Dell Storage Center with Flocker

Daniel Tan, UNIX / Linux Product Specialist
Dell Storage Applications Engineering
November 2015

Dell Best Practices

Revisions

Date	Revision	Description	Author
November 2015	1	Initial release	Daniel Tan

Table of contents

Executive summary

Docker® is built and architected around the basis of stateless compute/container resources. However, in certain use cases, a stateful compute/container resource may be more applicable to the business needs. In addition, Docker does not natively provide data portability when these compute/container resources are moved from one host to an alternate host.

A stateful compute/container resource would address these two needs by providing a framework for data persistence and data portability when moving compute/container resources from one host to an alternate host.

This paper presents the use and configuration of Dell™ Storage SC Series arrays with Flocker™ by ClusterHQ™ and highlights its strengths and value with use case scenarios.

1 Overview

Flocker is an agent-based cluster framework that piggybacks on top of Docker and Docker services to manage and orchestrate Docker-based containers while providing data volume management services allowing for data persistence and portability. Additionally, the Flocker framework provides transparency by allowing access to container resources regardless of the host where the container may actively reside.

The main components of Flocker include:

- Flocker CLI (this provides the `flocker-*` command set to manage Flocker deployments)
- Flocker Control Service (this is installed / run on one of the Node machines)
- Flocker Node Agent (one or multiple hosts)

The Flocker framework is logically represented below.

