Reference Architecture: Running HA databases using Docker UCP with Dell Storage SC Series and Flocker

Daniel Tan / Dell Storage Applications Engineering
Ryan Wallner / ClusterHQ Solutions Architect
### Revisions

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 2016</td>
<td>1.0</td>
<td>Initial release</td>
<td>Daniel Tan</td>
</tr>
</tbody>
</table>

*Page 2 / ClusterHQ Flocker with Dell Storage SC Series and Docker UCP*
Table of contents

Revisions
Executive summary
Overview
Audience
Purpose
Solution overview
  Flocker by ClusterHQ
  Dell SC Series storage
  Docker UCP and Swarm
Solution design
  The complete stack
  Component configuration
    Flocker
    Flocker Volume Plugin for Docker
    Dell Storage
    Docker UCP/Swarm
    Servers
    Network and transport
Putting it together
Use case with Docker Compose
  nginx
  MongoDB
  Docker Compose
Conclusion
Configuration details
Additional resources
  Technical support
  Related documentation
Executive summary

Docker® containers are architected mainly around the concept of stateless compute/container resources. However, within many applications and use cases, stateful compute resources such as databases are needed. Docker does not natively provide storage volume management or data persistence when porting these compute resources between hosts.

Flocker™ by ClusterHQ™ addresses these needs and enables the containers to be used for stateful applications, like databases, by providing a framework for volume management and data persistence when moving compute resources from one host to another. While Flocker works with all major container managers (including Docker Swarm, Kubernetes, and Apache Mesos), this paper focuses on using Docker Swarm, Flocker and Dell SC Series storage to run and migrate a Dockerized database.

Overview

This paper introduces and presents the architecture, use and configuration of Flocker in conjunction with Dell™ SC Series arrays and Docker UCP/Swarm. This environment achieves unified container management and orchestration with storage volume management and data persistence.

Audience

It is assumed that readers are DevOps professionals looking at microservice-based architectures to provide stateful workloads in modern enterprise environments. They should be familiar with the paradigm and characteristics of containers and container management and resource abstraction, virtualization, centralized storage management principles, and have a basic Linux/UX understanding.

Purpose

This paper presents the architectural guidelines, software configurations and versioning required to run a proven, flexible and reliable microservices framework with Flocker, Dell arrays and Docker services. A brief overview of this paper is:

- Overview of Flocker by ClusterHQ, SC Series arrays and Docker UCP/Swarm
- Benefits of each component
- Design and deployment of the solution stack
- Discussion of workloads and use cases
- Conclusion
Solution overview

This section presents each layer of the solution stack, discusses layer principles and purpose, details the logical components and how each layer interacts and functions. This solution overview is a starting point of consideration and should be objectively evaluated according to the needs of the respective business environment.

Flocker by ClusterHQ

Flocker is an open-source container data volume manager for Dockerized applications. By providing tools for data migrations, Flocker gives operation teams the tools needed to run containerized stateful services, like databases, in production.

Unlike a Docker data volume that is tied to a single server, a Flocker data volume (called a dataset) is portable and can be used with any container in a cluster.

Flocker manages Docker containers and data volumes together. When Flocker is used to manage a stateful microservice, the volumes will follow the containers when they move between different hosts in the cluster. Flocker is designed to work with other tools in order to build and run distributed applications. Flocker can be used with popular container managers or orchestration tools like Docker Swarm, Kubernetes, and Mesos. The Flocker framework (Figure 1) is optimized to work with ClusterHQ Volume Hub™. The Volume Hub makes managing the container data easier to understand and visualize by displaying a full catalog of the application data volumes and their statuses. Data volumes that are experiencing problems are automatically displayed so that you can easily track clusters and resolve issues quickly.
THE BENEFITS OF FLOCKER INCLUDE

- Lightweight framework designed to work with existing toolsets and facilitate building and running distributed applications
- Container and data mobility, treating the container and its data as a single atomic unit
- Works with Docker, Kubernetes or Mesos
- Seamless workflow integration with Docker UCP
- Supports multiple block-based storage backends including SC Series storage, OpenStack Cinder or Amazon EBS

Figure 1  Flocker logical deployment
Dell SC Series storage

SC Series storage represents the cornerstone of the unified Dell Storage portfolio and price/performance leader in block-based intelligent SAN storage for both flash and spinning media. An SC8000 array is used in this solution to provision iSCSI-based storage volumes to the Flocker™ nodes and is managed with the Dell Storage Manager (DSM) unified management interface shown below.

