Een 3 Node GlusterFS cluster op CentOS 7

GlusterFS als oplossing om je storage redundant te maken. Het idee is dat als er een disk, een netwerkkaart een voeding een processor, of alles aan 1 server kapot gaat moet de boel blijven draaien.

Begin met een kaal ingerichte CentOS 7 server met bij voorkeur een apart storage LAN. over een netwerk team.

needs re-write to striped-replicated
Work in progress

diskindeling

De zogenaamde “bricks” (een aangewezen directory in een filesysteem waarvan vervolgens de ruimte van het onderliggende blockdevice van wordt gebruikt) waar een gluster volume uit bestaan moeten thin provisioned zijn vanwege een eigenaardigheid van glusterfs waar we mee te maken hebben.

We willen namelijk niet gewoon 3 kopieën draaien op alle servers, dit zou jammer zijn van de diskruimte (slechts 1/3 bruikbaar). We willen 2 kopieën op de drie servers hebben. Echter geldt bij glusterfs dat het aantal bricks een veelvoud moet zijn van het aantal kopieën.

Hier is wel een soort oplossing voornamelijk de arbitrator brick. Hierop wordt alleen file metadata opgeslagen (ongeveer 4kb per file) die bij een split brain situatie wordt gebruikt om te beslissen welke kopie de juiste is.

Een probleem is dat een glustervolume alleen maar zo groot wordt als de kleinste brick. En de arbitrator brick wordt in deze gewoon als brick gezien. dus weer zonde van de ruimte.

De oplossing is thin provisioning met overprovisioning, aangezien we zeker zijn dat de arbitrator brick nooit die volledige ruimte in beslag zal nemen.

Dit betekent ook dat we een schijf altijd in drieën moeten delen om de schijf symmetrisch over de servers te verdelen.

Hoeveel overprovisioning precies is dus wat vaag, maar uit mijn ervaring met een multipurpose nas kan ik ongeveer schatten dat je rekening moet houden met 1/2 gb meta data per tb storage.

dus bijvoorbeeld voor een 4Tb schijf moeten we rekening houden met 2Gb aan arbitrator ruimte, dus 4000Gb-2Gb=3998Gb ruimte over voor werkelijke data, delen door 2 geeft een partitie grootte van 1999Gb. Daar dus 3 van, thin provisioned, twee waarvan werkelijk gevuld kunnen worden met data tot 1999Gb en één waarvan we hopen dat de data nooit groter zal worden dan 2Gb.

Nu gaan we de thin provisioned logical volumes aanmaken. (In mijn voorbeeld heb ik een 4 Gb disk gebruikt ipv een 4 Tb disk)

Vind de disks die we voor het gluster file systeem gaan gebruiken(op alle servers)

```
lsblk
```

```
[root@server1 ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0              11:0    1 1024M  0 rom
vda             252:0    0    9G  0 disk
├─vda1          252:1    0    1G  0 part /boot
└─vda2          252:2    0    8G  0 part
  ├─centos-root 253:0    0  7,1G  0 lvm  /
  └─centos-swap 253:1    0  924M  0 lvm  [SWAP]
vdb             252:16   0  3,7G  0 disk
[root@server1 ~]#
```

we gaan de laatste (nog niet ingedeelde) gebruiken.

In dit geval dus /dev/vdb. Maak vervolgens een partitie aan door: `parted -a optimal /dev/vdb`

```
mklabel gpt
mkpart primary 0% 100%
quit
```

nu zie je bij `lsblk` de partitie ook staan:

```
vdb             252:16   0  3,7G  0 disk
└─vdb1          252:17   0  3,7G  0 part
```

Vervolgens maken we op die partitie een volumegroup met logical volumes:
`pvcreate /dev/vdb1`
`vgcreate glusterVG0 /dev/vdb1`

Bij thin provisioning maak je eerst een thin pool logical volume aan waarbinnen vervolgens de thin logical volumes worden aangemaakt die voor glusterfs worden gemaakt.