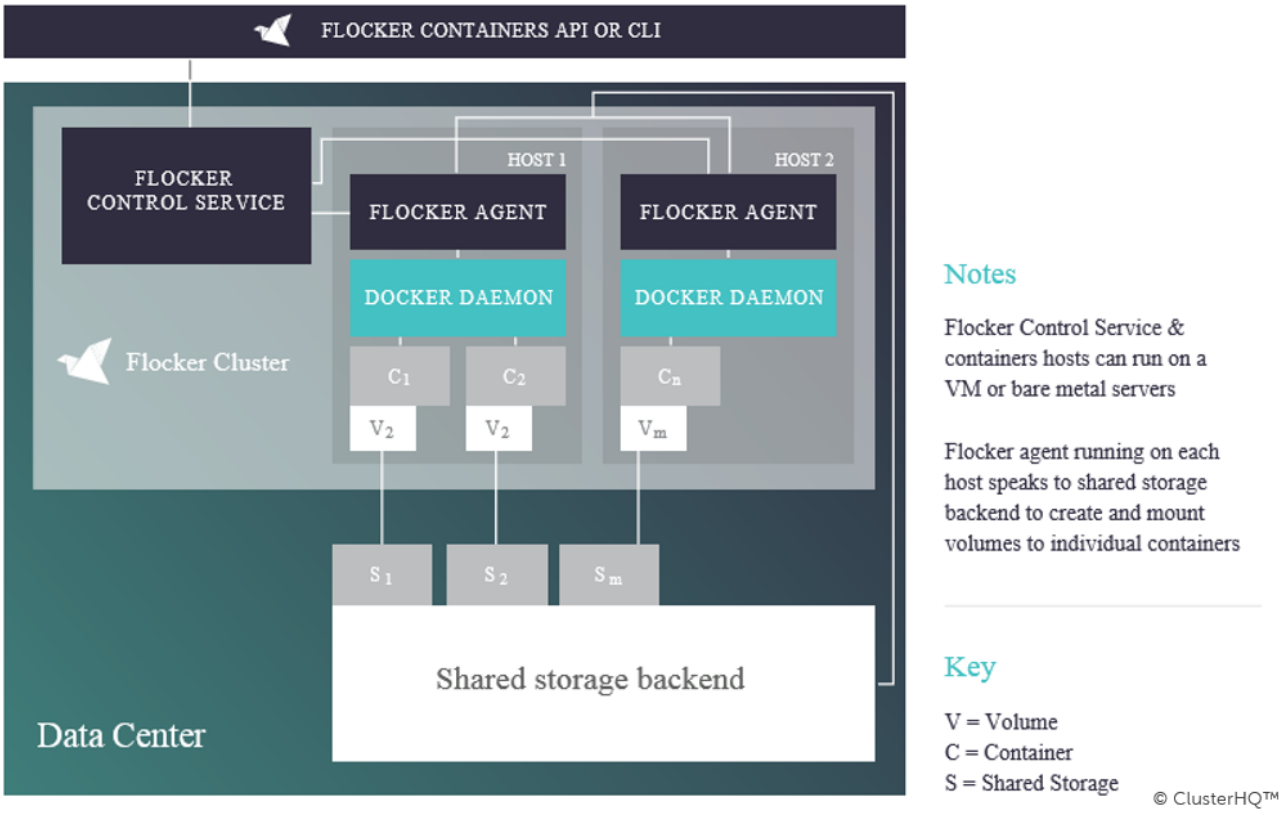

Figure 1 Flocker architecture with shared storage backend

2 Flocker

This section discusses the installation of Flocker, Docker and the various other services, processes and dependencies that are required.

2.1 Installation

Due to the rapid development and maturing cycle of the Flocker project, only the URL to the Flocker installation instructions is included. Some degree of system engineering and administration experience is required to deploy Flocker and Docker across one or multiple hosts.

https://docs.clusterhq.com/en/1.5.0/install/index.html

The configuration of Flocker is included below.

https://docs.clusterhq.com/en/1.5.0/config/index.html

2.2 Deployment

The Flocker cluster built for this concept deployment is shown below. It uses one virtual machine running the Flocker CLI and two physical hosts operating as Flocker agent nodes. The Flocker Control Service is also configured to operate on one of the two Flocker agent nodes.

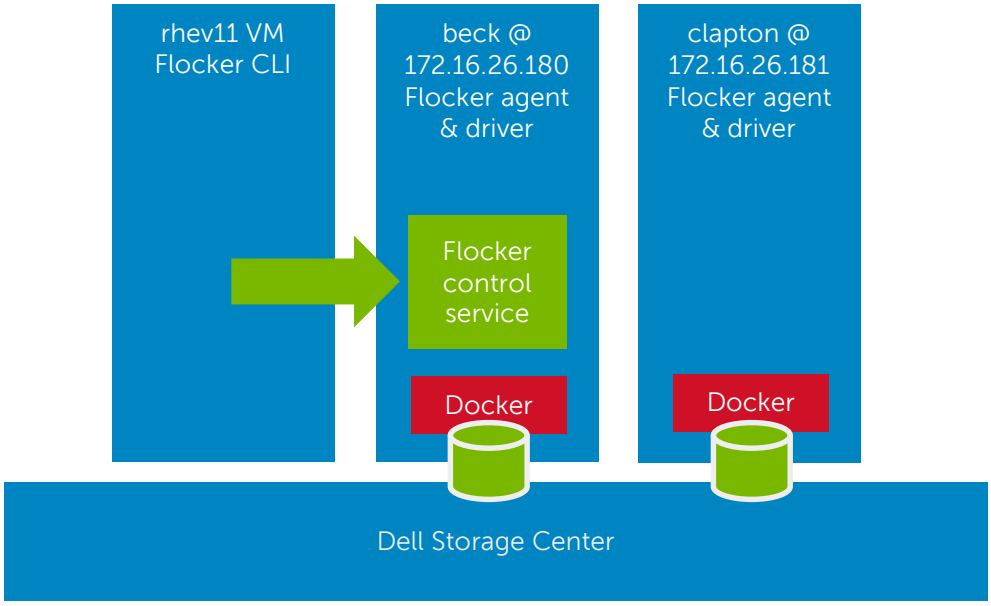

Figure 2 Flocker cluster used in testing

2.3 Processes

The Flocker CLI host does not have any dependent running processes or daemons. This host only contains the Flocker CLI package and commands.

The Flocker primary node contains Docker, the Flocker plugin for Docker, Flocker node services and Flocker control service. Note the existence of the `flocker-control` process on the primary node. The process tree when configured is shown below.

```
root@beck:~# ps -ef | egrep -e 'docker|flocker'
root      2984     1  0 Oct27 ?        00:06:01 /usr/bin/docker daemon
root      6790     1 12 Oct27 ?        02:10:46 /opt/flocker/bin/python
/usr/sbin/flocker-dataset-agent --logfile=/var/log/flocker/flocker-dataset-
agent.log
root      6800     1 16 Oct27 ?        02:53:33 /opt/flocker/bin/python
/usr/sbin/flocker-container-agent --logfile=/var/log/flocker/flocker-container-
agent.log
root      6988     1 11 Oct27 ?        02:04:19 /opt/flocker/bin/python
/usr/sbin/flocker-control -p tcp:4523 -a tcp:4524 --
logfile=/var/log/flocker/flocker-control.log
root     13795  2984  0 09:24 ?        00:00:00 docker-proxy -proto tcp -host-ip
0.0.0.0 -host-port 3306 -container-ip 172.17.0.4 -container-port 3306
root     16441     1  0 10:11 ?        00:00:00 /opt/flocker/bin/python
/usr/sbin/flocker-docker-plugin --logfile=/var/log/flocker/flocker-docker-
plugin.log
```

