**PART-A GPFS-Topics Administration || ClusterA – rh94-20,21,22,25; ClusterB – rh94-30**

| | |
|---|---|
| How to install gpfs manually<br><br>How to install using spectrumscale<br><br>How to upgrade manually<br><br>How to upgrade using spectrumscale<br><br>How to monitor health of the clusters<br><br>How to monitor the performance of the cluster<br><br>How to encrypt (data at rest and in motion) | |

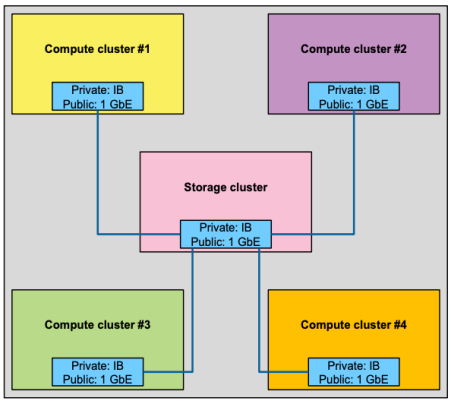
**PART-B GPFS-Topics || GPFS-Multicluster-AFM-CNFS-CES-commands**

ClusterA – vm01, vm02, vm03; ClusterB – vm04 || **ClusterA – rh94-20,21,22,25; ClusterB – rh94-30**

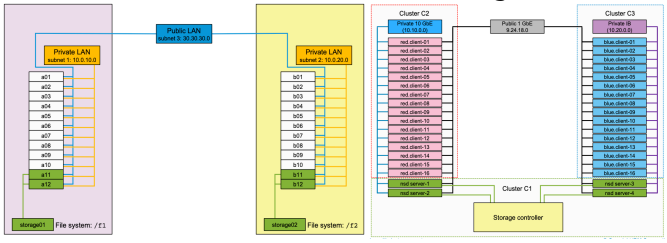
**Multicluster (Context: Cluster B mounting fs from Cluster A)**

| Cluster A (Remote cluster) | Cluster B (Home cluster) |
|---|---|
| **mmlscluster >> mmgetstate -a >> mmlslicense -L** | **mmlscluster >> mmgetstate -a >> mmlslicense -L** |
| **mmlsfs all -T (O/P: /dev/fs1 mounted on /gpfs/fs1)**<br>**mmlsnsd (nsd1,nsd2 (vm01,vm02); nsd3,nsd4 (vm02,vm01))**<br>**mmauth show** (Cipher List: AUTH ONLY) same in sides | **mmlsfs all -T (O/P: /dev/fs2 mounted on /gpfs)**<br>**mmlsnsd (nsd1,nsd2, nsd3,nsd4 (vm04))**<br>**mmauth show** (Cipher List: AUTH ONLY) |
| **scp vm04:/var/mmfs/ssl/id_rsa.pub /tmp/clusterB_id_rsa.pub** | **scp vm01:/var/mmfs/ssl/id_rsa.pub /tmp/clusterA_id_rsa.pub** |
| **mmauth add clusterB.vm04 -k /tmp/clusterB_id_rsa.pub** | **mmremotecluster add clusterA.vm01 -n vm01,vm02 -k /tmp/clusterA_id_rsa.pub** |
| **mmauth show (Cipher List: AUTH ONLY)**<br>**mmauth grant clusterB.vm04 -f fs1** | **mmremotefs add remotefsFromHome -f fs1 -C clusterA.vm01 -T /remotefs** |
| **mmlsmount all -L** | **mmmount remotefsFromHome**<br>**mmremotefs show**<br>**mmlsmount all -L** |
| **Challenges: consistent uid:gid across two clusters and cluster versions needs to be same**<br># uid:gid mapping @clusterA if not using ldap | |
| **mmauth grant clusterB -f fs1 -r uid:gid (root user uid:gid remap)** | |

**Multicluster MISC:**




Remote cluster facilitates administration delegation.
Home/Remote terminology is used per filesystem basis.
Remote cluster uses NSD protocol for high speed data sharing.
The TCP/IP traffic is carrying block device across fiber as IP payload.
GPFS daemons and the NIC creates the block payload and put it inside IP packet.
GPFS version must match (**mmlsconfig minReleaseLevel, mmlsfs all -V**), **mmauth show** must match.



