GlusterFS - despersed Type Volume 생성 실습

GFS Volume Features – Dispersed Volume

GFS Volume type
이해 편의상 비슷한 HW RAID 레벨의 특성을 같이 기재하였다
- Distribute Volume (RAID 0)
- Replicate Volume (RAID 1)
- Distribute + Replicate Volume (RAID 0+1)
- **Dispersed Volume (RAID 5,6)**
- Dispersed + Replicate Volume (RAID 50 , 60)

Dispersed Volume?
HW RAID Level 에서의 비슷한 유형을 찾는다면 RAID 5 와 6의 중간 단계에 있다.
최소 3개 이상의 Brick 부터 구성이 가능하며
볼륨 구성 시 복구 가능한 분산 패리티의 수량을 명시할 수 있어(Redundancy)
RAID 5 보다 복구 유연성이 좋다고 볼 수 있다.

Replicate 보다 가용량을 확보하며 최소한의 안전장치를 확보한다고 볼 수 있다.

Distribute 과의 차이점?
같은 분산의 의미이지만 Distribute 는 순수한 파일을 각 Brick 에 분배하는 것이고
Dispersed 는 erasure coding 방식 (패리티와 사실상 유사) 의
인코딩된 전용 데이터를 만들어 Brick 에 분산한다.

Create Command
e.x)
```
$ gluster volume create NEW-VOLNAME [disperse-data COUNT] [redundancy COUNT] [transport tcp | rdma | tcp,rdma] NEW-BRICK...
```

disperse-data COUNT : 실제 사용되는 용량에 들어가는 Brick 의 수

redundancy COUNT : 데이터 분산 패리티의 수량 지정 가능.
해당 수량만큼 Brick 을 가진 물리 디스크가 Failed 되어도 클러스터는 유지된다.

RAID 5가 1개의 패리티. RAID 6 이 2개의 패리티만이 가능한 것보다 유연하다.
이 값만 지정해 줄 시 자동적으로 disperse-data COUNT 가 계산된다.

RMW(Read-Modify-Write) 방식으로 데이터를 기록하기 때문에
효율적인 I/O 를 위한 Brick 비율이 존재한다
(e.x DISPERSE 6 REDUNDANCY 2)

GFS Volume Create

100GB * 3개의 Brick 을 Redundancy 1개 (RAID 5) 방식으로 구성

Peer Probe
Node01)

노드들을 우선 인식해야 한다. 마스터 자기 자신은 제외

```
gluster peer probe 211.239.151.197
gluster peer probe 211.239.151.198
gluster peer status
```

Volume Create
노드마다 100GB 씩 할당한 파티션을 묶어서 클러스터를 구성한다.

```
gluster volume create data disperse-data 2 redundancy 1 transport tcp 211.239.151.196:/gfs_node 211.239.151.197:/gfs_node 211.239.151.198:/gfs_node force

# check
volume create: data: success: please start the volume to access data
```

기본 상태 확인
```
gluster volume info
```

볼륨 시작. 처음 생성시 stop 되어 있으니 반드시 시작해야 함
```
gluster volume start
volume start: data: success
```

볼륨 확인
```
gluster volume status
```

```
Status of volume: data
Gluster process                                 TCP Port  RDMA
Port  Online  Pid
------------------------------------------------------------
----------------
Brick 211.239.151.196:/gfs_node                   49152     0
Y       65695
Brick 211.239.151.197:/gfs_node                   49152     0
Y       13261
Brick 211.239.151.198:/gfs_node                   49152     0
Y       13513
Self-heal Daemon on localhost                   N/A       N/A
Y       65712
Self-heal Daemon on 211.239.151.198             N/A       N/A
Y       13530
Self-heal Daemon on 211.239.151.197             N/A       N/A
Y       13278

Task Status of Volume data
------------------------------------------------------------
----------------
```

마운트 및 용량 확인
(All Node)
brick 2개 만큼의 용량 확보 되었고 1개는 redundancy(패리티) 상태이다.

```
mount -t glusterfs gfs-node01:/data /mnt
df -h /mnt
Filesystem        Size  Used Avail Use% Mounted on
gfs-node01:/data  200G  2.1G  198G   2% /mnt
```

fstab 자동 마운트 등록
클러스터이기 때문에 어떤 노드의 엔드포인트를 지정하여도 마운트에 문제는 없다.

```
echo "gfs-node01:data /mnt glusterfs defaults,_netdev 0 0" >>
/etc/fstab
```

Node Fail-over TEST

(노드 다운)
** 1 node down

Node 하나를 Poweroff 시 약 1분 정도의 딜레이가 걸리지만
클러스터는 정상 작동한다
gluster status volume

** 2 node down
# Redundancy 가 1개이므로 노드 2개가 다운되면 클러스터 볼륨은 연결 실패하게 된다.
[root@node01 mnt]# ll
ls: cannot open directory .: Transport endpoint is not connected

(노드 업)
** 1 node up
노드 2개가 살아났을 경우 남아있던 노드는 자동으로 다시 연결된다.
up 노드는 리마운트 해주어야 한다.