THE BENEFITS OF SC SERIES INCLUDE:

- Data storage and placement is adaptive and virtualized, with no degradation of performance as data ages and expands
- Storage costs are reduced, at any capacity
- Storage management is unified across the storage portfolio using DSM

These advantages are evidence that SC Series technology provides a smart and efficient environment.
Docker UCP and Swarm

Docker UCP and ancillary Docker services including Swarm, Compose, Engine and Machine support this solution by providing the virtualization and resource abstraction layer to spawn, host and manage the compute container resources. Docker UCP enables the management and orchestration of your container-based resources through a unified single pane of glass as shown below.

THE BENEFITS OF DOCKER UCP AND SWARM AND COMPOSE INCLUDE:

- Agility, flexibility and freedom to fast-deploy environments and applications in IT operations, thus enabling the ability to fluidly adapt to change
- Control the solution stack from infrastructure to applications to achieve standardization, security and scale
- Portability and abstraction to build, ship and run applications anywhere without concern of hardware-based resource dependencies
Solution design

The solution design is intended to demonstrate a complete and operational containerized application stack running on bare metal including the use of Docker UCP/Swarm as the container management and orchestration layer.

This solution design is a starting point and may be used in both operational deployments or as a proof of concept.

The complete stack

As has often been said, a picture is worth a thousand words. The complete solution stack is represented in Figure 4 and details for the individual component configurations are in section 3.2.

Figure 4- The Complete Stack
Component configuration

**FLOCKER**

Flocker provides storage volume management and data persistence while maintaining container portability of Docker-based container resources.

Flocker 1.12.0 was used in this solution.

```
$ beck:/etc/flocker$ /opt/flocker/bin/flocker-volume --version
```
Flocker installation instructions are available at https://docs.clusterhq.com/en/latest/docker-integration/

The components of Flocker include:

- **Flocker control CLI (provides flocker-* CLI command set)**
  - The `flockerctl` command can also be used to manage the Flocker environment and is available at https://docs.clusterhq.com/en/latest/flocker-features/flockerctl.html

- **Flocker control node (installed on a standalone host)**
  - Processes: flocker-control-agent (**mandatory**)
  - Purpose: One of these must run in a cluster on a control node; typically in a separate control node from other application nodes.

- **Flocker node agent (installed on all Flocker nodes hosting applications)**
  - Processes: flocker-dataset-agent (**mandatory**)
  - Purpose: Must run on every node that wishes to access Hedvig storage for containers
  - Processes: flocker-container-agent (optional)
  - Purpose: Only needs to run if you are using Flocker Container API; in most cases including this architecture, it is not needed. Running the node agent does not degrade system performance in any way.

- **Flocker docker plugin**
  - Processes: flocker-docker-plugin (optional)
  - Purpose: Must run on every node that wishes to use the Docker API/CLI to provision and use storage for Docker containers.

- **Flocker storage backend configuration, application and deployment configurations**

**FLOCKER VOLUME PLUGIN FOR DOCKER**

Docker/UCP can use Flocker services by utilizing the Flocker Volume Plugin for Docker. This plugin runs as an agent on the application nodes and handles requests for volumes from the Docker API.
The Flocker Volume Plugin can also be invoked by applying the `--volume-driver` option to the `docker run` command, or via the `-d` flag to the `docker volume` command.

The driver can also be used through the Docker Remote API with the GET, POST or DELETE directives to volumes from the remote engine API.

PORTS

The following ports are required for function, and cannot be blocked on your Docker UCP control node:

12385, 12386, 443, 2376, 12376, 12379, 12380, 12382, 12383, 12384, 12381

The following ports are required for function, and cannot be blocked on your Flocker control node:

4523, 4524

The following ports are required for function, and cannot be blocked on your Flocker agent node:

4524

FLOCKER CONTROL MODE

The Flocker control node should run as a standalone server or control host so that it’s CPU and RAM resources can operate in dedicated fashion. Since the control service manages all nodes and provides an API into the Flocker cluster it is recommended that you run it in this manner.
The Control Service node recommended minimum requirements is two CPUs and 500MB of RAM for every 100 agent nodes run.

