Table 1.

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6/7/2013</td>
<td>1.0</td>
<td>Initial Draft</td>
</tr>
<tr>
<td>7/15/2013</td>
<td>1.1</td>
<td>Revised</td>
</tr>
<tr>
<td>1/8/2014</td>
<td>1.2</td>
<td>Revised</td>
</tr>
<tr>
<td>3/12/2014</td>
<td>1.3</td>
<td>Revised iSCSI Setting</td>
</tr>
<tr>
<td>4/11/2014</td>
<td>1.4</td>
<td>Added multi-array storage pools</td>
</tr>
<tr>
<td>11/17/2014</td>
<td>1.5</td>
<td>Updated iSCSI &amp; Multipath</td>
</tr>
</tbody>
</table>
Table of Contents

IMPORTANT – Must Read ......................................................................................................................................................... 5

Introduction ................................................................................................................................................................................. 5

Audience ...................................................................................................................................................................................... 5

General......................................................................................................................................................................................... 5
  Sysfs pseudo file system ....................................................................................................................................................... 5
  Disk Labels and UUIDs for Persistence ............................................................................................................................ 5
  Useful Tools.............................................................................................................................................................................. 7

Nimble Array Volume Info ......................................................................................................................................................... 7

Software iSCSI ............................................................................................................................................................................ 9
  iSCSI Timeout & Performance Settings for Nimble Storage ............................................................................................. 9

TCP Tuning for Nimble Array ...................................................................................................................................................10

10GbE iSCSI with Nimble Array .............................................................................................................................................10

Volume Management ..............................................................................................................................................................11
  Adding Volumes post install ............................................................................................................................................... 11
  Multipath Configuration with Nimble Array ..................................................................................................................... 11
  Disk IO Scheduler with Nimble Array ............................................................................................................................. 11
  Setting of CPU Governor with Nimble Array ................................................................................................................ 12
  Setting of max_sectors_kb with Nimble Array ................................................................................................................... 12
  Setting of vm dirty writeback and expire with Nimble Array .......................................................................................... 12

Logical Volume Manager (LVM).............................................................................................................................................13
  EXT File System with Nimble Array ................................................................................................................................. 13
  Resizing Volumes post install ........................................................................................................................................... 14
  Resize EXT4 File System ................................................................................................................................................... 15
  XFS File System .................................................................................................................................................................. 18

non-LVM with Nimble Array ....................................................................................................................................................19

Cloning LVM volumes with Nimble Array .................................................................................................................................................19
**IMPORTANT – Must Read**
Currently, multi-array storage pool is **NOT** supported on any Linux distributions (i.e. RHEL, Oracle Linux, Ubuntu, CentOS, SuSE, Debian, etc).

**Introduction**
The purpose of this technical white paper is to provide an overview of specific information required for administering storage on Red Hat Enterprise Linux (RHEL) 6 and Oracle Linux (OL) 6 connected to a Nimble Storage.

It is important to note that in Linux, there are many different ways to achieve what is being covered in this document. This guide does not contain every possible method for all situations and only intended as a starting point of reference for System Administrators.

**Audience**
This guide is intended for Linux solution architects, storage engineers, system administrators and IT managers who analyze, design and maintain a robust Linux environment on Nimble Storage. It is assumed that the reader has a working knowledge of network design for iSCSI SAN, and basic Nimble Storage operations. Knowledge of Red Hat Enterprise Linux and Oracle Linux operating system is required.

**General**

**Sysfs pseudo file system**
Sysfs is a virtual file system provided by Linux. Sysfs exports information about devices and drivers from the kernel device model to user space and is also used for configuration. Usually each disk block device will have an entry under `/sys/block/` and each Host Bus Adapter (HBA) will have an entry under `/sys/class/scsi_host/hostX` where X is the number (starting at 0) of the HBA in the server.

**Disk Labels and UUIDs for Persistence**
All modern Linux operating are capable of discovering multiple volumes from the Nimble Storage. The newly discovered disks are given a device designation of `/dev/sda`, `/dev/sdb`, etc., depending upon how they are discovered by the Linux operating system via various interfaces connecting the server to the storage.

However, Linux operating system lacks the ability to track these disk designations persistently through reboots and dynamic additions of new volumes. There are multiple ways to ensure these disks are referenced persistently across reboots by using Disk Label or UUIDs.