[root@node01 mnt]# ll
ls: cannot open directory .: Transport endpoint is not connected
[root@node01 mnt]# ll
total 0
-rw-r--r--. 1 root root 0 Jan 13 10:42 a
-rw-r--r--. 1 root root 0 Jan 13 10:44 b
-rw-r--r--. 1 root root 0 Jan 13 10:44 c

[root@node01 mnt]# gluster volume status
Status of volume: data
Gluster process   TCP Port RDMA Port Online Pid

Brick 211.239.151.196:/gfs_node   49152   0   Y   67046
Brick 211.239.151.197:/gfs_node   49153   0   Y   3551
Self-heal Daemon on localhost   N/A   N/A   Y   67063
Self-heal Daemon on 211.239.151.197   N/A   N/A   Y   3108

Task Status of Volume data

** 2 node up
정상화 된 것으로 보인다.
다수의 다량 데이터 검증은 필요하다

** 3 (all) node up
부팅 순서에 따라 자동 마운트는 안 되지만 스토리지 볼륨은 정상 online 된다.

Volume Expansion (ADD-BRICK)

node03 에 100GB Brick 3EA 추가할 것이다..
Dispersed 타입 볼륨은 3(1패리티) 비율의 Brick 으로 구성하였고
6(2 패리티) 조합이 밸런스가 권장된다.
- 기존 GFS Volume 에 증설
- 기존 브릭 교체 실습

```
# Check now
[root@node02 ~]# gluster volume info | egrep -iE "brick|t
ype"
Type: Disperse
Number of Bricks: 1 x (2 + 1) = 3
Transport-type: tcp
Bricks:
Brick1: 211.239.151.196:/gfs_node
Brick2: 211.239.151.197:/gfs_node
Brick3: 211.239.151.198:/gfs_node

# 추가할 BRICK 파티셔닝 ( sdc , sdd , sde 3개 구성하자 )
fdisk /dev/sdc <<EOF
n
p




w
EOF
mkfs.xfs -i size=512 /dev/sdc1
mkdir -p /add_brick1
mount /dev/sdc1 /add_brick1

echo "/dev/sdc1        /add_brick1        xfs      defaults
0 0" >> /etc/fstab
```

```
# check
[root@node03 ~]# df -h | grep brick
/dev/sdc1               100G   33M  100G   1% /add_bric
1
/dev/sdd1               100G   33M  100G   1% /add_bric
2
/dev/sde1               100G   33M  100G   1% /add_bric
3

# Volume ADD issue
```
3개 미만의 브릭을 추가하려고 하면 에러가 발생한다
```
[root@node03 ~]# gluster volume add-brick data gfs-
node03:/add_brick1
volume add-brick: failed: Incorrect number of bricks supplied
1 with count 3
```

새 Peer (물리적 노드) 가 아닌 같은 서버에 Brick 다수 중첩될 경우 경고 발생
```
[root@node03 ~]# gluster volume add-brick data gfs-
node03:/add_brick1    gfs-node03:/add_brick2    gfs-
node03:/add_brick
3
volume add-brick: failed: Multiple bricks of a disperse volume
are present on the same server. This setup is not optimal.
Bricks should be on different nodes to have best fault
tolerant configuration. Use 'force' at the end of the command
if you want to override this behavior.

force 옵션을 요구함
volume add-brick: success

# Volume brick check
```
체크 시 6개의 브릭 중 2개가 패리티로 설정된 균형비율 확인
```
[root@node03 ~]# gluster volume info | egrep -iE "brick"
Number of Bricks: 2 x (2 + 1) = 6
Bricks:
Brick1: 211.239.151.196:/gfs_node
Brick2: 211.239.151.197:/gfs_node
Brick3: 211.239.151.198:/gfs_node
Brick4: gfs-node03:/add_brick1
Brick5: gfs-node03:/add_brick2
Brick6: gfs-node03:/add_brick3
```

```
# Volume Size Check
brick 추가시 volume size 도 라이브하게 변경되었다.
[root@node03 ~]# df -h | grep mnt
gfs-node01:data          400G  4.2G  396G   2% /mnt


# Rebalance
추가된 Brick 으로의 고른 데이터 분산을 위해 사용
gluster volume rebalance data start
volume rebalance: data: success: Rebalance on data has been
started successfully. Use rebalance status command to check
status of the rebalance process.
ID: 9782e187-6cda-4e2b-aae9-0b78746d69fa


# check
1) status 에서 내용을 확인할 수 있다
Task Status of Volume data
------------------------------------------------------------
----------------
Task                 : Rebalance
ID                   : 9782e187-6cda-4e2b-aae9-0b78746d69fa
Status               : completed

2) 리밸런스 후 추가된 brick 에 패리티가 일부 분산된 내용을 확인할 수 있었다
[root@node03 add_brick3]# ll ( 리밸런스 전 )
total 0
[root@node03 add_brick3]# ll ( 리밸런스 후 )
total 8
-rw-r--r--. 2 root root 512 Jan 13 14:57 d
[root@node03 add_brick3]#
```

Setting Up Volumes – Gluster Docs

[AWS] EC2 직렬 콘솔에서 응급모드 진입 설정 – Amazon Linux 2

리눅스에서 패스워드 리셋이나 마운트 에러 등의 이슈로 응급 모드로 진입하여 작업해야 할 때가 있는데 AWS는 기본적으로 원격 관리이기 때문에 콘솔 화면이 보이지 않습니다.

