Microsoft

Red Hat Storage / GlusterFS ON AZURE

Khaled Elbedri

khaled.elbedri@microsoft.com

TSP GBB OSS

9/29/2017

CONTENTS

LAB OBJECTIVES AND PRE-REQUISITES

This lab describes the steps necessary to deploy highly available GlusterFs (Red Hat Storage) environment on Azure using a basic two-node CENTOS 7 configuration. GlusterFS is the upstream project of Red Hat storage and is considered as a test bed incubator.

You don't need a Red Hat subscription to perform the lab instructions. But you will need a valid Azure account. Create your free azure account (https://azure.microsoft.com/en-us/free/) today.

The Lab covers:

- GlusterFS Architecture and Installation
- Creating Highly Available (Replicated) GlusterFS Volume on Azure
- Creating a distributed GlusterFS Volume on Azure
- Connecting from Linux/Windows clients with various protocols
- Extending GlusterFS Volumes without downtime
- Configuring and exploring the graphical console managing GlusterFS clusters

INTRODUCTION TO REDHAT STORAGE

Red Hat Gluster FS Storage is designed to provide a flexible file services layer for users and applications in a way that can be easily scaled to adjust to storage demanding workloads. Deployment flexibility is a key strength of Red Hat Gluster FS Storage. GlusterFS can be deployed to virtual or physical servers on-premises environments, in private clouds, and public clouds. Microsoft and Red Hat have signed a partnership that includes support to run Red Hat Storage on Microsoft Azure.

Azure offers multiple cloud solutions either as infrastructure-as-a-service (IaaS) or platform-as-a-service (PaaS). For GlusterFS, we will leverage Azure IAAS capabilities to build logical containers (virtual machines) backed by software defined storage (Azure disks). Then, we will deploy and configure the shared filesystem bricks. Azure provides network services like DNS and DHCP, which makes managing the infrastructure like managing a physical deployment.

EXERCISE1: PLANNING THE DEPLOYMENT ON AZURE

Terminology:

- Gluster Storage server: The virtual machine which hosts the file system in which data will be stored.
- Gluster Storage client: The virtual machine which mounts the GlusterFS shared volumes.
- Brick: The brick is a disk partition with XFS file system that has been assigned to a Volume.
- GlusterFS Volume: The logical collection of bricks.

Lab configuration:

We will use Azure virtual machines to create a two nodes GlusterFS cluster. A Linux and Windows clients will be used to demonstrate mounting and consuming software defined storage exported by the GlusterFS cluster

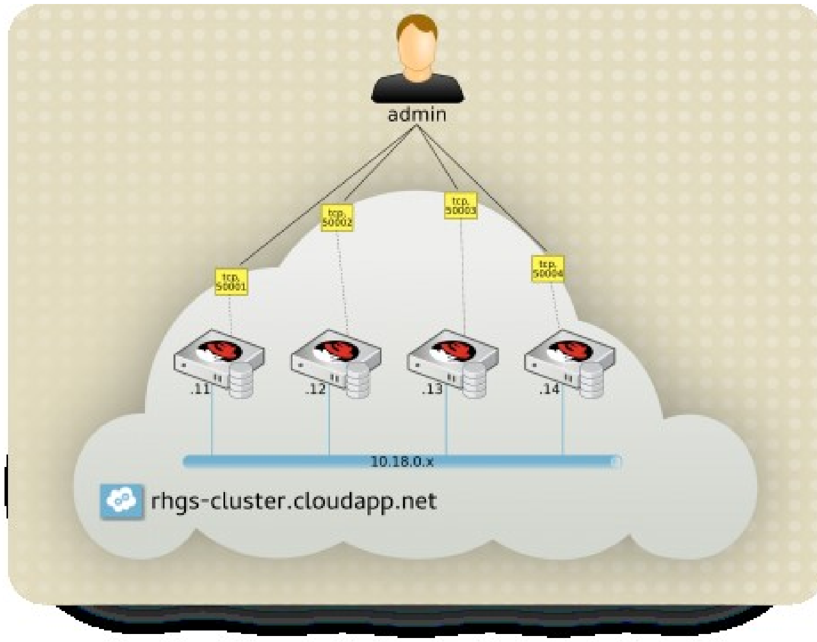


Fig1: GlusterFS simplified architecture

Servers	Node1	Node2
	CentOS 7.3+	CentOS 7.3+
Clients	Node3	Windows-client
	CentOS 7.3+	Windows Server 2008 x64

To add another layer of resiliency to our architecture, we will provision GlusterFS cluster nodes into an Azure availability set (https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-infrastructure-availability-sets-guidelines#availability-sets).

