysql1-01     mysql1        ENABLED  409600  -        ACTIVE  -        -
sd docker1_disk_2-03 mysql1-01  ENABLED 409600  0        -        -        -
pl mysql1-02     mysql1        ENABLED  409600  -        ACTIVE  -        -
sd docker2_disk_2-03 mysql1-02  ENABLED 409600  0        -        -        -
dc mysql1_dco    mysql1        -        -        -        -        -        -
v  mysql1_dcl    gen           ENABLED  67840  -        ACTIVE  -        -
pl mysql1_dcl-01 mysql1_dcl    ENABLED 67840  -        ACTIVE  -        -
sd docker1_disk_2-04 mysql1_dcl-01 ENABLED 67840  0        -        -        -
pl mysql1_dcl-02 mysql1_dcl    ENABLED 67840  -        ACTIVE  -        -
sd docker2_disk_2-04 mysql1_dcl-02 ENABLED 67840  0        -        -        -
```

All this was transparent for the Docker user that was able to keep running their containers using persistent storage from any of the nodes within the cluster.



## Quick Guide to Test InfoScale 7.1 and Docker Plug-in

### About Veritas Technologies LLC

Veritas Technologies LLC enables organizations to harness the power of their information, with solutions designed to serve the world's largest and most complex heterogeneous environments. Veritas works with 86 percent of Fortune 500 companies today, improving data availability and revealing insights to drive competitive advantage.

For specific country offices and contact numbers, please visit our website.

Veritas World Headquarters  
500 East Middlefield Road  
Mountain View, CA 94043  
+1 (650) 933 1000  
[www.veritas.com](http://www.veritas.com)

© 2015 Veritas Technologies LLC. All rights reserved.  
Veritas and the Veritas Logo are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.