The Flocker secondary node contains Docker, the Flocker plugin for Docker and the Flocker node service only. The configured process tree is shown below.

```
root@clapton:~# ps -ef | egrep -e 'docker|flocker'
root      9551     1 81 10:19 ?        00:00:00 /opt/flocker/bin/python
/usr/sbin/flocker-docker-plugin --logfile=/var/log/flocker/flocker-docker-
plugin.log
root     19972     1  0 Oct27 ?        00:03:40 /usr/bin/docker daemon
root     21980     1 42 Oct27 ?        07:39:59 /opt/flocker/bin/python
/usr/sbin/flocker-dataset-agent --logfile=/var/log/flocker/flocker-dataset-
agent.log
root     21990     1 48 Oct27 ?        08:42:51 /opt/flocker/bin/python
/usr/sbin/flocker-container-agent --logfile=/var/log/flocker/flocker-container-
agent.log
```

2.4 Components

Flocker is configured using a set of YAML-based files on the Flocker CLI (deployment) node as well as on the agent nodes. It is recommended to contain these YAML-based files in the /etc/flocker directory on all hosts for ease of management. Note the idiosyncrasies of editing YAML-based files and its preference for spaces only instead of tab-based character and line delimiting.

In addition, update the /etc/hosts file on all of the hosts to reflect IP address and resolvable node names for all hosts in the Flocker cluster.

The Flocker CLI node /etc/flocker directory contains the following files.

```
root@rhev11:/etc/flocker# ls -1
cluster.crt
myflocker_app_mysql.yml
myflocker_deploy_mysql_alt.yml
myflocker_deploy_mysql_pri.yml
myflocker_deploy_stop.yml
user.crt
user.key
```

The .crt and .key files are the certificate files created in the installation and configuration URLs noted above.

The file myflocker_app_mysql.yml defines the application to be deployed in the Flocker cluster as shown.

```
root@rhev11:/etc/flocker# cat myflocker_app_mysql.yml
"version": 1
"applications":
  "mysql-local":
    "image": "mysql:latest"
    "environment":
      "MYSQL_ROOT_PASSWORD": "<your_password>"
    "ports":
    - "internal": 3306
      "external": 3306
    "volume":
      "mountpoint": "/var/lib/mysql"
```

The file myflocker_deploy_mysql_pri.yml defines the application deployment schema (for example, the host where the application should be run). The deployment schedule shown defines the mysql-local application to deploy on the host defined by IP address 172.16.26.180.

```
root@rhev11:/etc/flocker# cat myflocker_deploy_mysql_pri.yml
"version": 1
"nodes":
  "172.16.26.180": ["mysql-local"]
  "172.16.26.181": []
```

The Flocker agent nodes contain the following files in the /etc/flocker directory.

```
root@beck:/etc/flocker# ls -la
total 48
drwx------  2 root root 4096 Oct 27 16:18 .
drwxr-xr-x 92 root root 4096 Oct 28 09:24 ..
-rw-r--r--  1 root root  158 Oct 27 13:12 agent.yml
```

```
-rw-------  1 root root 1952 Oct  5 12:20 cluster.crt
-rw-------  1 root root 1899 Oct  5 12:20 control-service.crt
-rw-------  1 root root 3272 Oct  5 12:20 control-service.key
-rw-------  1 root root 1858 Oct  5 13:50 node.crt
-rw-------  1 root root 3272 Oct  5 13:50 node.key
```

The .crt and .key files are the certificate files as created in the installation and configuration URLs noted above.