An Azure availability set provides a level of fault tolerance to the instances it holds, protecting against system failure or planned outages. This is achieved by ensuring instances within the same availability set are deployed across different fault and upgrade domains within an Azure datacenter. By using availability sets in the replication design, incidents within the Azure infrastructure cannot affect all members of a replica set simultaneously.

1. Login to Azure portal and start a new Bash Cloud shell session.

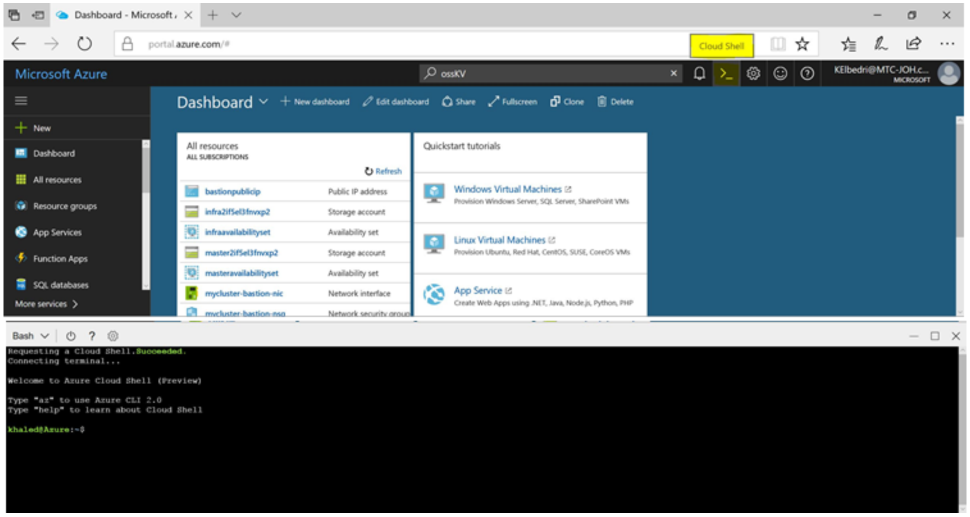


Fig2: Azure Cloud Shell

2. Create an *ssh* keypair with a blank passphrase

```
[Azure]$ ssh-keygen
```

3. Create a new Azure resource group *glusterfsRG* in your preferred region

```
[Azure]$ az group create -n glusterfsRG -l northeurope
{
  "id": "/subscriptions/f3a5dfdb-e863-40d9-b23c-
752b886c0260/resourceGroups/glusterfsRG",
  "location": "northeurope",
  "managedBy": null,
  "name": "glusterfsRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

4. Create a new availability set *glusteras*

```
[Azure]$ az vm availability-set create -g glusterfsRG -n
glusteras -l northeurope
```

```
{
  "id": "/subscriptions/f3a5dfdb-e863-40d9-b23c-752b886c0260/resourceGroups/glusterfsRG/providers/Microsoft.Compute/availabilitySets/glusteras",
  "location": "northeurope",
  "name": "glusteras",
  "platformFaultDomainCount": 2,
  "platformUpdateDomainCount": 5,
  "resourceGroup": "glusterfsRG",
  "sku": {
    "capacity": null,
    "name": "Aligned",
    "tier": null
  },
  "statuses": null,
  "tags": {},
  "type": "Microsoft.Compute/availabilitySets",
  "virtualMachines": []
}
```


5. Create a new Linux CentOS virtual machine and replace CHANGEME by 3 random alpha numeric string, of your choice. The string will insure uniqueness across the multiple candidates performing the lab.
   The following second command creates a new vm named *node1* in the availability set *glusteras* with a virtual network name *gluster-vnet*, a subnet name *gluster-snet,* a virtual address network prefix *10.0.0.0/16,* a virtual network subnet address prefix *10.0.1.0/24,* a public IP name *node1-pub,* and a public IP domain name *node1glusterXXX*. The username *azureuser* and the public key *id_rsa.pub* will be used to login to the newly created node. Start, by exploring the various options available with the vm creation command, then create the node.