`lvcreate -l 100%FREE -T glusterVG0/thin_pool`

Nu kunnen de thin logical volumes worden aangemaakt voor de 2 replicated bricks en de arbitrator bricks. hoeveel ruimte heeft de thin pool:
`lvdisplay glusterVG0/thin_pool`

```
  --- Logical volume ---
  LV Name                thin_pool
  VG Name                glusterVG0
  LV UUID                3u7VPa-omd2-ka05-Ex32-AaFa-Kt0D-tBt9Bj
  LV Write Access        read/write
  LV Creation host, time server2.auriel.nl, 2018-12-23 15:18:19 +0100
  LV Pool metadata       thin_pool_tmeta
  LV Pool data           thin_pool_tdata
  LV Status              available
  # open                 0
  LV Size                <3,69 GiB
  Allocated pool data    0,00%
  Allocated metadata     11,13%
```

```
  Current LE             944
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:4
```

dus 994 LE, dat is ongeveer 3900kb per LE. Stel dat we 10 test files gaan neerzetten moet er rekening worden gehouden met ±40kb aan arbitrator data, om het hele verhaal even te houden gaan we dus 942 LE door 2 delen dus 471 LE per thin logical volume≈1791Mb (er is dus 2LE≈7800kb voor de arbitrator brick, ruimschoots voldoende)

```
lvcreate -V 1791M -T glusterVG0/thin_pool -n gluster_replica0
```

```
  Rounding up size to full physical extent 1,75 GiB
  Logical volume "gluster_replica0" created.
```

```
lvcreate -V 1791M -T glusterVG0/thin_pool -n gluster_replica1
```

```
  Rounding up size to full physical extent 1,75 GiB
  Logical volume "gluster_replica1" created.
```

```
lvcreate -V 1791M -T glusterVG0/thin_pool -n gluster_arbitrator0
```

```
  Rounding up size to full physical extent 1,75 GiB
  WARNING: Sum of all thin volume sizes (5,25 GiB) exceeds the size of thin
pool glusterVG0/thin_pool and the size of whole volume group (<3,70 GiB).
  WARNING: You have not turned on protection against thin pools running out
of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to
trigger automatic extension of thin pools before they get full.
  Logical volume "gluster_arbitrator0" created.
```

Hier krijgen we de waarschuwing dat we overprovisioned zijn, maar de logical volume wordt wel aangemaakt.

vervolgens een xfs filesysteem op al die logical volumes:

```
mkfs -t xfs /dev/mapper/glusterVG0-gluster_replica0
mkfs -t xfs /dev/mapper/glusterVG0-gluster_replica1
mkfs -t xfs /dev/mapper/glusterVG0-gluster_arbitrator0
```

het filesysteem mounten.

maak de mountpoints:

```
mkdir -p /GLUSTER0_DATA/brick0
mkdir -p /GLUSTER0_DATA/brick1
mkdir -p /GLUSTER0_DATA/arbitrator0
```

maak de mount `vi /etc/fstab` en voeg toe:

```
/dev/mapper/glusterVG0-gluster_replica0      /GLUSTER0_DATA/brick0
```

```
xfs   defaults   0 0
/dev/mapper/glusterVG0-gluster_replica1      /GLUSTER0_DATA/brick1
xfs   defaults   0 0
/dev/mapper/glusterVG0-gluster_arbitrator0   /GLUSTER0_DATA/arbitrator0
xfs   defaults   0 0
```

en mount : `mount -a`

en maak de volume directory aan waar glusterfs uiteindelijk mee gaat werken:

```
mkdir /GLUSTER0_DATA/brick0/gv1
mkdir /GLUSTER0_DATA/brick1/gv1
mkdir /GLUSTER0_DATA/arbitrator0/gv1
```