The agent.yml file defines the agent node, where the cluster control service is located as well as the data source. The other agent.yml.* files above contain various other configuration schemas that can be copied into place. In this initial example, the loopback data source is defined as shown.

```
root@beck:/etc/flocker# cat agent.yml
"version": 1
"control-service":
  "hostname": "beck.techsol.beer.town"
  "port": 4524

"dataset":
  "backend": "loopback"
  "root_path": "/root/flocker_data"
```

3 Storage with Flocker

This section discusses use of storage back ends with Flocker, namely the loopback and its use with Dell Storage.

3.1 Loopback configuration

The Flocker loopback storage interface is used primarily for testing and validation. It also provides the framework for discussion of other backend storage options.

The data source is defined in the agent.yml file, typically located in the /etc/flocker directory of each agent node. The agent.yml shown below defines where the cluster control service is located as well as the data source.

Note: Changes made to the agent.yml file and schema is only captured and applied when the Flocker daemons and container objects on the host are restarted.

```
root@beck:/etc/flocker# cat agent.yml
"version": 1
"control-service":
  "hostname": "beck.techsol.beer.town"
  "port": 4524

"dataset":
  "backend": "loopback"
  "root_path": "/root/flocker_data"
```

The loopback data source and its root_path variable, define the folder on the host where a pseudo-block device is created and loop mounted where the application container is actively run.

```
root@beck:/etc/flocker# ls -laR /root/flocker_data/
/root/flocker_data/:
total 16
drwxr-xr-x 4 root root 4096 Oct 27 16:17 .
drwx------ 5 root root 4096 Oct 28 09:01 ..
drwxr-xr-x 3 root root 4096 Oct 27 16:23 attached
drwxr-xr-x 2 root root 4096 Oct 28 09:24 unattached

/root/flocker_data/attached:
total 12
drwxr-xr-x 3 root root 4096 Oct 27 16:23 .
drwxr-xr-x 4 root root 4096 Oct 27 16:17 ..
drwxr-xr-x 2 root root 4096 Oct 28 09:24 adb68fd8-8215-41ae-800c-2d74c824296b

/root/flocker_data/attached/adb68fd8-8215-41ae-800c-2d74c824296b:
total 364608
drwxr-xr-x 2 root root         4096 Oct 28 09:24 .
```

```
drwxr-xr-x 3 root root         4096 Oct 27 16:23 ..
-rw-r--r-- 1 root root 107374182400 Oct 28 09:24 block-f421b21f-a716-41a3-83f9-
cfde3f9b2554_107374182400

/root/flocker_data/unattached:
total 8
drwxr-xr-x 2 root root 4096 Oct 28 09:24 .
drwxr-xr-x 4 root root 4096 Oct 27 16:17 ..
```

df -k output:

```
root@beck:/etc/flocker# df -k
Filesystem                1K-blocks    Used Available Use% Mounted on
/dev/mapper/beck--vg-root   3877420 3300212    360532  91% /
none                              4       0         4   0% /sys/fs/cgroup
udev                        49452028       4  49452024   1% /dev
tmpfs                        9892636     984   9891652   1% /run
none                            5120       0      5120   0% /run/lock
none                        49463164      60  49463104   1% /run/shm
none                          102400       0    102400   0% /run/user
/dev/sda1                     240972   70583    157948  31% /boot
/dev/sdb                   103081248  437892  97384092   1% /var/lib/docker
/dev/loop0                 103081248  275936  97546048   1% /flocker/f421b21f-
a716-41a3-83f9-cfde3f9b2554
```

3.2 Using Flocker with SC Series array native driver

The SC Series array and Storage Center OS integrate Flocker using a native driver. The Dell driver is obtained and installed using the methods described below.

```
# git clone https://github.com/dellstorage/storagecenter-flocker-driver
# cd ./storagecenter-flocker-driver
# /opt/flocker/bin/python setup.py install
```

The driver currently supports use with iSCSI-based single or multi-pathed volumes. The Flocker agent hosts need to be setup as iSCSI initiators and logged into the intended SC Series array with appropriate iSCSI-based server objects created. The installation of the open-iSCSI package onto Ubuntu™ and its configuration is not covered within the scope of this document. However, a URL discussing Ubuntu and iSCSI configuration is included in Appendix B.2.