```
[Azure]$ az vm create --help


[Azure]$ az vm create -g glusterfsRG -n node1 -l northeurope --vnet-name gluster-vnet --subnet gluster-snet --vnet-address-prefix 10.0.0.0/16 --subnet-address-prefix 10.0.1.0/24 --public-ip-address node1-pub --public-ip-address-dns-name node1glusterCHANGEME --image centos --admin-username azureuser --ssh-key-value .ssh/id_rsa.pub --availability-set glusteras
```

```
{
  "fqdns": "node1gluster001.northeurope.cloudapp.azure.com",
  "id": "/subscriptions/f3a5dfdb-e863-40d9-b23c-752b886c0260/resourceGroups/glusterfsRG/providers/Microsoft.Compute/virtualMachines/node1",
  "location": "northeurope",
  "macAddress": "00-0D-3A-B2-A1-E8",
  "powerState": "VM running",
  "privateIpAddress": "10.0.1.4",
  "publicIpAddress": "13.74.250.242",
  "resourceGroup": "glusterfsRG",
  "zones": ""
}
```

6. Create similar virtual machine *node2* in the same vnet/subnet as *node1*. Replace CHANGEME by the same random string, you used in the previous command.

```
[Azure]$ az vm create -g glusterfsRG -n node2 -l northeurope --vnet-name gluster-vnet --subnet gluster-snet --public-ip-address node2-pub --public-ip-address-dns-name node2glusterCHANGEME --image centos --admin-username azureuser --ssh-key-value .ssh/id_rsa.pub --availability-set glusteras
{
  "fqdns": "node2gluster001.northeurope.cloudapp.azure.com",
  "id": "/subscriptions/f3a5dfdb-e863-40d9-b23c-752b886c0260/resourceGroups/glusterfsRG/providers/Microsoft.Compute/virtualMachines/node2",
  "location": "northeurope",
  "macAddress": "00-0D-3A-B5-05-9B",
  "powerState": "VM running",
  "privateIpAddress": "10.0.1.5",
  "publicIpAddress": "40.113.94.232",
  "resourceGroup": "glusterfsRG",
  "zones": ""
}
```

7. Show the two, previously, provisioned nodes

```
[Azure]$ az vm list -g glusterfsRG | grep node

[Azure]$ az vm show -g glusterfsRG -n node1
```

8. Note the public IP addresses of your newly created vms

```
[Azure]$ az vm list-ip-addresses -g GLUSTERFSRG -o table
VirtualMachine    PublicIPAddresses    PrivateIPAddresses
----------------  -------------------  -----------------
node1             13.74.250.242        10.0.1.4
node2             40.113.94.232        10.0.1.5
```

9. Ssh to node1 using the private key you created in step-2 and install available updates.

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP

[node1]$ sudo yum update -y
```

10. Use your preferred editor to create and add a new yum repository */etc/yum.repos.d/glusterfs-epel.repo* as following:

```
[node1]$ nano /etc/yum.repos.d/glusterfs-epel.repo

[glusterfs-epel]
name=GlusterFS is a clustered file-system capable of scaling to several petabytes.
baseurl=http://buildlogs.centos.org/centos/7/storage/x86_64/gluster-3.8/
enabled=1
skip_if_unavailable=1
gpgcheck=0
```

11. Install the latest *EPEL* repository from *fedoraproject.org* to resolve all dependencies needed later-on:

```
[node1]$ sudo yum -y install
http://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
```

12. Make sure both repositories are enabled by default:

```
[node1]$ sudo yum repolist
... output omitted ...
repo id              repo name                    status
base/7/x86_64        CentOS-7 - Base               9,007
epel/x86_64          Extra Packages for Enterprise Linux 7
- x86_64                         10,765
extras/7/x86_64      CentOS-7 - Extras              393
glusterfs-epel       GlusterFS is a clustered file-system
capable of scaling to several p    162
openlogic/7/x86_64 CentOS-7 - openlogic packages for
x86_64                                48
updates/7/x86_64     CentOS-7 - Updates          2,560
repolist: 22,935
```

13. Install *GlusterFS* Server and *Samba* packages

```
[node1]$ sudo yum install glusterfs-server samba -y
```

EXERCISE2: CREATING THE BRICKS

In the next steps, we will add two new disks to each GlusterFS cluster node. To create a big backend storage pool. We will stripe the disks into a RAID0 array. That configuration, allows higher IOPS.

For simplicity reasons, we will use disks with 10Gb capacity, only. The array on each node will then be used to create two GlusterFS bricks. The bricks will be used to create the GlusterFS volumes.