**TLS**

Flocker uses TLS certificates for intra-node communication. Each node must have a Flocker node certificate, and a cluster certificate. In this architecture, nodes must also have a plugin certificate for the Docker plugin. This certificate can be shared on all nodes.
Certificates can be created by using the flocker-ca tool or flocker-openssl.

Flocker is deployed on a VM and across two Dell PowerEdge R620 servers as shown below.

```bash
flocker_vm:/etc/flocker$ ps -ef | grep flocker
root  1409  1 1 May20 ?   01:37:18 /opt/flocker/bin/python
/usr/sbin/flocker-control -p tcp:4523 -a tcp:4524
--logfile=/var/log/flocker/flocker-control.log

beck:/etc/flocker$ ps -ef | grep flocker
root  1437  1 0 May20 ?   00:04:37 /opt/flocker/bin/python
/usr/sbin/flocker-dataset-agent --logfile=/var/log/flocker/flocker-dataset-agent.log
root  1784  1 1 May20 ?   02:35:08 /opt/flocker/bin/python
/usr/sbin/flocker-container-agent --logfile=/var/log/flocker/flocker-container-agent.log
root 26371  1 0 May25 ?   00:02:36 /opt/flocker/bin/python
/usr/sbin/flocker-docker-plugin --logfile=/var/log/flocker/flocker-docker-plugin.log

clapton:/etc/flocker$ ps -ef | grep flocker
root  1427  1 0 May20 ?   00:03:40 /opt/flocker/bin/python
/usr/sbin/flocker-dataset-agent --logfile=/var/log/flocker/flocker-dataset-agent.log
root  1773  1 1 May20 ?   01:35:41 /opt/flocker/bin/python
/usr/sbin/flocker-container-agent --logfile=/var/log/flocker/flocker-container-agent.log
root  9745  1 3 May25 ?   00:42:30 /opt/flocker/bin/python
/usr/sbin/flocker-docker-plugin --logfile=/var/log/flocker/flocker-docker-plugin.log
```

**DELL STORAGE**

Dell provides a Flocker-compliant driver that enables the use of SC Series iSCSI-based block volumes to be used with Flocker and bound to Docker-based containers. The installation of this driver is detailed below and available at [https://github.com/dellstorage/storagecenter-flocker-driver](https://github.com/dellstorage/storagecenter-flocker-driver).

```bash
# sudo apt-get update && apt-get –y install open-iscsi
# cd /etc/flocker
# sudo git clone https://github.com/dellstorage/storagecenter-flocker-driver
# cd /storagecenter-flocker-driver
# sudo /opt/flocker/bin/python setup.py install
```

Once the installation is complete, Dell Storage is configured as a Flocker storage backend using the procedure detailed below.

The `/etc/flocker/agent.yml` file is YAML-based and should only contain spaces (no tab values). Additionally, this file needs to reside in the `/etc/flocker` directory and exist on all Flocker node agent hosts. The agent.yml file should be configured as shown below. A restart of the flocker-dataset-agent, flocker-container-agent and flocker-docker-plugin processes is recommended after making any changes to this file.
beck:/etc/flocker$ cat ./agent.yml
"version": 1
"control-service":
  "hostname": "beck" # name/IP of Flocker control node
  "port": 4524 # default Flocker port: 4524

"dataset":
  "backend": "dell_storagecenter_driver" # backend definition for Dell SC
  "storage host": "172.16.21.161" # IP address of Dell DSM instance
  "dell_sc_ssn": "716" # Dell SC Controller Index
  "username": "admin" # Login for Dell DSM instance
  "password": "password" # Password for Dell DSM instance
  "volume folder name": "Unix/Linux/flocker" # Volume object path
  "server folder name": "Unix/Linux/flocker" # Server object path

DOCKER UCP/SWARM

Docker UCP/Swarm provides a unified client-facing single pane of glass to construct application definitions, manage containers and deploy such applications across the Swarm-defined container-based cluster. Docker CS engine 1.10.3-cs3 is used for this solution.

beck:/etc/flocker$ sudo docker info | grep Version
Server Version: 1.10.3-cs3
Kernel Version: 3.19.0-59-generic