Nitro Hypervisor 를 지원하는 T3 급 이상의 인스턴스는 AWS Console 에서 직접 시리얼 콘솔을 접근할 수 있는 옵션이 추가가 되었는데

GRUB Timeout 설정 시간이 짧아서 응급모드 진입이 어려운 이슈가 있습니다.
Amazon Linux 2 OS 의 GRUB 타임아웃 시간을 늘려서 확인할 수 있도록 해 봅니다.

마운트 실패로 인한 응급 모드의 큰 틀은 다음의 2가지로 분류가 됩니다.

1) `GRUB TIMEOUT` 변경
2) 응급 모드 진입 후 `mount` 옵션에 `nofail` 추가 시, 실패해도 진행 가능

IAM 접근 최소 조건

`# IAM`이 `Administrator Group` 이라면 상관 없으나, 디테일 권한 관리될 경우 다음의 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetSerialConsoleAccessStatus",
                "ec2:EnableSerialConsoleAccess",
                "ec2:DisableSerialConsoleAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

GRUB TIMEOUT 값 수정하기

root 권한에서 Default GRUB 의 값을 수정 및 추가합니다.
Console Serial 은 root 패스워드가 콘솔 로그인 가능해야 합니다.
아마존의 타임아웃 권장값은 1입니다만 넉넉하게 10(초)로 변경합니다.

```
sed -i 's/GRUB_TIMEOUT=0/GRUB_TIMEOUT=10/g' /etc/default/grub
sed  -i  's/GRUB_TERMINAL="ec2-console"/GRUB_TERMINAL="console
serial"/g' /etc/default/grub
 echo  -e  GRUB_SERIAL_COMMAND=\"serial  --speed=115200\"  >>
/etc/default/grub

# check
cat /etc/default/grub

GRUB_CMDLINE_LINUX_DEFAULT="console=tty0
console=ttyS0,115200n8   net.ifnames=0   biosdevname=0
nvme_core.io_timeout=4294967295    rd.emergency=poweroff
rd.shell=0"
GRUB_TIMEOUT=10
GRUB_DISABLE_RECOVERY="true"
GRUB_TERMINAL="console serial"
GRUB_X86_USE_32BIT="true"
GRUB_SERIAL_COMMAND="serial --speed=115200"

# 실제 GRUB 에 적용합니다
grub2-mkconfig -o /boot/grub2/grub.cfg
```

EC2 Reboot 후 직렬 콘솔 확인

10초 지연 후 지나가므로 e 키를 바로 눌러 응급모드 진입

응급모드 진입

해당 위치에 single 구문을 추가 후 Ctrl + x 로 부팅

FSTAB 에서 nofail 옵션 사용

추가 마운트 옵션에 `nofail` 을 넣으면 부팅 시 실패해도 정상 진행됩니다

AWS 공식 링크(이미지 별)

[AWS] Cloud-init SSH RootPassword 세팅하기

AWS 의 Amazon-Linux OS 로 생성한 이미지 등의 경우 기본적으로 Password 방식의 로그인이 허용되어 있지 않으며, ssh 설정에서 주석처리를 해제하여도 이미지 백업 후 복원 시 Cloudinit 에 의해 다시 주석이 잠기는 보안성이 존재합니다.

이는 다량의 작업을 해야 할 때 번거로움으로 작용할 수 있는데
패스워드 로그인을 허용하는 방법은 2가지로 구분할 수 있습니다.

1. 이미지에서 인스턴스 생성 시에 Advance의 User-data 에서 리눅스 스크립트로 부팅 시 root 잠금 해제
2. 기존 이미지에서 Cloudinit Config 의 설정 변경

이미지에서 인스턴스 생성 시 다음의 스크립트를 추가한다.

```
# sshd Permitrootlogin
sed -i 's/#PermitRootLogin yes/PermitRootLogin yes/g' /etc/ssh/sshd_config
sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/g' /etc/ssh/sshd_config

# Key root account permit
sed -i 's/^.*10" //g' /root/ssh/authorized_keys

# set root password
echo "P@ssw0rd" | passwd root --stdin
```

```
# sshd restart
systemctl restart sshd
```

이미지에서 인스턴스 생성 시 User-data 추가

생성 후 직렬 콘솔을 통해 root password 로그인이 가능하다.

이미지 생성 시에 리셋되지 않도록 하려면 Cloudinit 설정을 변경한다

```
disable_root : true --> false ( 0 )
ssh_pwauth:  false --> true ( 1 )
```