Disk Label is very useful when scripting using the Nimble zero-copy clone feature. In a case where a clone of a production volume is mapped to another server, it is not necessary to know what drive letter is assigned. Therefore, Disk Label can be used in the mount command.
Note: Disk Labels will not work with a multipathed LUN and should not be used. Multipath device (LUNs) names are persistent by default and will not change. DM-Multipath does support aliasing the multipath device names for human readable names.

Below are examples of creating a file system with a label.

**Create a file system with a label:**

```
[root@mktg04 ~] # mkfs -t ext3 -L <label name> /dev/sdc
[root@mktg04 ~] # mkfs -t ext4 -L <label name> /dev/sdc
```

**Create a label after file system creation:**

```
[root@mktg04 ~] # e2label /dev/sdc <label name>
[root@mktg04 ~] # tune2fs -L <label name> /dev/sdc
```

**Discover Existing Labels**

To discover the label of an existing device, the following simple command can be used.

```
[root@mktg04 ~] # e2label /dev/sdc
```

**Using Disk Label in /etc/fstab**

The LABEL syntax can be used in a variety of places including the mount command and GRUB boot loader configuration.

```
LABEL=<label name> /mountpoint ext4 defaults 0 0
```

**Swap Space**

Swap space can also be labeled, however only at the time of creation. This isn’t a problem since no static data is stored in swap. To label an existing swap partition, execute the following commands.

```
[root@mktg04 ~] # swapoff /dev/sda1
[root@mktg04 ~] # mkswap -L <swap label> /dev/sda1
[root@mktg04 ~] # swapon LABEL=<swap label>
```

**UUIDs**

An alternative to disk label is UUID. UUID is static and safe for use but their long string can make it difficult to work with. UUID is assigned at file system creation. UUID can be discovered by using the command “tune2fs -l /dev/sdc”.

---

**Page dimensions:** 612.0x792.0

---
Useful Tools

**lsscsi**

`lsscsi` is a tool that parses information from the `/proc` and `/sys` virtual file systems into a simple human readable output. It is not included in the base install but can be install separately via the rpm command from the ISO.

```
[root@mktg04 ~]# lsscsi
[0:0:0:0] cd/dvd TEAC DV-28S-W 1.2A /dev/sr0
[1:0:0:0] disk ATA ST3500413AS JC4B /dev/sda
[9:0:0:0] disk Nimble Server 1.0 /dev/sdd
[10:0:0:0] disk Nimble Server 1.0 /dev/sde
[15:0:0:0] disk Nimble Server 1.0 /dev/sdi
[16:0:0:0] disk Nimble Server 1.0 /dev/sdk
```

**lspci**

`lspci` provides information regarding currently attached PCI devices. This can be useful for identifying not only what is currently visible to the server, but what variables/features have been associated with that device.

**scsi_id**

`scsi_id` can be used to report the volume serial number and is available in all base installations. The serial number can be matched to the Nimble Storage’s output.

![💡 Note:](image) The operating system adds a prefix (number 2) to the Nimble Storage volume’s serial number. This is the serial number that must be used in the `/etc/multipath.conf` file for multipathing.

```
[root@mktg04 ~]# scsi_id -u -g /dev/sdd
292163be9586f0dab6c9ce900251fa9df
```

**Nimble Array Volume Info**

![💡 Note:](image) To get Nimble volume serial on the Nimble Storage array, ssh to the Nimble array and login as “admin” user and issue the `vol --info <volume name>` command. This is helpful for creating the `/etc/multipath.conf` file.

```
/ $ vol --info mktg04-rman
Name: mktg04-rman
Serial number: 92163be9586f0dab6c9ce900251fa9df
```