**mmchconfig subnets="10.0.10.0" -N nodelist.a**
**mmchconfig subnets="10.0.20.0" -N nodelist.b**

**mmchconfig subnets="10.10.0.0 10.10.0.0/C1" -N all**
**mmchconfig subnets="10.20.0.0 10.20.0.0/C1" -N all**
**mmchconfig subnets="10.10.0.0 10.10.0.0/C2" -N nsd1,nsd2**
**mmchconfig subnets="10.20.0.0 10.10.0.0/C3" -N nsd3,nsd4**

**AFM (Context: Cluster A is cache cluster, Cluster B is Home cluster)**

**Cluster B (Home cluster)**

```
mmcrfileset fs2 home1
mmlinkfileset fs2 home1 -J /gpfs/home1

/etc/exports:
/gpfs/home1 *(no_root_squash,rw,fsid=123)

systemctl restart nfs-server
exportfs -a

mmafmconfig enable /gpfs/home1
dd if=/dev/zero of=/gpfs/home1/file1 bs=64k count=10
dd if=/dev/zero of=/gpfs/home1/file2 bs=64k count=10
```

**Tuning** at cache fileset side **Cluster A (Cache cluster):**

**afmAsyncDelay = 60s**
>> this delay cache multiple writes to single writes for write intensive loads

**afmFastCreate = yes**
>> reduces RPC workload between application and GW.

For NFS parallel transfer (used only when filesize exceeds this):
**afmParallelReadThreshold = 2048 MiB**
**afmParallelWriteThreshold = 2048 MiB**

**afmNumReadThreads = 8 || afmNumWriteThreads = 8**

**afmParallelReadChunkSize = 1MB**
**afmParallelWriteChunkSize = 1MB**

```
# cache revalidation interval
afmDirLookupRefreshInterval afmDirOpenRefreshInterval
afmFileLookupRefreshInterval  fmFileOpenRefreshInterval
```

**Cluster A (Cache cluster)**

```
mmchnode --gateway -N vm03  >> mmlscluster

RO cache example:
mmcrfileset fs1 read-only1 -p afmMode=read-only
--inode-space new  \
-p afmTarget=nfs://vm04/gpfs/home1
```

**(you can add multiple nfs server like vm04 at Home)**

```
mmlinkfileset fs1 read-only1 -J /gpfs/read-
only1

ls -ls /gpfs/read-only1
(display file1 and file2 with size = 0 as the
leftmost column)

cat /gpfs/read-only1/file1 > /dev/null
ls -ls /gpfs/read-only1
(reading will prefetch the file)

Single writer cache example:
mmcrfileset fs1 single-writer -p afmMode=sw --
inode-space new  -p
afmTarget=nfs://vm04/gpfs/home1

# for SW cache
dd if=/dev/zero of=/gpfs/single-writer/swfile1 bs=64k
count=10
(these files arrear at home side)
```

**AFM MISC:**

**mmafmctl fs1 getstate**

| Fileset Name | Fileset Target | Cache State | Gateway Node | Queue Length | Queue numExec |
|---|---|---|---|---|---|
| read-only1 | nfs://rh94-30/gpfs/home1 | Active | rh94-22 | 0 | 5 |

States are: **Active**| **Dirty**| **Needs resync**| **Disconnected**

**mmafmctl fs1 resync -j singe-writer1** >> **mmafmctl fs1 resumeRequeued -j singe-writer1 (only for SW)**
(cannot be done for IW and RO cache)

**mmlsfileset fs1 -afm (to display afmTarget mappings)**

**mmchfileset -p afmTarget=disable** (converts cache fileset to independent fileset):

```
RO does periodic revalidation with Home
LU, once updated, does not do revalidation
SW writes to Home, Home must be read-only.

IW does revalidation, last writer wins.
Do not mix SW and IW to same Home!!

A Home export can be cached by multiple cache filesets.
A cache fileset can cache one Home export.
resync cannot be done for RO and IW
Each cache fileset is mapped to one GW, that acts as primary GW for the fileset.
```

If the primary GW for a cache fileset fails another GW takes over, acting as primary until the original primary GW returns.

