

## Adding an Additional Volume Group to an existing Application

### Description

This short document describes the process of how an existing clustered application can be extended with another volume group. This is useful if you intend to add further file systems to an application.

Please note that this process is not suitable for *extending* an existing file system - to do that you instead need to follow a different set of instructions - these should appear on the website shortly.

### Existing Sample Application

The existing servers make use of “/dev/sda” for storage, which is configured as follows:

```
root@slack10s1:~# fdisk -l
```

```
Disk /dev/sda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	366	2939863+	83	Linux
/dev/sda2		367	398	257040	82	Linux swap
/dev/sda3		399	1044	5188995	5	Extended
/dev/sda5		399	460	497983+	8e	Linux LVM

The application running under Linuxha.net makes use of “/dev/sda5” as a physical volume called “testvg” which is used for two existing file systems, which are mounted as follows:

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/drbd0	126931	4129	116249	4%	/testvg/test1
/dev/drbd1	63461	4127	56058	7%	/testvg/test2

The application “test” is already running on the first server in the cluster:

```
root@slack10s1:~# clstat
Cluster: slack10c1 - UP
```

Node	Status
slack10s1	UP
slack10s2	UP

Application	Node	State	Started	Monitor	Stale	Fail-over?
test	slack10s1	STARTED	0:00:06	Running	0	Yes
test2	N/A	DOWN	N/A	N/A	N/A	Yes

## Additional Storage Configuration

### Make New Physical Volumes

Additional storage has been made available to each server, in the form of a 1Gb IDE disc:

```
root@slack10s1:~# fdisk -l /dev/hda

Disk /dev/hda: 1073 MB, 1073741824 bytes
16 heads, 63 sectors/track, 2080 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

Disk /dev/hda doesn't contain a valid partition table
```

Since the application is already up and running, adding a partition table to the disk and having to reboot for the kernel to recognise it is not an option, so on each host simply make the whole disk a physical volume for volume manager:

```
root@slack10s1:~# pvcreate /dev/hda
pvcreate -- physical volume "/dev/hda" successfully created

root@slack10s2:~# pvcreate /dev/hda
pvcreate -- physical volume "/dev/hda" successfully created
```

### Creating New Volume Group

On each machine use the “vgcreate” command using the “/dev/hda” device to create a new volume group - in this case called “newvg”:

```
root@slack10s1:~# vgcreate newvg /dev/hda
vgcreate -- INFO: using default physical extent size 32 MB
vgcreate -- INFO: maximum logical volume size is 2 Terabyte
vgcreate -- doing automatic backup of volume group "newvg"
vgcreate -- volume group "newvg" successfully created and activated

root@slack10s2:~# vgcreate newvg /dev/hda
vgcreate -- INFO: using default physical extent size 32 MB
vgcreate -- INFO: maximum logical volume size is 2 Terabyte
vgcreate -- doing automatic backup of volume group "newvg"
vgcreate -- volume group "newvg" successfully created and activated
```

### Add New File System

In this case the new file system to mount as part of the cluster will be “/test/newfs”, and will be 256 Mb and be part of the new “newvg” volume group. Since the application is already running on “slack10s1” the new volume is created and mounted on that host, for example:

```
root@slack10s1:~# lvcreate -L 256 -n newfs /dev/newvg
lvcreate -- doing automatic backup of "newvg"
lvcreate -- logical volume "/dev/newvg/newfs" successfully created
```

Now create an “ext3” (or “xfs”, “reiserfs” or “jfs”) file system on that device:

```
root@slack10s1:~# mkfs -t ext3 /dev/newvg/newfs
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=1
32 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185
```

```
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

This filesystem will be automatically checked every 31 mounts or 180 days, whichever comes first. Use `tune2fs -c` or `-i` to override.

Create the mount point *on both* servers, and mount the file system on “slack10s1” (where the application is currently running):

```
root@slack10s1:~# mkdir /test /test/newfs
root@slack10s1:~# chmod -R 755 /test
```

```
root@slack10s2:~# mkdir /test /test/newfs
root@slack10s2:~# chmod -R 755 /test
```

```
root@slack10s1:~# mount /dev/newvg/newfs /test/newfs
```

Now consider the mounted file systems on “slack10s1”:

```
root@slack10s1:~# df
Filesystem                1K-blocks      Used Available Use% Mounted on
/dev/sda1                  2847720    907396   1793332   34% /
/dev/drbd0                  126931      4129    116249    4% /testvg/test1
/dev/drbd1                   63461      4127     56058    7% /testvg/test2
/dev/newvg/newfs           253871      8239    232525    4% /test/newfs
```

Obviously the new file system is making use of local storage only - it needs to be integrated into the cluster.

## Modifying Cluster Application “test”

### Configuration File Changes

The “`clbuildapp`” command can be used against a running application - though it should be used on the node where the application is currently running - “slack10s1” in this example. The first step is to change the “`/etc/cluster/test/appconf.xml`” to recognise the new volume group.