To get all volumes serial numbers on a Nimble array, create a password-less ssh key on the Nimble array for the server running the script.
Sample output from the script:

```
[root@mktg03 ~]# determine_vol.sh cs420
Volume Name                                Serial Number
-----------                                             -------------
bd1-data1                                     414a51fd139ae4ed6c9ce900d0cc2bd34
bd1-data2                                     aa61998225b659d86c9ce900d0cc2bd34
bd1-data3                                     8ff201e535ad1056c9ce900d0cc2bd34
bd1-data4                                     607a32c5b02956df6c9ce900d0cc2bd34
bd1-data5                                     fd4881ec5929c2766c9ce900d0cc2bd34
bd1-data6                                     56d351ee48876b3e6c9ce900d0cc2bd34
bd1-data7                                     5bf6711764ff586b6c9ce900d0cc2bd34
bd1-data8                                     0cecea30d4fa67456c9ce900d0cc2bd34
bd1-log1                                      c94cf11e887b2976c9ce900d0cc2bd34
bd1-log2                                      12d281dd8e1649c76c9ce900d0cc2bd34
bd1-log3                                      17f5a106aa8e36f46c9ce900d0cc2bd34
bd1-log4                                      e87cd2e6240b9bcd6c9ce900d0cc2bd34
bd1-ocr                                       a47b02d66a9a78566c9ce900d0cc2bd34
```

Script:

```bash
#!/bin/ksh

if [ $# -ne 1 ]; then
    echo "Error: specify an array name"
    echo "Ex: `basename $0` cs420"
    echo
    exit 1
fi

ARRAY=$1

typeset heading="
-------------------- -------------------- -------------------- --------------------
Name  Size    Online Offline Usage      Reason  (MB) (MB)
----------- ----------- ---------- ---------- ----------...

# print "$heading"
TMPFILE=/tmp/file.txt

ssh admin@$ARRAY vol --list | tail -n +5 | awk '{print $1}' > $TMPFILE

printf "%-45s %s 45s %s 45s %s\n" "Volume Name" "Serial Number"

for vol in $(cat $TMPFILE)
do
    VOL=$vol
    SERIAL=$(ssh admin@$ARRAY vol --info $VOL | grep "Serial number:" | awk '{print $NF}')
    printf "%-45s %s \n" "$VOL" "$SERIAL"
done
```
Software iSCSI

iSCSI is considered to be a mature technology that allows organizations to economically scale into the world of enterprise storage. The scope of this document is focused on the Linux iSCSI software initiator (open-iscsi). For more advanced implementations (for examples leveraging iSCSI HBAs, or drivers that make use of iSCSI offload engines) please consult the associated vendor's documentation.

iSCSI Timeout & Performance Settings for Nimble Storage

Understanding the meaning of these iSCSI timeouts allows administrators to set these timeouts appropriately. These iSCSI timeouts parameters in the `/etc/iscsi/iscsi.conf` file should be set as follow:

```
node.session.timeo.replacement_timeout = 120
node.conn[0].timeo.noop_out_interval = 5
node.conn[0].timeo.noop_out_timeout = 10
node.session.nr_sessions = 4
node.session.cmds_max = 2048
node.session.queue_depth = 1024
```

= = = NOP-Out Interval/Timeout = = =

```
node.conn[0].timeo.noop_out_timeout = [ value ]
```

iSCSI layer sends a NOP-Out request to each target. If a NOP-Out request times out (default - 10 seconds), the iSCSI layer responds by failing any running commands and instructing the SCSI layer to requeue those commands when possible. If dm-multipath is being used, the SCSI layer will fail those running commands and defer them to the multipath layer. The multipath layer then retries those commands on another path. If dm-multipath is not being used, those commands are retried five times (node.conn[0].timeo.noop_out_interval) before failing altogether.

```
node.conn[0].timeo.noop_out_interval [ value ]
```

Once set, the iSCSI layer will send a NOP-Out request to each target every [ interval value ] seconds.

= = = SCSI Error Handler = = =

```
node.session.timeo.replacement_timeout = [ value ]
```

If the SCSI Error Handler is running, running commands on a path will not be failed immediately when a NOP-Out request times out on that path. Instead, those commands will be failed after replacement_timeout seconds.

```
Important: Controls how long the iSCSI layer should wait for a timed-out path/session to reestablish itself before failing any commands on it. The recommended setting of 120 seconds above allows ample time for controller failover. Default is 120 seconds.
```

```
Note: If set to 120 seconds, IO will be queued for 2 minutes before it can resume.
```

The “1 queue_if_no_path” option in `/etc/multipath.conf` sets iSCSI timers to immediately defer commands to the multipath layer. This setting prevents IO errors from propagating to the application; because of this, you can set replacement_timeout to 60-120 seconds.
TCP Tuning for Nimble Array