There could be multiple GW, secondary kick off when primary fails
All nodes other than the Primary GW for the fileset are application nodes.
Read cache req are handles synchronously, writes are queued by GW node and handled asynchronously.

Multiple GW nodes can be mapped to each NFS svr.
Each GW node can be mapped to only one NFS svr.

When numExec increase, it means either data was accessed first time OR needs to be revalidated.

**Disconnected happen when there is a wan issue for more than afmDisconnectTimeout (60sec). A stat() on the directory triggers AFM to recognize loss if connections. AFM recognizes when the connection to Home is available and restores itself.**

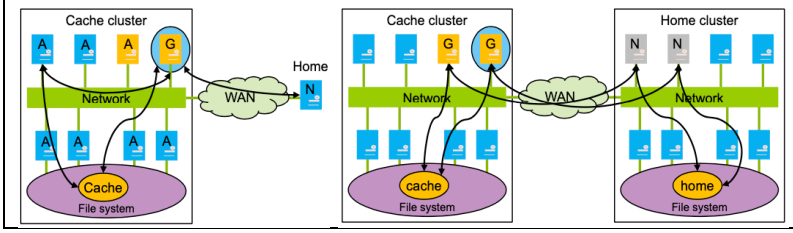
If a fileset is running in disconnected mode for a long time it may be resynched to Home by the fact that disconnection might have caused some local or remote outages.

Multiple GW nodes can be configured per cache fileset basis to do Parallel transfer (one gw is primary, other gw is participating nodes), Home must have multiple NFS svrs.

To enable parallel transfer
**mmafmconfig add map1 –export-map nfs1/gw1, nfs2/gw2**
**mmafmconfig show all**

pagepool >> Disk >> Remote Home

Qns: When do you use AFM?
- Geographically dispersed system share their files over WAN.
- Files should only be transferred when accessed and cached locally
- Asynchronous operation, no consistent lock checking




**Fileset(s)**

**mmcrfileset fs2 fs2set2 --inode-space new** (independent)
**mmcrfileset fs2 fs2set1** (dependent)

**mmlinkfileset fs2 fs2set2 -J /gpfs/fs2set2**
(now create a file there, let's say /gpfs/fs2set2/file1)

**mmlsattr -L /gpfs/fs2set2/file1** (shows which fileset it belongs to)

**mmcrsnapshot fs2 snap2 -j fs2set2>>** ls -l /gpfs/fs2set2/.snapshots >> **mmrestorefs fs2 snap1 -j fs2set2**

**mmunlinkfileset** >> **mmdelfileset** (To undo)

Misc:
How to identify independent Vs dependent fileset? (Hint: **mmlsfileset fs2 -i** >> InodeSpace = 0 for depen.)
Difference between independent and dependent?
When to use them?
Snapshot create and restore of independent fileset

**CNFS (Context: Cluster A is CNFS cluster, Cluster B plain client mounting fs1 as NFS)**

**Cluster A (cnfs nodes: vm02, vm03) must be admin nodes**

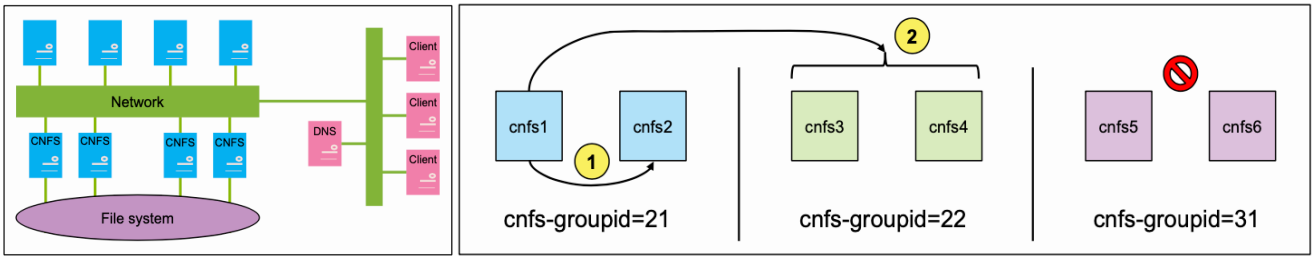
```
mmdsh -N all systemctl stop|disable nfs|nfs-lock 2> /dev/null

mkdir /gpfs/cnfsSR >> mmchconfig cnfsSharedRoot=/gpfs/cnfsSR

mkdir /gpfs/exp1 >> mmdsh -N vm02,vm03 cat /etc/exports
vm02, vm03: /gpfs/exp1 *(no_root_squash,rw,fsid=234)

mmchnode --cnfs-interface=192.168.1.90 -N vm02
mmchnode --cnfs-interface=192.168.1.91 -N vm03

showmount -e vm02| vm03 >> /gpfs/exp1 *

mmlscluster --cnfs (look at CNFS state, group, CNFS IP)
Node  Daemon node name   IP address  CNFS state  group  CNFS IP address
---------------------------------------------------------------------
   2  vm02              192.168.1.2 enabled        0   192.168.1.90
   3  vm03              192.168.1.3 enabled        0   192.168.1.91
```