GlusterFS server installatie

repo

installeer de nodige repositories:
~~yum install epel-release~~
`yum install centos-release-gluster`

name resolving

Aangezien de storage meestal onderin de stack zit en de DNS als een VM op die stack draait is het niet echt verstandig om voor de name resolving van het cluster afhankelijk te zijn van een DNS. Daarom verdient het aanbeveling de hostnames van de nodes in de hosts file op te nemen:

`vi /etc/hosts`

en voeg toe:

```
192.168.120.21  server1.voorbeeld.nl    server1
192.168.120.22  server2.voorbeeld.nl    server2
192.168.120.23  server3.voorbeeld.nl    server3
```

software

install de gluster server software:
`yum install glusterfs-server`

start de glusterfs software nu en in de toekomst:
```
systemctl enable glusterd
systemctl start glusterd
```

firewall

Er zijn een aantal standaard poorten die GlusterFS gebruikt, maar met elke nieuwe brick kiest GlusterFS een nieuwe poort erbij. Het is, ook wat betreft netwerk load aan te bevelen een appart netwerk voor het storage verkeer te gebruiken.

door apparte NIC's te gebruiken voor storage verkeer is het voldoende om onderin de netwerk config file:
`vi /etc/sysconfig/network-scripts/ifcfg-<juste_device>`

de zone te veranderen:

```
ZONE=trusted
```

start een Trusted pool

Door een probe wordt het cluster geinitieerd.

van server1: `gluster peer probe server2`

van server2: `gluster peer probe server1`

Na het eerste pair moet een nieuw lid van het cluster eerst vanaf een geverifieerd lid van het cluster ge-probed worden:

van server2: `gluster peer probe server3`

nu kan je controleren welke nodes in de gluster pool draaien:
`gluster pool list` (bijvoorbeeld op server3)

```
[root@server3 ~]#  gluster pool list
UUID                    Hostname                State
c4db0a33-a877-44fd-91a6-fa93dfe0514a    server2.voorbeeld.nl    Connected
af00ee80-c7ec-4f91-bc31-b0bccc1e33df    server1.voorbeeld.nl    Connected
4a257d8a-633b-467c-b44e-ec82f73a4df9    localhost               Connected
```

Define en start een GlusterFS volume

Nu kan je een GlusterFS volume aanmaken, we hadden al een aantal directory aangemaakt op een gemount logical volume, een zogenaamde “brick” . In dit voorbeeld
`/GLUSTER0_DATA/brick1/gv1/` etc..

De naamgeving van een brick directory is freeform, en het gebruik van LVM is ook niet perse noodzakelijk. Voor zover ik kan nagaan zou je zelfs de brick kunnen definiëren als een rauw mountpoint, dus zonder de extra directory daarin.

```
gluster volume create gv1 replica 2 arbiter 1
server1:/GLUSTER0_DATA/brick0/gv1/ server2:/GLUSTER0_DATA/brick0/gv1/
```

```
server3:/GLUSTER0_DATA/arbitrator0/gv1/ server1:/GLUSTER0_DATA/brick1/gv1/
server3:/GLUSTER0_DATA/brick0/gv1/ server2:/GLUSTER0_DATA/arbitrator0/gv1/
server2:/GLUSTER0_DATA/brick1/gv1/ server3:/GLUSTER0_DATA/brick1/gv1/
server1:/GLUSTER0_DATA/arbitrator0/gv1/
```

```
volume create: gv1: success: please start the volume to access data
```

Het volgende is nodig om te zorgen dat op tijd een andere brick gebruikt wordt als een file een volle brick zou gaan veroorzaken. Zonder deze settings krijg je errors in de trand van “no space left on device” als er een brick vol is (ook al is er nog ruimte in andere bricks)

```
gluster volume set gv1 cluster.min-free-disk 4KB
```

De waarde is wel belangrijk, want pas als de min-fre-disk berijkt is zal een file naar een andere brick geschreven worden. Dus deze waarde moet gezet worden op de grootte van de te verwachten files.