3.2.1 Configuring Flocker for SC Series array native driver

Storage Center is configured in Flocker using the same agent.yml file discussed in Section 3.1. However, the dataset clause in the schema is changed to point Flocker to a Dell Enterprise Storage Manager installation that proxies the request to the configured SC Series array.

Note: Changes made to the agent.yml file and schema is only captured and applied upon the restart of Flocker daemons and container objects on the host.

The modified agent.yml (showing just the dataset clause) file:

```
root@beck:/etc/flocker# cat agent.yml.native; cp agent.yml.native agent.yml
[snip]
dataset:
        backend: "dell_storagecenter_driver"
        storage_host: "172.16.21.161"        # IP addr of Dell EM installation
        storage_port: "3033"                 # default port for EM
        dell_sc_ssn: "716"                   # Serial Number of SC array
        username: "admin"                    # Login credentials for EM
        password: "<password>"               # Login password for EM
        volume_folder_name: "Unix/Linux/OpenStack/flocker"
        server_folder_name: "Unix/Linux/OpenStack/flocker"
```

3.3 Using Flocker with OpenStack Cinder

Dell SC Series arrays integrate with Flocker using the Dell Storage Center OpenStack Cinder driver interface and API.

The installation of OpenStack is outside the scope of this document. The Dell Storage Center Cinder driver is available in OpenStack Kilo and newer OpenStack releases.

3.3.1 Configuring Flocker for OpenStack Cinder

Storage Center is configured in Flocker using the same agent.yml file discussed in Section 3.1. However, the dataset clause in the schema is changed to point Flocker to an OpenStack/DevStack installation that is configured to a Dell Enterprise Storage Manager installation; it proxies the request to the configured SC Series array.

Note: Changes made to the agent.yml file and schema are only captured and applied when the Flocker daemons and container objects on the host are restarted.

The modified agent.yml (showing just the dataset clause) file:

```
root@beck:/etc/flocker# cat agent.yml.openstack; cp agent.yml.openstack
agent.yml
[snip]
dataset:
        backend: "openstack"
        region: “MN”
        verify_peer: “false”
        auth_plugin: “password"
        username: “admin”
```

```
password: "<password>"
auth_url: "https://<IP addr or FQDN of OpenStack/DevStack installation"
```

4 Using Flocker

This section discusses using Flocker to deploy a MySQL™-based container across the cluster, viewing the container status and validating data volume persistence and portability.

4.1 Deploying MySQL

The following command run on the Flocker CLI node initiates the download of a MySQL image, and spawns a container running MySQL on the host defined at IP address 172.16.26.180. The agent.yml file on each agent node has also been modified to use the Storage Center data source instead of the loopback block storage.

```
root@rhev11:/etc/flocker# flocker-deploy --cacert $PWD/cluster.crt --cert
$PWD/user.crt --key $PWD/user.key 172.16.26.180 myflocker_deploy_mysql_pri.yml
myflocker_app_mysql.yml
The cluster configuration has been updated. It may take a short while for
changes to take effect, in particular if Docker images need to be pulled.
```

This initiated a MySQL container to be spawned and run on the host defined at IP address 172.16.26.180.

4.2 Validating MySQL and data volume persistence and portability

This section discusses the validation of the MySQL container, its locality and its data persistence and portability.

4.2.1 Flocker transparency

Flocker obscures the location of the running container applications. With the container application running on either node(s) of a Flocker cluster, the application can be accessed using either IP address of the nodes as shown.

```
# mysql –h172.16.26.180 –uroot –p<your_password>
```

Optionally, the command line below will provide access to the running MySQL container, regardless of where the container is actively running.

```
# mysql –h172.16.26.181 –uroot –p<your_password>
```

The container status is obtained using the command line of either node with the command:

```
# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                    NAMES
c061bc8dc5f1        mysql:latest        "/entrypoint.sh mysql"   4 hours ago
Up 4 hours          0.0.0.0:3306->3306/tcp   flocker--mysql-local
```

4.2.2 SC Series volume

The SC Series volume is created and mapped to the Flocker agent node defined at IP address 172.16.26.180 via iSCSI (Figure 3).