The current contents of this section of the configuration file are:

```
<vg>
    <name>testvg</name>
    <type>filesystems</type>
</vg>
```

This should be changed to:

```
<vg>
    <name>testvg</name>
    <type>filesystems</type>
    <name>newvg</name>
    <type>filesystems</type>
</vg>
```

### Application Rebuild

The next step is to run the “`vgbuild`” option of the “`clbuildapp`” command. This generates much output - only the important lines are repeated below:

```
root@slack10s1:~# clbuildapp -A test -V --build -F
...
INFO 09/03/2006 17:00:33 REBUILD: Original fsmap entry count: 2
INFO 09/03/2006 17:00:33 Getting list of defined volume groups on slack10s2
INFO 09/03/2006 17:00:33 Validated Meta volume exists for test1 (and is 128mb)
INFO 09/03/2006 17:00:33 Validated Meta volume exists for test2 (and is 128mb)
```

```

INFO 09/03/2006 17:00:34 All LV checks on VG testvg completed
INFO 09/03/2006 17:00:34 VG newvg does not contain LV newfs_meta on slack10s1 -
creating...
INFO 09/03/2006 17:00:35 VG/LV newvg/newfs_meta of 131072Kb built on slack10s1
INFO 09/03/2006 17:00:35 VG newvg does not contain LV newfs on slack10s2 - creating...
INFO 09/03/2006 17:00:35 VG/LV newvg/newfs of 262144Kb built on slack10s2
INFO 09/03/2006 17:00:35 VG newvg does not contain LV newfs_meta on slack10s2 -
creating...
INFO 09/03/2006 17:00:36 VG/LV newvg/newfs_meta of 131072Kb built on slack10s2
INFO 09/03/2006 17:00:36 All LV checks on VG newvg completed
INFO 09/03/2006 17:00:36
INFO 09/03/2006 17:00:36 Now run with the --build to ensure all required resources
INFO 09/03/2006 17:00:36 are allocated on both nodes
INFO 09/03/2006 17:00:36

```

Next use the "--build" option - though please note the below very carefully - there is a bug in 1.0.8 which must be worked around:

```
root@slack10s1:~# clbuildapp -A test -V --build -F
```

This command will fail - with something similar to:

```

ERROR 09/03/2006 17:05:23 the clbuildapp commands has not been rerun to validate the
ERROR 09/03/2006 17:05:23 changes. You can ignore this warning and add a --nochecksums
ERROR 09/03/2006 17:05:23 option to the command if you so wish.

```

However simply re-mount the file system and repeat the command:

```

root@slack10s1:~# mount /dev/newvg/newfs /test/newfs
root@slack10s1:~# clbuildapp -A test -V --build -F

```

This should work this time, and include output similar to the following:

```

INFO 09/03/2006 17:09:15 REBUILD: Current fsmap being replaced on slack10s1.
INFO 09/03/2006 17:09:15 Validated/created fsmap directory on slack10s1
INFO 09/03/2006 17:09:15 Validated/created fsmap directory on slack10s2
INFO 09/03/2006 17:09:15 Validated/created application config directory on slack10s2
INFO 09/03/2006 17:09:15 Copied config data for test to slack10s2
INFO 09/03/2006 17:09:15 REBUILD: Written testvg:test1:/testvg/test1:ext3:rw:131072
INFO 09/03/2006 17:09:15 REBUILD: Written testvg:test2:/testvg/test2:ext3:rw:65536
INFO 09/03/2006 17:09:15 Validated file system mount point /test/newfs
INFO 09/03/2006 17:09:15 REBUILD: Removing original fsmap file on slack10s1.
INFO 09/03/2006 17:09:15 REBUILD: Recreated fsmap data for test on slack10s1
INFO 09/03/2006 17:09:16 Copied fsmap data for test to slack10s2
INFO 09/03/2006 17:09:16 REBUILD: Un-mounted local /test/newfs.
INFO 09/03/2006 17:09:16 REBUILD: DRBD device for /test/newfs started to synchronise.
INFO 09/03/2006 17:09:16 REBUILD: Fsck for /test/newfs; "/sbin/fsck -t ext3 -a
/dev/drbd10"...
INFO 09/03/2006 17:09:16 REBUILD: Mounting file system /test/newfs
(PATH=$PATH:/sbin:/bin:/usr/sbin; mount -t ext3 -o rw /dev/drbd10 /test/newfs)
INFO 09/03/2006 17:09:16 REBUILD: File system /test/newfs mounted successfully.

```

Following this the mounted file systems should be as expected:

```

root@slack10s1:~# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/sda1        2847720    907684   1793044   34% /
/dev/drbd0       126931      4129    116249    4% /testvg/test1
/dev/drbd1       63461      4127    56058    7% /testvg/test2
/dev/drbd10     253871      8239   232525    4% /test/newfs

```

Note: The "/test/newfs" is mounted, but this time on a DRBD device.

## Check Synchronization Progress

Once the file system has been mounted the contents will be synchronized to the other node in the cluster. The "clstat" command can be used to query this synchronization:

```
root@slack10s1:~# clstat -A test
Cluster: slack10cl - UP
```

Application	Node	State	Runnig	Monitor	Stale	Fail-over?
test	slack10s1	STARTED	0:00:50	Running	1	Yes

#### File Systems

Mount Point	Valid	Type	State	% Complete	Completion
/testvg/test1	both	drbd	Sync		
/testvg/test2	both	drbd	Sync		
/test/newfs	local	drbd	Syncing	1 %	0:00:32:12

#### General Monitors

Type	Name	Status
Flag Check	flag_check	Running
FS Monitor	fsmonitor	Running

If the synchronization is too slow, consider temporarily raising it using "drbd\_tool". For example to increase it to 2Mb/sec use:

```
root@slack10s1:~# /sbin/cluster/utils/drbd_tool --application test \  
--action set_rate --rate=2048 -verbose
```

Now the application in question is working with a new file system in a new volume group - all without having to stop the application.