Bit of a bummer i know.

En start het volume:

```
gluster volume start gv1
```

```
gluster volume start gv1
```

controleren:
```
gluster volume info gv1
```

```
Volume Name: gv1
Type: Distributed-Replicate
Volume ID: ea51a568-6f47-47b3-a29d-9ec2edabc04f
Status: Started
Snapshot Count: 0
Number of Bricks: 3 x (2 + 1) = 9
Transport-type: tcp
Bricks:
Brick1: server1:/GLUSTER0_DATA/brick0/gv1
Brick2: server2:/GLUSTER0_DATA/brick0/gv1
Brick3: server3:/GLUSTER0_DATA/arbitrator0/gv1 (arbiter)
Brick4: server1:/GLUSTER0_DATA/brick1/gv1
Brick5: server3:/GLUSTER0_DATA/brick0/gv1
Brick6: server2:/GLUSTER0_DATA/arbitrator0/gv1 (arbiter)
Brick7: server2:/GLUSTER0_DATA/brick1/gv1
Brick8: server3:/GLUSTER0_DATA/brick1/gv1
Brick9: server1:/GLUSTER0_DATA/arbitrator0/gv1 (arbiter)
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

GlusterFS client koppelen

client software instaleren

Als eerste moet de fuse module in de kernel geladen worden:

`modprobe fuse`

controleren: `dmesg | grep -i fuse`

```
[ 5647.281780] fuse init (API version 7.22)
```

Vervolgens de client voor de betreffende client (voor zover er niet in de vervolg stack een native driver zit, zoals in openshift en oVirt)

CentOS7

eerst de repository koppelen:
`yum install centos-release-gluster`

dan de client software installeren:
`yum install fuse fuse-libs openib libibverbs glusterfs glusterfs-fuse`

Ubuntu

`apt-get install glusterfs-client`

Gluster volume mounten

maak een mountpoint:
`mkdir /glustervolume1`

mount met de hand:
`mount -t glusterfs server1:/gv1 /glustervolume1`

controleren : `df` en daar staat dan tussen:

```
server1:/gv1              5474304  154968   5319336   3% /glustervolume1
```

en maak een permanente entry in de fstab: `vi /etc/fstab`

en voeg toe:

Expanding with another server

replace a faulty disk

expand with an extra disk

troubleshoot

verwijder een volume

Soms maak je in het begin een fout en wil je even opnieuw beginnen.

verwijder het gluster volume:
```
gluster volume stop gv1
gluster volume delete gv1
```

geef de brick directory weer vrij:
```
setfattr -x trusted.glusterfs.volume-id /GLUSTER_DATA/brick1/gv1/
setfattr -x trusted.gfid /GLUSTER_DATA/brick1/gv1
rm -rf /GLUSTER_DATA/brick1/gv1/.glusterfs
```

Bronnen

https://wiki.centos.org/SpecialInterestGroup/Storage/gluster-Quickstart

https://docs.gluster.org/en/latest/Administrator%20Guide/Split%20brain%20and%20ways%20to%20deal%20with%20it/

https://linuxconfig.org/configuration-of-high-availability-storage-server-using-glusterfs?google_comment_id=z121dnohyk3hifn1q22txvi5xpmlztpvo&jsn_setmobile=no#h2-1-peer-probe

https://support.rackspace.com/how-to/getting-started-with-glusterfs-considerations-and-installation/

https://sysadmins.co.za/setup-a-3-node-replicated-storage-volume-with-glusterfs/

https://serverfault.com/questions/803308/gluster-file-system-used-space-seen-by-df

https://gluster.readthedocs.io/en/latest/Administrator%20Guide/arbiter-volumes-and-quorum/

https://joejulian.name/post/replacing-a-glusterfs-server-best-practice/

https://access.redhat.com/solutions/276483

https://access.redhat.com/solutions/1298163