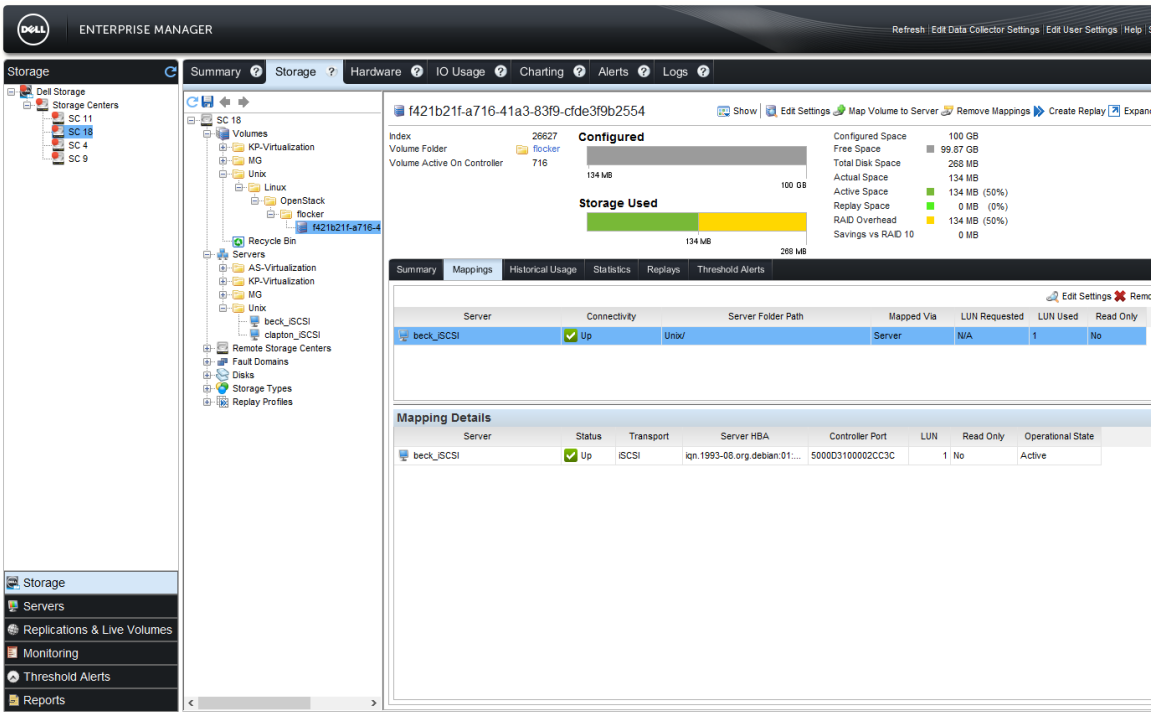

Figure 3 SC Series volume mapped to Flocker

4.2.3 Moving MySQL and the SC Series volume

The Flocker deployment YAML file is run again, this time deploying the MySQL container to the alternate node.

```
root@rhev11:/etc/flocker# flocker-deploy --cacert $PWD/cluster.crt --cert
$PWD/user.crt --key $PWD/user.key 172.16.26.180 myflocker_deploy_mysql_alt.yml
myflocker_app_mysql.yml
The cluster configuration has been updated. It may take a short while for
changes to take effect, in particular if Docker images need to be pulled.
```

The SC Series volume is shown below, where it is now mapped to the Flocker agent node defined at IP address 172.16.26.181 via iSCSI.