**Cluster B**

```
mmshutdown

mkdir /mnt/exp1

mount cnfs:/gpfs/exp1 /mnt/exp1

(hard mounted)

Done!
```

**Tuning (At cluster A):**
**cnfsNFSDProcs = 64**
**nfsPrefetchStrategy=1**
**maxFilestoCache**
**pagepool**

**CNFS Misc:**




**mmnfsmonitord** monitors nfsd, mountd, statd, lockd.
CNFS server must be admin nodes(same for CES nodes).
Load distribution happens via round-robbin DNS config to multiple CNFS server.
A single DNS hostname is mapped over multiple IP of the CNFS server.
Very low cost over standard TCP/IP (~ 10G) than IB.
CNFS is preferred over CES for metadata intensive workloads.

CNFS monitoring utility detects the failure >>
CNFS nodes enter a grace period to block all NFS client lock requests >>
GPFS does lock recovery including release of any locks held by failing node >>
CNFS IP is moves over to other nodes and guarantees NFS lock recovery including gratuitous ARPs >>
Clients reclaim locks according to NFS standard.

Alias IP recommended as the node does not have to wait for network switch to accept link during IP fail.
Default failover group is 0.
CNFS transfers load in the same range of ten as the failing node.

There is no fileset creation needed

**Troubleshooting:**
**mmlscluster --cnfs** >> **/var/adm/ras/mmfsadm.log** >> **ip -a**

**CES (Context: Cluster A is CES cluster, Cluster B plain Client)**

**Cluster A (CES nodes: vm02, vm03) Static IP is not possible**

**systemctl stop|disable nfs,nfslock || mmchnode --nogateway -N vm03**

**spectrumscale node list >> spectrumscale node add vm02 -p (same for vm03) >> spectrumscale**

**mkdir /gpfs/cesSR >> mmchconfig cesSharedRoot=/gpfs/cesSR**

**mmchnode --ces-enable -N vm02 (vm03)** >> mmdsh -N cesnodes ps -e | grep **mmcesmonitord**

To disable **mmchnode -ces-disable ..**

**mmces node list** (watch for Node Groups and Node Flags)>> **mmchnode --ces-group group1 -N vm02,vm03**

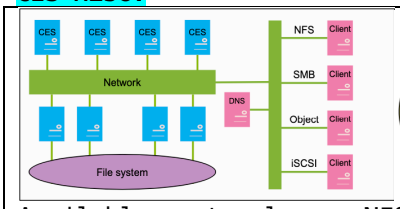
**mmces address add --ces-node vm02 --ces-ip 192.168.1.100** ( same for **.. vm03 --ces-ip 192.168.1.101** )

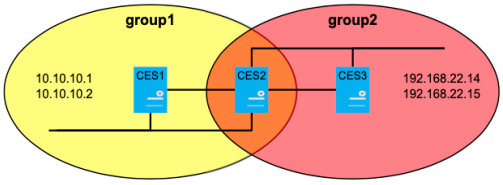
**mmces address change --ces-ip 192.168.1.100 --ces-group group1** ( same for **--ces-ip 192.168.1.101 ..**)

**mmces node list || mmces address list || mmces service list**

**mmlscluster --ces (shows even-coverage**, default) >> **mmces address policy balanced-load** (we set this)

**CES Misc:**




Available protocols are NFS, SAMBA, S3. Protocols are enabled individually.
All CES nodes must have same architecture and should not be NSD server.
**spectrumscale node add <node-name> -p** >> **spectrumscale install**

cesSharedRoot directory is monitored by **mmcesmonitord.**
If a directory is not available in mmces node list shows no shared root and a failover is triggered.