Like most modern OSes, Linux does a good job of auto-tuning the TCP buffers but the default maximum Linux TCP buffer sizes are still too small. Here are some parameters in the `/etc/sysctl.conf` file that can be modified. Please test thoroughly before implementing in production.

```
# Increase TCP max buffer size settable using setsockopt()
net.core.rmem_max = 16780000
net.core.wmem_max = 16780000

# Increase Linux autotuning TCP buffer limit
net.ipv4.tcp_rmem = 10240 87380 16780000
net.ipv4.tcp_wmem = 10240 87380 16780000
```

10GbE iSCSI with Nimble Array

If 10GbE iSCSI is being used, Nimble Storage recommends using Jumbo Frames. The MTU size can be set in the `/etc/sysconfig/network-scripts/ifcfg-ethX` file. If changing the file on the Ethernet interface that is already running, a restart of the Ethernet interface is required.

```
Example of MTU setting for eth1:
DEVICE=eth1
HWADDR=00:25:B5:00:00:BE
TYPE=Ethernet
UUID=31bf296f-5d6a-4caf-8858-8887e883edc
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=static
IPADDR=172.18.127.134
NETMASK=255.255.255.0
MTU=9000

To change MTU on an already running interface:
[root@bigdata2 ~]# ifconfig eth1 mtu 9000
```
**Volume Management**

**Adding Volumes post install**

Use the `iscsiadm` command to target an iSCSI IP address on the Nimble Storage:

```bash
Discover target:
[root@mktg04 ~]# iscsiadm --m discovery --t st --p <discovery IP>

Log into target:
[root@mktg04 ~]# iscsiadm --m node --T <target volume IQN> --login
```

**Multipath Configuration with Nimble Array**

The multipath parameters in the `/etc/multipath.conf` file should be set as follow in order to sustain a failover:

```conf
defaults {
    user_friendly_names yes
    find_multipaths yes
}

devices {
    device {
        vendor                  "Nimble"
        product                 "Server"
        path_grouping_policy    group_by_serial
        path_selector           "round-robin 0"
        features                "1 queue_if_no_path"
        path_checker            tur
        rr_min_io_rq            10
        rr_weight               priorities
        failback                immediate
    }
}
```

**Disk IO Scheduler with Nimble Array**

Use “noop” io scheduler on all Nimble volumes.

To set at boot time, add the elevator option at the kernel line in the `/etc/grub.conf` file:

```
elevator=noop
```

To manually set for a particular disk:

```
[root@mktg04 ~]# echo noop > /sys/block/sd?/queue/scheduler
```

Script to change IO Scheduler for all devices. Note: If using the script, when new devices are added, the scripts must be executed again.
Setting of CPU Governor with Nimble Array

Use "performance" setting for all available CPUs on the host when possible.

To change CPU governor setting to performance:

```
[root@mktg04 ~]# echo performance > /sys/devices/system/cpu/cpu#/cpufreq/scaling_governor
```

Change all CPUs

```
[root@mktg04 ~]# for a in $(ls -ld /sys/devices/system/cpu/cpu[0-9]* | awk '{print $NF}'); do echo performance > $a/cpufreq/scaling_governor; done
```

Note: To make this change persistent after reboot, add the commands in /etc/rc.local file.

Setting of max_sectors_kb with Nimble Array

Change max_sectors_kb on all volumes to 1024 (default 512).

To change max_sectors_kb to 1024 for a single volume:

```
[root@mktg04 ~]# echo 1024 > /sys/block/sd?/queue/max_sectors_kb
```

Change all volumes:

```
multipath -ll | grep sd | awk -F":" '{print $4}' | awk '{print $2}' | while read LUN; do echo 1024 > /sys/block/$LUN/queue/max_sectors_kb; done
```

Note: To make this change persistent after reboot, add the commands in /etc/rc.local file.

Setting of vm dirty writeback and expire with Nimble Array

Change vm dirty writeback and expire to 100 (default 500 and 3000 respectively)
To change vm dirty writeback and expire:

[root@mktg04 ~]# echo 100 > /proc/sys/vm/dirty_writeback_centisecs
[root@mktg04 ~]# echo 100 > /proc/sys/vm/dirty_expire_centisecs

Note: To make this change persistent after reboot, add the commands in /etc/rc.local file.