Docker UCP/Swarm installation instructions is available at https://docs.docker.com/ucp/installation/install-production/

THE COMPONENTS OF DOCKER UCP/SWARM INCLUDE:

- Docker UCP (provides a unified interface to manage applications, containers, images, deployments and more)
- Docker Swarm (presents multiple Docker-based containers as a single unified compute resource)
- Docker Compose (framework to define applications as a logical construct of containers)
- Docker Engine (enables the resource abstraction layer where Docker-based containers are hosted)
- Docker Machine (automated toolset to facilitate deployment of containers onto a variety of target hosts including bare metal, hypervisors and more)
Docker UCP/Swarm consists of a microservice-based grouping of multiple running containers, where each container runs a docker/ucp-based image and service. The list of docker/ucp-based images required to operate the Docker UCP/Swarm platform is shown below. The host named `beck` serves a dual purpose as both Docker UCP controller as well as a host to running container instances.

```bash
beck:/etc/flocker$ sudo docker ps -a
CONTAINER ID    IMAGE                   COMMAND                  CREATED          STATUS          PORTS               NAMES
6c03460cc6f0    docker/ucp-controller:1.1.0   /bin/controller serv   18 hours ago   Up 18 hours     0.0.0.0:443->8080/tcp   ucp-controller
7ce375f9ba1     docker/ucp-auth:1.1.0         /usr/local/bin/enzi    18 hours ago   Up 18 hours     0.0.0.0:12386->4443/tcp   ucp-auth-worker
46534906ba05    docker/ucp-auth:1.1.0         /usr/local/bin/enzi    18 hours ago   Up 18 hours     0.0.0.0:12385->4443/tcp   ucp-auth-api
584f5440b3b2    docker/ucp-auth-store:1.1.0  /usr/local/bin/rethi   18 hours ago   Up 18 hours     0.0.0.0:12383-12384->12383-12384/tcp ucp-auth-store
3811f7d06ce     docker/ucp-cfssl:1.1.0       /bin/cfssl serve -ad  18 hours ago   Up 18 hours     8888/tcp, 0.0.0.0:12381->12381/tcp   ucp-cluster-root-ca
8433a153860b     docker/ucp-swarm:1.1.0       /swarm manage --tlsv  18 hours ago   Up 18 hours     0.0.0.0:2376->2375/tcp   ucp-swarm-manager
190a3b638a86     docker/ucp-swarm:1.1.0       /swarm join --discov  18 hours ago   Up 18 hours     2375/tcp                      ucp-swarm-join
85170a97c7f5b     docker/ucp-proxy:1.1.0     /bin/run               18 hours ago   Up 18 hours     0.0.0.0:12376->2376/tcp    ucp-proxy
7796af25ba4      docker/ucp-etcd:1.1.0        /bin/etcd --data-dir  18 hours ago   Up 18 hours     2380/tcp, 4001/tcp, 7001/tcp, 0.0.0.0:12380->12380/tcp, 0.0.0.0:12379->2379/tcp   ucp-join
```

```bash
clapton:/etc/flocker$ sudo docker ps -a
CONTAINER ID    IMAGE                   COMMAND                  CREATED          STATUS          PORTS               NAMES
4d1f56174dc3    docker/ucp-swarm:1.1.0   /swarm join --discov    18 hours ago   Up 18 hours     2375/tcp                      ucp-swarm-join
85255c50cc358   docker/ucp-proxy:1.1.0   /bin/run                18 hours ago   Up 18 hours     0.0.0.0:12376->2376/tcp    ucp-proxy
```
The Swarm cluster and status are shown below representing a single Swarm master (beck) and two Swarm nodes (beck and clapton).

```
beck:/etc/flocker$ sudo docker --tlsverify -H beck:2376 info
Containers: 31
  Running: 12
  Paused: 0
  Stopped: 19
Images: 30
Server Version: swarm/1.1.3
Role: primary
Strategy: spread
Filters: health, port, dependency, affinity, constraint
Nodes: 2
  beck: 172.16.26.180:12376
    Status: Healthy
    Containers: 22
    Reserved CPUs: 0 / 33
    Reserved Memory: 0 B / 99.09 GiB
    Labels: executiondriver=native-0.2, kernelversion=3.19.0-59-generic, operating_system=Ubuntu 14.04.3 LTS, storagedriver=aufs
    Error: (none)
    UpdatedAt: 2016-06-02T19:29:38Z
  clapton: 172.16.26.181:12376
    Status: Healthy
    Containers: 9
    Reserved CPUs: 0 / 33
    Reserved Memory: 0 B / 99.06 GiB
    Labels: executiondriver=native-0.2, kernelversion=3.19.0-59-generic, operating_system=Ubuntu 14.04.3 LTS, storagedriver=aufs
    Error: (none)
    UpdatedAt: 2016-06-02T19:29:51Z
Plugins:
  Volume:
  Network:
Kernel Version: 3.19.0-59-generic
Operating System: linux
Architecture: amd64
CPUs: 66
Total Memory: 198.1 GiB
Name: 8433a153860b
```
SERVERS