You cannot have static IP, ces nodes only takes alias IP (unlike cnfs).

Even-coverage is default, we set balanced-load. Other options are node-affinity (for node with more cpu)
Suspend is used for maintenance and it fails over IP, Stop does not failover IP and not persistent.

**Troubleshooting:**

**mmhealth cluster show || mmces state show -a**

```
[root@vm02 ~]# mmces state cluster NFS
NODE        COMPONENT  STATE    EVENTS
vm02        NFS        TIPS     ccr_quorum_nodes_ok,gpfs_up,gpfs_maxfilestocache_small,servic
e_running,quorum_up,gpfs_pagepool_small,csm_resync_forced,node_resumed,nfs_sensors_not_conf
igured,nfs_dbus_ok,service_disabled
vm03        NFS        TIPS     service_running,quorum_up,gpfs_up,no_longwaiters_found,servic
e_running,nfs_dbus_ok,nfs_sensors_not_configured,service_disabled
```

**mmces node list** (Look at **Node flags = None/ Suspended/Failed/no-shared-root/network-down/starting-up**)

**None** = is healthy

**Suspended** = suspended by user for maintenance. IP fails over and state is persistent on reboot.

**Failed** = failover happened and reflected in **mmlscluster --ces**

**Starting-up** = ces node is starting

**network-down** = you know it!!

```
Node Number   Node Name   Node Groups   Node Flags
------------- ----------- ------------- ----------
2             rh94-21     group1        none
3             rh94-22     group1        none
```

**mmces address list**

```
Address     Node      Ces Group   Attributes
----------- --------- ----------- ----------
10.0.0.85   rh94-21   group1      none
10.0.0.86   rh94-22   group1      none
```

**mmces service list** (no CES services enabled ?) || **mmces state show [NFS|SMB] -a**

**/var/adm/ras/mmfs.log.latest || /var/adm/ras/mmcesmonitor.log || mmces log level 4**

**Maintenance:**

**mmces move -ces-ip <...> || mmces remove -ces-ip <...>**

**mmces service start|stop NFS (or SMB) -N vm03 || mmces node suspend (resume) -N vm03**

## CES NFS GANESHA

**Cluster A:**

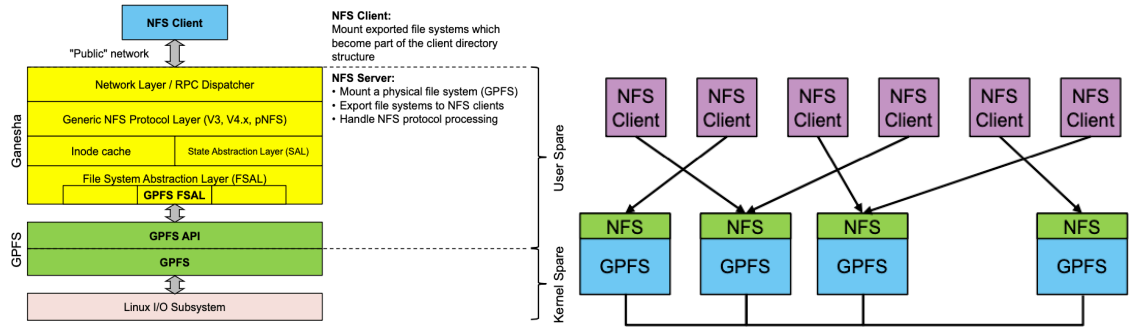
**mmces service enable [disable] NFS** >> **mmces service list**
**mmnfs config change MINOR_VERSIONS=1** (to enable v4.1, default is 4.0)

```
# before any NFS exports are defined, file access method user authentication must be changed
# 2 different authentication service can be configured, one for NFS and SMB and other for S3
# mmuserauth simply provides infor it needs to contact the external service
# Bellow tells the NFS that user ID and gid mapping is handled outside the scope of the cluster
mmuserauth service create --data-access-method file --type userdefined >> mmuserauth service list

mmcrfileset fs1 cesnfs --inode-space new >> mmlinkfileset fs1 cesnfs -J /gpfs/cesnfs

mmnfs export add /gpfs/cesnfs --client "*(ACCESS_TYPE=RW,SQUASH=no_root_squash)"
mmnfs export remove /path_to_folder (to remove)

mmces service list -a || showmount -e vm02    # To verify exported path
mmnfs export list || mmnfs export list --nfsdefs /gpfs/cesnfs
```