**Logical Volume Manager (LVM)**

LVM allows grouping of multiple volumes into a single logical volume for file system creation. The Nimble Storage array supports the creation of file system on a single volume or a logical volume. Users who choose LVM over a single volume for file system creation must understand in depth how LVM behaves.

LVM does provide more performance gain (especially read) over a single volume (non-LVM) due to multiple volumes accessing the data.

Note: Nimble does NOT recommend using fdisk for partitioning. Only use whole disk approach. Also, if using Nimble Storage Snapshot feature with LVM volumes for cloning on the same host, the command `vgimportclone` must be used in order to change the VGID and PVID.

**EXT File System with Nimble Array**

The number of Nimble volumes for a LVM volume group depends on the number of CPU cores. Best practice is to have the number of volumes equals half the number of CPU cores. For example, if a server has 16 physical cores, then the number of volumes for a LVM volume group should be 8.

When creating an EXT file system on a logical volume, the `stride` and `stripe-width` options must be used. These two parameters minimize the IO un-alignment on the Nimble array.

For example:

- `stride=2,stride-width=16` (for Nimble performance policy 8KB block size with 8 volumes)
- `stride=4,stride-width=32` (for Nimble performance policy 16KB block size with 8 volumes)
- `stride=8,stride-width=64` (for Nimble performance policy 32KB block size with 8 volumes)

Note: The stripe-width value depends on the number of volumes, and the stride size. The calculator can be found here [http://busybox.net/~aldot/mkfs_stride.html](http://busybox.net/~aldot/mkfs_stride.html)

For example: If there is one Nimble volume with 8KB block size performance policy, then it should look like this.
Create a volume group with 8 physical volumes with 8KB block size:
[root@mktg04 ~]# vgcreate vg01 /dev/mapper/lun[1-8]

Create a logical volume:
[root@mktg04 ~]# lvcreate -l <number of extents> -i 8 -I 4096 -n vol1 vg01

Create an EXT4 file system:
[root@mktg04 ~]# mkfs.ext4 /dev/vg01/vol1 -b 4096 -E stride=2,stripe-width=16

Add the entry in /etc/fstab file:
/dev/vg01/vol1  /mountpoint  ext4  _netdev,noatime,nodiratime,discard,barrier=0  0 0

Mount the file system:
[root@mktg04 ~]# mount -a

---

**Resizing Volumes post install**

After resizing the Nimble volumes using the Nimble Storage GUI, use the `iscsiadm` command to scan for the newly sized volumes:

**Rescan:**
[root@mktg04 ~]# iscsiadm -m node -R


/var/log/messages output after resize operation:
Jan 14 10:56:30 mktg04 kernel: sd 3:0:0:0: [sdb] 209715200 512-byte logical blocks: (107 GB/100 GiB)
Jan 14 10:56:30 mktg04 kernel: sdb: detected capacity change from 53687091200 to 107374182400

Reload multipath configuration if multipath is used:
[root@mktg04 ~]# /etc/init.d/multipathd reload
Reloading multipathd: [ OK ]
Resize EXT4 File System

The below example shows how to resize an EXT4 file system mounted on /u02.

Before Resize:

- 2 Nimble LUNs (50GB each) were created and mapped to a host

```
[root@mktg02 ~]# multipath -ll
data2 (2f92a2d58518cb6ea6c9ce900251fa9df) dm-0 Nimble,Server
  size=50G features='1 queue_if_no_path' hwhandler='0' wp=rw
  `-+- policy='round-robin 0' prio=1 status=active
     | - 29:0:0:0 sdd 8:48 active ready running
     | `- 30:0:0:0 sde 8:64 active ready running

data1 (25ad718a0dad588d06c9ce900251fa9df) dm-1 Nimble,Server
  size=50G features='1 queue_if_no_path' hwhandler='0' wp=rw
  `-+- policy='round-robin 0' prio=1 status=active
     | - 27:0:0:0 sdb 8:16 active ready running
     | `- 28:0:0:0 sdc 8:32 active ready running
```

Create a volume group

```
[root@mktg02 ~]# vgcreate vg01 /dev/mapper/data1 /dev/mapper/data2
No physical volume label read from /dev/mapper/data1
No physical volume label read from /dev/mapper/data2
Writing physical volume data to disk "/dev/mapper/data1"
Physical volume "/dev/mapper/data1" successfully created
Writing physical volume data to disk "/dev/mapper/data2"
Physical volume "/dev/mapper/data2" successfully created
Volume group "vg01" successfully created
```