Dell PowerEdge R620 servers were used in this solution. Each Dell PowerEdge server is equipped with dual Xeon 2.9Ghz 8-way processors, 96GB RAM, showing BIOS firmware on each server as 2.5.2, and a dual-port dual-VLANed QLogic 2562 HBA. Each server is equipped with an Ubuntu 14.04.03 LTS installation, while a VM runs Ubuntu 15.10 that hosts both the Flocker control CLI and Flocker control node installation.

NETWORK AND TRANSPORT

Each Dell PowerEdge server is equipped with a single Ethernet connection (em1) for primary networking and a second Ethernet connection (em2) for isolated iSCSI-based data traffic as shown. The docker0 interface is a virtual network bridge that provides network connectivity to hosted Docker-based containers.

```
$ ifconfig -a

docker0   Link encap:Ethernet  HWaddr 02:42:a4:68:db:16
           inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
           inet6 addr: fe80::42:a4ff:fe68:db16/64 Scope:Link
           UP BROADCAST MULTICAST  MTU:1500  Metric:1
           RX packets:3 errors:0 dropped:0 overruns:0 frame:0
           TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:216 (216.0 B) TX bytes:258 (258.0 B)

em1       Link encap:Ethernet  HWaddr bc:30:5b:ee:67:34
           inet6 addr: fe80::be30:5bff:feee:6734/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:8800805 errors:0 dropped:1758403 overruns:0 frame:0
           TX packets:587087 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:948361485 (948.3 MB) TX bytes:266985373 (266.9 MB)
           Memory:d8b00000-d8bfffff

em2       Link encap:Ethernet  HWaddr bc:30:5b:ee:67:35
           inet addr:10.10.83.50  Bcast:10.10.255.255  Mask:255.255.0.0
           inet6 addr: fe80::be30:5bff:feee:6735/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:137841 errors:0 dropped:910 overruns:0 frame:0
           TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:9394863 (9.3 MB) TX bytes:2692 (2.6 KB)
           Memory:d8c00000-d8cfffff
```
Putting it together

A sample containerized application stack is assembled and deployed, using the Docker Compose use case in section 4. The use case demonstrates the `docker-up` and `flockerctl` commands showing proper initialization and deployment of the environment, containers and mapped volumes from an SC Series array.

Create the SC Series volume using the Flocker driver.

```bash
# sudo docker volume create -d flocker --name flockersc_01 -o size=5G
flockersc_01
```

List all of the volumes visible to Docker.

```bash
# sudo docker volume ls
DRIVER   VOLUME       NAME
flocker   flockersc_01
flocker   flockersc_01
local     beck/3fbe95327e137a14ac90eb629baac22a229e02f2431aba00d02705f3070c791a
```

Deploy the nginx/mongodb application using the predefined `docker-compose.yml` file. The application prefix `volplugindemo` is inherited from the path of the present working directory `/volplugindemo` where the `docker-compose.yml` file resides.

```bash
# sudo docker-compose -f docker-compose.yml up -d
Creating network "volplugindemo_default" with the default driver
Pulling mongodb (clusterhq/mongodb:latest)...
clapton: Pulling clusterhq/mongodb:latest... : downloaded
beck: Pulling clusterhq/mongodb:latest... : downloaded
clapton: Pulling nginx:latest... : downloaded
beck: Pulling nginx:latest... : downloaded
Creating volplugindemo_mongodb_1
Creating volplugindemo_web_1
```