**Cluster B:**

**mkdir /mnt/cesnfs >> mount cesip:/gpfs/cesnfs /mnt/cesnfs**

## CES NFS GANESHA Misc:




Monitored by **mmcesmonitord** >> Customized version of pNFS >> Userspace implementation
More flexible than kernel space NFS, making code modifications easy.
Easy to control resources >> easy to upgrade (no reboot required)
Managing large memory allocation is easy >> Encountering problem does not crash OS kernel

Do not do **systemctl stop nfs-ganesha** >> **mmces service stop -N <nodename>**
**mmces service start** retrieves the config from CCR and stores locally on ces nodes.
Files are stord in **/var/mmfs/ces/ces-config** dir
(gpfs.ganesha**.exportfs.conf**, **log.conf**, **main.conf**, **nfsd.conf**, **statdargs.conf**)

**Delegation** is a part of NFSv4 protocol >> allows the client to locally service operations such as OPEN, CLOSE, LOCK, READ without intermediate action from server. The client is guaranteed certain semantics with respect to sharing the file with other clients.

**mmnfs export list**

```
Path         Delegations  Clients
-----------  -----------  -------------
/gpfs/cesnfs  NONE         *
/gpfs/exp1    NONE         *
/gpfs/exp1    NONE         192.168.1.100
```

**mmnfs export list --nfsdefs /gpfs/fs1/cesnfs**  # long listing

```
Path            Delegations  Clients  Access_Type  Protocols  Transports  Squash         Anonymous_uid  Anonymous_gid  SecType  PrivilegedPort  DefaultDelegations  Manage_Gids  NFS_Commit  Pseudo Path
--------------  -----------  -------  -----------  ---------  ----------  -------------  -------------  -------------  -------  --------------  ------------------  -----------  ----------  ---------------
/gpfs/fs1/cesnfs  NONE         *        RW           3,4        TCP         NO_ROOT_SQUASH  -2             -2             SYS      FALSE           NONE                FALSE        FALSE       /gpfs/fs1/cesnfs
```

**mmnfs export change /gpfs/exp2 --nfsadd "192.168.10.32(ACCESS_TYPE=RW)"**  # To add new client access
**mmnfs export add /gpfs/exp3 --client "*(ACCESS_TYPE=RO)"**  # To add new export
**mmnfs export list --nfsdefs /gpfs/exp2**

```
Path       ... Clients       Access_Type Protocols Transports Squash      ...
---------- ... ------------- ----------- --------- ---------- ----------- ...
/gpfs/exp2 ... *             RO          3,4       TCP        ROOT_SQUASH ...  0
/gpfs/exp2 ... 192.168.10.32 RW          3,4       TCP        ROOT_SQUASH ...  1
```

**mmnfs export change /gpfs/exp2 --nfschange "192.168.10.32(SQUASH=no_root_squash)" --nfsposition 0**
Previously 192.168.10.32 had RO with root_squash and now it has RW with no_root_squash (first match)

**CES SAMBA (Context: Cluster A is CES cluster, Cluster B plain Client)**

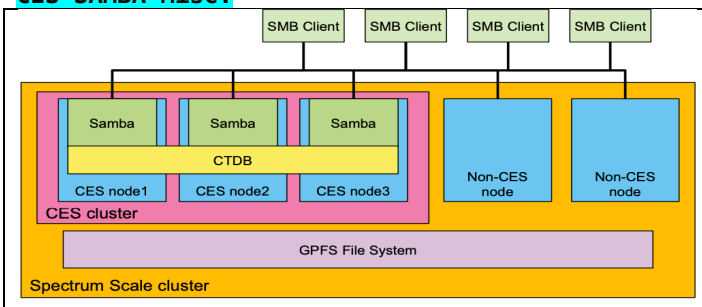
**Cluster A (CES nodes: vm02, vm03)**

```
mmuserauth service create --data-access-method file --type userdefined
mmces service enable SMB
(starts gpfs-smb, gpfs-ctdb process)

mmdsh -N cesnodes id cesuser
smbpasswd -a cesuser (one node only as it commits to CTDB)

mmcrfileset fs1 cessmb --inode-space new
mmlinkfileset fs1 cessmb -J /gpfs/cessmb

mkdir /gpfs/cessmb/share1

chown cesuser.cesuser /gpfs/cessmb/share1

mmsmb export add SMBCES /gpfs/cessmb/share1

mmces service list -a
mmuserauth service list
mmsmb export list
```