Verify VG & PV

```
[root@mktg02 ~]# vgdisplay vg01
--- Volume group ---
  VG Name       vg01
  System ID
  Format        lvm2
  Metadata Areas 2
  Metadata Sequence No 1
  VG Access      read/write
  VG Status      resizable
  MAX LV         0
  Cur LV         0
  Open LV        0
  Max PV         0
  Cur PV         2
  Act PV         2
  VG Size        99.99 GiB
  PE Size        4.00 MiB
  Total PE       25598
  Alloc PE / Size 0 / 0
  Free PE / Size 25598 / 99.99 GiB
  VG UUID       3rhxM1-uRxc-AQIC-XICu-13IS-x7Jw-nA1YIX

[root@mktg02 ~]# pvs
PV  VG   Fmt Attr PSize  PFree
/dev/mapper/data1 vg01 lvm2 a-- 50.00g 0
/dev/mapper/data2 vg01 lvm2 a-- 50.00g 0
```

Create LV
[root@mktg02 ~]# lvcreate -l 25598 -i 2 -I 4096 -n vol1 /dev/vg01
  Logical volume "vol1" created

[root@mktg02 ~]# mkfs.ext4 -b 4096 /dev/vg01/vol1

Create EXT4 File System

[root@mktg02 ~]# mount /dev/vg01/vol1 /u02

Mount File System and verify df command output

[root@mktg02 ~]# df /u02

Resize Operations

Increase the 2 Nimble volumes to 100GB each using the Nimble Management GUI

[root@mktg02 ~]# lscliadm -m node -R

Verify rescan from /var/log/messages

Jul 15 10:33:44 mktg02 kernel: sd 27:0:0:0: [sdb] 209715200 512-byte logical blocks: (107 GB/100 GiB)
Jul 15 10:33:44 mktg02 kernel: sdb: detected capacity change from 53687091200 to 107374182400
Reload multipath and verify

```
root@mktg02 ~]# /etc/init.d/multipathd reload
Reloading multipathd:                                      [ OK ]
```

```
[root@mktg02 ~]# multipath -ll
data2 (2f92a2d58518cb6a6c9ce900251fa9df) dm-0 Nimble,Server
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
  `-- policy='round-robin 0' prio=1 status=active
     `-- 30:0:0:0 sde 8:64 active ready running
     `- 29:0:0:0 sdd 8:48 active ready running

data1 (25ad718a0dad588d06c9ce900251fa9df) dm-1 Nimble,Server
size=100G features='1 queue_if_no_path' hwhandler='0' wp=rw
  `-- policy='round-robin 0' prio=1 status=active
     `- 27:0:0:0 sdb 8:16 active ready running
     `- 28:0:0:0 sdc 8:32 active ready running
```

Resize PV and verify

```
[root@mktg02 ~]# pvresize /dev/mapper/data1
Physical volume "/dev/mapper/data1" changed
1 physical volume(s) resized / 0 physical volume(s) not resized
```

```
[root@mktg02 ~]# pvresize /dev/mapper/data2
Physical volume "/dev/mapper/data2" changed
1 physical volume(s) resized / 0 physical volume(s) not resized
```

```
[root@mktg02 ~]# pvs
PV       VG  Fmt  Attr PSize   PFree
/dev/mapper/data1 vg01 lvm2 a--  100.00g 50.00g
/dev/mapper/data2 vg01 lvm2 a--  100.00g 50.00g
```

Verify VG w/new size

```
[root@mktg02 ~]# vgdisplay vg01
--- Volume group ---
  VG Name               vg01
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               199.99 GiB
  PE Size               4.00 MiB
  Total PE              51198
  Alloc PE / Size       25598 / 99.99 GiB
  Free PE / Size        25600 / 100.00 GiB
  VG UUID               3rhoM1-uRx-AQiC-XICu-13IS-x7Jw-nA1YIX
```
- **Resize LV (increase from 100GB to 150GB)**

[root@mktg02 ~]# lvresize -l+12799 vg01/vol1
Using stripesize of last segment 4.00 MiB
Rounding size (38397 extents) up to stripe boundary size for segment (38398 extents)
Extending logical volume vol1 to 149.99 GiB
Logical volume vol1 successfully resized