The running application containers are shown below.

```bash
# sudo docker-compose -f docker-compose.yml ps
Name                  Command                  State         Ports
volplugindemo_mongodb_1 /bin/sh -c /home/mongodb/m ... Up       27017/tcp
volplugindemo_web_1    nginx -g daemon off; Up       443/tcp, 172.16.26.180:80->80/tcp
```
The `volpluginsdemo_mongodb_1` container is inspected to demonstrate that the flocker volume named `flockersc_01` is mounted to the `/data` directory within the running container instance. The string shown in **RED** is also the name of the Dell SC volume on the storage array as shown.

```bash
# sudo docker inspect -f "{{.Mounts"}} volpluginsdemo_mongodb_1
{"2d7f848fb487f779f1d2eb2d65735946354b24537727812a083e98a828b9ab5b
/var/lib/docker/volumes/2d7f848fb487f779f1d2eb2d65735946354b24537727812a083e98a828b9ab5b/data
/data/log local true }

{flockersc_01 /flocker/be89bb8c-f058-4f4d-b953-a1fe53d93b21 /data flocker rw true rprivate
{803aaf7c165cc619417a5a2e5b5c7e77fbfde527d60e02fd2a6bbc7d2d2c630
/var/lib/docker/volumes/803aaf7c165cc619417a5a2e5b5c7e77fbfde527d60e02fd2a6bbc7d2d2c630/data
/data/db local true }
```

**Figure 7** Dell SC Flocker volume

This final command displays all volumes presently within Flocker scope and management.

```bash
# sudo flockerctl ls

<table>
<thead>
<tr>
<th>DATASET</th>
<th>SIZE</th>
<th>METADATA</th>
<th>STATUS</th>
<th>SERVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0f8aaff3-3f82-49af-9f81-c9f8562eab73</td>
<td>5.00G</td>
<td>maximum size=5368709120, name=flockersc_01</td>
<td>attached</td>
<td>c109798f (172.16.26.180)</td>
</tr>
</tbody>
</table>
```
Use case with Docker Compose

This section details a microservice-based application use case based on the free, open source projects nginx and MongoDB.
The nginx server is used to offer a forward-facing HTTP interface, while MongoDB is used to service any backend database requests built on the Flocker persistent data and volume management services.

nginx

nginx (pronounced “engine X”) is another free, open source project providing a high performance HTTP server, reverse proxy as well as a IMAP/POP3 proxy server.
Nginx, unlike traditional HTTP servers, relies on a much more scalable event-driven (asynchronous) architecture using smaller and more predictable memory under load, and therefore a smaller memory footprint. The architecture of nginx allows it to scale from the smallest applications up through large server cluster deployments.

MongoDB

MongoDB is a free, open source NoSQL-based database, relying upon JSON-like document structure rather than traditional table-based relational databases.
MongoDB strengths lie within its schema-less data structure, ease of scalability and its in-memory working sets enabling faster query and data access.

Docker Compose

Docker Compose is used to assemble these services into a fluid and scalable application stack based on these services served from Docker-based containers and load-balanced with Docker Swarm.
The Docker Compose YML syntax to assemble this application stack is shown below.
version: '2'
services:
  web:
    image: docker-nginx:latest
    ports:
    - "80:80"
    links:
    - "mongodb:mongodb"

  mongodb:
    image: clusterhq/mongodb:latest
    volumes:
    - testvol:/data

volumes:
  testvol:
    external:
      name: flockervol_01
Conclusion

The Flocker by ClusterHQ, Dell SC Series storage and Docker UCP stack presented in this paper achieves and delivers an elastic, scalable and complete application platform solution.

This platform solution empowers modern enterprise environments with the elegance, agility and freedom to define and fast-deploy microservice-based applications and container resources on a myriad of hardware or virtualized platforms, backed by Dell’s award-winning and best in class price/performance SC Series storage. These container resources are stateful and provide persistent data volume management enabled through Flocker agents and services.