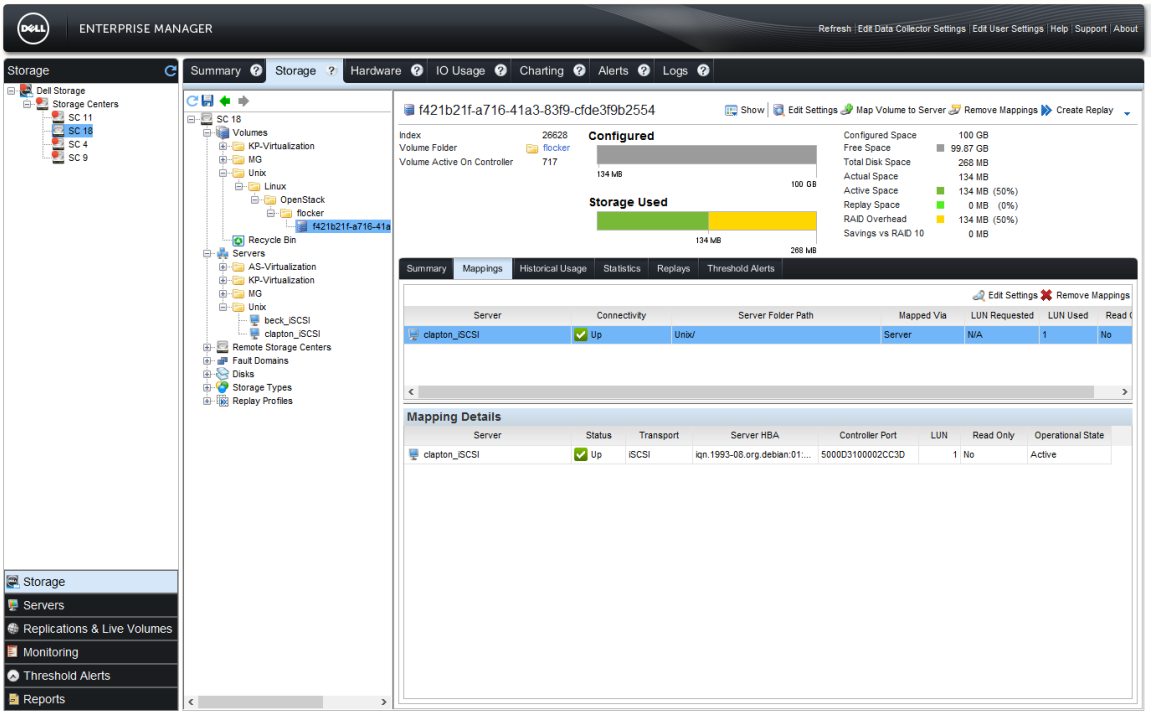

Figure 4 MySQL container mapped to the alternate node

4.3 Stopping MySQL

Since Flocker manages the Docker containers, Flocker should also be used to terminate the containers in a discrete and clean manner without compromising data integrity. If Flocker-managed Docker containers are stopped or killed with native Docker commands, these containers are automatically re-spawned by Flocker daemons.

A Flocker deployment file located in /etc/flocker on the Flocker CLI node is created as shown, defining that neither agent node should run any applications.

```
root@rhev11:/etc/flocker# cat myflocker_deploy_stop.yml
"version": 1
"nodes":
  "172.16.26.180": []
  "172.16.26.181": []
```

This deployment file is initiated to terminate any existing MySQL containers running on either agent node.

```
root@rhev11:/etc/flocker# flocker-deploy --cacert $PWD/cluster.crt --cert
$PWD/user.crt --key $PWD/user.key 172.16.26.180 myflocker_deploy_stop.yml
myflocker_app_mysql.yml
The cluster configuration has been updated. It may take a short while for
changes to take effect, in particular if Docker images need to be pulled.
```

A Configuration details

Table 1 Component table

Component	Description
OS	Ubuntu 14.04.* LTS, Flocker 1.5 & Docker 1.8.3
Driver version	NA
Firmware version	NA
Switch	NA
Cabling	Fiber Channel
Server	Dell PowerEdge R620
Storage	Dell Storage Center OS (SCOS) 6.5.30 @ Virtual port mode
Enterprise Manager	2015 R1

B Additional resources

B.1 Technical support and resources

For Copilot support of Dell SC Series products:

- Global online support
- Email: support@compellent.com (non-emergency business hours)
- Phone: 866-EZ-STORE (866-397-8673) (United States only)

The Dell SC Series Portal is an online portal for existing customers. A valid portal account is required to access the Knowledge Center. Once login to the portal, go to "Knowledge center".

Dell TechCenter is an online technical community for IT professionals and is a great resource to discover and learn about a wide range of technologies such as storage, servers, networking, software, and cloud management.

B.2 Related documentation

Table 2 lists the referenced or recommended resources related to this document.

Table 2 Referenced or recommended resources

Vendor	Resource
Flocker	https://docs.clusterhq.com/en/1.5.0/introduction/index.html
Docker	https://docs.docker.com/installation/ubuntulinux
OpenStack	https://www.openstack.org/software/kilo
OpenStack	http://docs.openstack.org/kilo/config-reference/content/dell-storagecenter-driver.html
Ubuntu iSCSI	http://www.server-world.info/en/note?os=Ubuntu_14.04&p=iscsi&f=2