- **Verify PV & LV**

[root@mktg02 ~]# pvs
PV      VG   Fmt  Attr PSize   PFree
/dev/mapper/data1 vg01 lvm2 a--  100.00g 25.00g
/dev/mapper/data2 vg01 lvm2 a--  100.00g 25.00g

[root@mktg02 ~]# lvdisplay /dev/vg01/vol1
--- Logical volume ---
LV Path                /dev/vg01/vol1
LV Name                vol1
VG Name                vg01
LV UUID                xb9Jo1-YdeF-26SE-7htE-xVsS-CFFI-lvZv07
LV Write Access        read/write
LV Creation host, time mktg02, 2013-07-15 10:16:13 -0700
LV Status              available
# open                 1
LV Size                149.99 GiB
Current LE             38398
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     32768
Block device           252:2

- **Resize EXT4 File System**

[root@mktg02 ~]# resize2fs /dev/vg01/vol1
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/vg01/vol1 is mounted on /u02; on-line resizing required
old desc_blocks = 7, new_desc_blocks = 10
Performing an on-line resize of /dev/vg01/vol1 to 39319552 (4k) blocks.
The filesystem on /dev/vg01/vol1 is now 39319552 blocks long.

- **Verify df command**

[root@mktg02 ~]# df /u02
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg01-vol1 154810348    191956 146755536   1% /u02

---

**XFS File System**

Create a volume group with 8 physical volumes:
[root@mktg04~]# vgcreate vg01 /dev/mapper/lun[1-8]

Create a logical volume:
**non-LVM with Nimble Array**

When creating a file system on a single volume, Nimble Storage does NOT recommend using fdisk to create a partition. Please refer to the “EXT file system with Nimble Array” section for stride and stripe-width settings.

Create a file system on a single volume with 8KB block size:

```
[root@mktg04 ~]# mkfs.ext4 /dev/mapper/mpathX -b 4096 -E stride=2,stripe-width=2
```

**Cloning LVM volumes with Nimble Array**

The Nimble array has the snapshot feature that is useful for backup and recovery. However, when recovering files using the Nimble snapshot in an LVM environment, administrators should understand how it works. Since the snapshot is identical to the original volumes, VGID and PVID must be changed before mounting the logical volume. This can be accomplished by using the "vgimportclone" command. After creating a snapshot of volume(s) on a Nimble array belonging to a LVM volume group, clone volume(s) must be created from the snapshot and presented to the same server.

Step by step for mounting a clone:

1. Create a snapshot of volume(s) using the Nimble GUI or CLI
2. Create clone volume(s) and map to the same server
3. Scan for the cloned volumes
   ```
   [root@bigdata2 ~]# iscsiadm -m discovery -t st -p x.x.x.x
   [root@bigdata2 ~]# iscsiadm -m node –login
   ```
4. Reload multipathd
   ```
   [root@bigdata2 ~]#/etc/init.d/multipathd reload
   ```
5. Run the vgimportclone command. The /dev/mapper/mpathX devices are found by using the multipath -ll command.
   ```
   [root@bigdata2 ~]# vgimportclone --basevgname vg01_snap /dev/mapper/mpathf /dev/mapper/mpathg /dev/mapper/mpathi
   ```
6. Activate vg01_snap volume group
   ```
   [root@bigdata2 ~]# vgchange -a y vg01_snap
   ```
7. Mount the cloned logical volume
   ```
   [root@bigdata2 ~]# mkdir /vg01_snap
   [root@bigdata2 ~]# mount /dev/vg01_snap/vol1 /vg01_snap
   ```
Space Reclamation with Nimble Array

The Nimble array has the ability to reclaim un-used space back to the storage pool. In order to reclaim space to the storage pool, the "discard" mount option for EXT file system must be specified in the mount command or in /etc/fstab.

**Mount command:**

```
[root@bigdata2 ~]# mount -t ext4 -o _netdev,noatime,nodiratime,barrier=0,discard /dev/mapper/vol1 /mountpoint
```

**/etc/fstab file:**

```
/dev/mapper/vol1 /mountpoint ext4 _netdev,noatime,nodiratime,barrier=0,discard 0 0
```