Additionally, these container resources operate independent of hardware, and can be managed with the unified Docker UCP interface. The use of Docker Swarm to logically abstract these container resources into a single compute unit, further enables the ability to seamlessly and fluidly scale up, down or allocate these resources onto new hardware platforms without any concern of business/application impact or downtime.
### Configuration details

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Dell PowerEdge R620 12g x 2</td>
</tr>
<tr>
<td>Firmware/BIOS</td>
<td>2.5.2</td>
</tr>
<tr>
<td>CPU</td>
<td>Xeon 2.9/8 x 2</td>
</tr>
<tr>
<td>Memory</td>
<td>96GB</td>
</tr>
<tr>
<td>Storage</td>
<td>Boot from SAN, SAN data volumes</td>
</tr>
<tr>
<td>HBA</td>
<td>Qlogic 2562</td>
</tr>
<tr>
<td>HBA Firmware/BIOS</td>
<td>8.01.02/3.29</td>
</tr>
<tr>
<td><strong>Flocker by ClusterHQ</strong></td>
<td></td>
</tr>
<tr>
<td>Flocker version</td>
<td>1.12</td>
</tr>
<tr>
<td>Flocker Client/OS</td>
<td>Ubuntu 15.10 64bit</td>
</tr>
<tr>
<td>Flocker Nodes/OS</td>
<td>Ubuntu 14.04 64bit</td>
</tr>
<tr>
<td><strong>Dell Storage SC Series</strong></td>
<td></td>
</tr>
<tr>
<td>Storage Center</td>
<td>SC8000</td>
</tr>
<tr>
<td>SCOS version</td>
<td>6.7.5.x</td>
</tr>
<tr>
<td><strong>Docker</strong></td>
<td></td>
</tr>
<tr>
<td>UCP</td>
<td>1.1.0</td>
</tr>
<tr>
<td>Swarm</td>
<td>1.1.0</td>
</tr>
<tr>
<td>Engine</td>
<td>1.10.3-cs3</td>
</tr>
<tr>
<td>Networking</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>
Technical support

Dell.com/support is focused on meeting customer needs with proven services and support.

Table 1  Additional support information on specific array models

<table>
<thead>
<tr>
<th>Dell Storage</th>
<th>Online support</th>
<th>Email</th>
<th>Phone support (US only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC Series and Compellent</td>
<td><a href="https://customer.compellent.com">https://customer.compellent.com</a></td>
<td><a href="mailto:support@compellent.com">support@compellent.com</a></td>
<td>866-EZ-STORE (866-397-8673)</td>
</tr>
<tr>
<td>SCv Series</td>
<td><a href="http://www.dell.com/support">http://www.dell.com/support</a></td>
<td>Specific to service tag</td>
<td>800-945-3355</td>
</tr>
<tr>
<td>XC Series</td>
<td><a href="http://www.dell.com/support">http://www.dell.com/support</a></td>
<td>Specific to service tag</td>
<td>800-945-3355</td>
</tr>
<tr>
<td>PS Series (EqualLogic)</td>
<td><a href="http://eqlsupport.dell.com">http://eqlsupport.dell.com</a></td>
<td><a href="mailto:eqlx-customer-service@dell.com">eqlx-customer-service@dell.com</a></td>
<td>800-945-3355</td>
</tr>
</tbody>
</table>

Dell TechCenter is an online technical community where IT professionals have access to numerous resources for Dell software, hardware and services. Storage Solutions Technical Documents on Dell TechCenter provide expertise that helps to ensure customer success on Dell Storage platforms.

Related documentation

Table 2  Referenced or recommended resources

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell</td>
<td>Dell Storage Center System Manager Administrator’s Guide available on the Knowledge Center</td>
</tr>
<tr>
<td>Dell</td>
<td><a href="https://github.com/dellstorage/storagecenter-flocker-driver/blob/master/README.md">https://github.com/dellstorage/storagecenter-flocker-driver/blob/master/README.md</a></td>
</tr>
<tr>
<td>ClusterHQ</td>
<td><a href="https://clusterhq.com/flocker/introduction/">https://clusterhq.com/flocker/introduction/</a></td>
</tr>
<tr>
<td>ClusterHQ</td>
<td><a href="https://docs.clusterhq.com/en/latest/">https://docs.clusterhq.com/en/latest/</a></td>
</tr>
<tr>
<td>ClusterHQ</td>
<td><a href="https://clusterhq.com/2016/02/16/ucp-flocker/#why-do-i-need-flocker-if-i-have-ucp">https://clusterhq.com/2016/02/16/ucp-flocker/#why-do-i-need-flocker-if-i-have-ucp</a></td>
</tr>
</tbody>
</table>