**Windows PC**

```
net use Z: \cesip\SMBCES
(provide cesuser credential)

echo hello > Z:\testfile1.txt

dir Z:

(Bingo!)
```

**CES SAMBA Misc:**




Max of 16 ces nodes when SMB is configured.
A customized version of CTDB (cluster aware trivial database) is configured by GPFS internally.
CTDB spans multiple physical hosts, it stores persistent and non-persistent data.

persistent data are: registry, share ACLS, id mapping, group info, machine passwd
non-persistent data are open files, share mode, locks, session info etc.

Files are under **/var/lib/ctdb** on ces nodes.

Like ces-nfs, you must configure file access method. We used LDAP for external ID mapping.

**mmsmb export list** >> Now mount the SMB share using net use -Z: \\ces.test.net\SMBCES

| export | path | browseable | guest ok | server smb encrypt |
|---|---|---|---|---|
| SMBCES | /gpfs/fs1/share1 | yes | no | auto |

**CES OBJECT Protocol (Context: Cluster A is CES cluster, Cluster B plain Client)**

| **Cluster A (CES nodes: vm02, vm03)** | |
|---|---|

**CES OBJECT Protocol Misc:**

**Commands to access objstorage: swift, curl, s3curl**
**swift post mycontainer**
**swift upload mycontainer myfile**

**AFM DR (Context: Cluster A is Primary, Cluster B is Secondary)**

| **Cluster A** | **Cluster B** |
|---|---|
| **mmcrfileset fs1 primary1 -p afmMode=primary --inode-space new -p afmTarget=nfs://vm04/gpfs/home2**<br>(note the afmPrimaryId) | **mmcrfileset fs2 secondary1 -p afmMode=secondary --inode-space new -p afmPrimaryId=$PID** |
| **mmchnode --gateway -N vm03** | **mmlinkfileset fs2 secondary1 -J /gpfs/home2** |
| | **/etc/exports:**<br>/gpfs/home2 *(no_root_squash,rw,fsid=123) |
| | **systemctl restart nfs-config; systemctl restart nfs**<br>**exportfs** |
| **mmlinkfileset fs1 primary1 -J /gpfs/primary1**<br>(creates psnap0-rpo-C0A801036050DF15-1) | |
| **mmafmctl fs1 getstate -j primary1** | |
| **echo "test1" > /gpfs/primary1/test1; mmafmctl fs1 getstate -j primary1** | **ls -l /gpfs/home2**<br>(file is there) |
| | **systemctl stop nfs**<br>**mmafmctl fs2 failoverToSecondary -j secondary1** |
| **Failover test:**<br>**mmunlinkfileset fs1 primary1** | **echo "test3" > /gpfs/home2/test3** |
| **mmafmctl fs1 getstate -j primary1**<br>**mmlinkfileset fs1 primary1 -J /gpfs/primary1** | **systemctl start nfs** |
| **Failback:**<br>**mmafmctl fs1 failbackToPrimary -j primary1 -start** | |
| **mmafmctl fs1 applyUpdates -j primary1**<br>(a couple of times, like rsync'ing the delta) | |
| **mmafmctl fs1 failbackToPrimary -j primary1 --stop** | |
| | **mmunlinkfileset fs2 secondary1 -f** |
| | **mmchfileset fs2 secondary1 -p afmMode=secondary -p afmPrimaryId=$PID** |
| | **mmlinkfileset fs2 secondary1 -J /gpfs/home2** |
| | **systemctl restart nfs-config; systemctl restart nfs** |
| **echo "test4" > /gpfs/primary1/test4** | **ls -l /gpfs/home2** |
| **mmafmctl fs1 getstate -j primary1** | |