1. Exit node1 and attach 2 x 10 GB data disks to *node1* and *node2*

```
[node1]$ exit


[Azure]$ for n in {1..2}; do az vm disk attach --disk
disk1$n --new --size-gb 10 --resource-group
glusterfsRG --vm-name node1; done


[Azure]$ for n in {1..2}; do az vm disk attach --disk
disk2$n --new --size-gb 10 --resource-group
glusterfsRG --vm-name node2; done
```

2. Ssh again to node1, list the system's partition table and make sure you have 2 new disks (*/dev/sdc* and */dev/sdd*)

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP
[node1]$ sudo fdisk -l
... output omitted ...
Device Boot  Start   End      Blocks     Id  System
/dev/sda1 *  2048    62914559 31456256   83  Linux
... output omitted ...
/dev/sdb1    128     14678015 7338944    83  Linux
... output omitted ...
Disk /dev/sdc: 10.7 GB, 10737418240 bytes, 20971520
sectors
... output omitted ...
Disk /dev/sdd: 10.7 GB, 10737418240 bytes, 20971520
sectors
```

3. Combine the virtual disks with *mdadm* to allow the LUN to deliver IOPS beyond that of a single virtual disk. Use *mdadm* to combine disks to form a larger RAID0 disk.

```
[node1] $ sudo mdadm --create md0 --level=0 --
chunk=256K --raid-devices=2 /dev/sdc /dev/sdd

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md/md0 started.


[node1]$ sudo mdadm --examine --scan | sudo tee
/etc/mdadm.conf

ARRAY /dev/md/md0  metadata=1.2
UUID=f92d3a2d:2c14157b:5bc8ef77:27ca57b7
name=node1:md0
```

4. Create the file system (2 *bricks*) that will be used to create the *Glusterfs* volume. First we will convert the RAID device to a physical volume, then we create a volume group (a virtual big disk). Finally, we use the volume group as a backend to create logical volumes (virtual partitions).

```
[node1]$ sudo pvcreate --dataalignment 1024K /dev/md/md0

  Physical volume "/dev/md/md0" successfully created


[node1]$ sudo vgcreate --physicalextentsize 256K glustervg-data
/dev/md/md0

  Volume group "glustervg-data" successfully created


[node1]$ sudo vgs

  VG             #PV #LV #SN Attr   VSize  VFree

  glustervg-data   1   2   0 wz--n- 19.98g 9.98g


[node1]$ for n in {1..2}; do sudo lvcreate -L 5G -n brick$n
glustervg-data; done

  Logical volume "brick1" created.

  Logical volume "brick2" created.


[node1]$ sudo lvs
```

```
  LV    VG          Attr      LSize Pool Origin Data%  Meta%  Move
 Log Cpy%Sync Convert
  brick1 glustervg-data -wi-ao---- 5.00g
  brick2 glustervg-data -wi-ao---- 5.00g
```

5. Format the bricks with *XFS* file system:

```
[node1]$ for n in {1..2}; do sudo mkfs.xfs
/dev/glustervg-data/brick$n; done
```

6. Create mount points and mount XFS bricks:

```
[node1]$ sudo mkdir -p /bricks/brick{1,2}

[node1]$ for n in {1..2}; do sudo mount /dev/glustervg-
data/brick$n /bricks/brick$n; done
```

7. Add the following lines to */etc/fstab*:

```
[node1]$ sudo -s

[node1]# echo "/dev/glustervg-data/brick1
/bricks/brick1    xfs     defaults    0 0" >> /etc/fstab

[node1]# echo "/dev/glustervg-data/brick2
/bricks/brick2    xfs     defaults    0 0" >> /etc/fstab

[node1]# exit
```

8. Mount the created bricks and make sure they show as new file systems

```
[node1]$ sudo mount -a

[node1]$ sudo df -h
... output omitted ...
/dev/mapper/glustervg--data-brick1  5.0G   33M  5.0G
1% /bricks/brick1
```

```
/dev/mapper/glustervg--data-brick2  5.0G   33M  5.0G
1% /bricks/brick2
```

EXERCISE3: CONFIGURING STORAGE POOL

In this section, we will enable the *GlusterFS* cluster on *node1* and *node2*

1. Enable and start *glusterd* service on *node1*:

```
[node1]$ sudo systemctl enable glusterd
Created symlink from /etc/systemd/system/multi-
user.target.wants/glusterd.service to
/usr/lib/systemd/system/glusterd.service.

[node1]$ sudo systemctl start glusterd
[node1]$ exit
```

2. To avoid, repeating the same steps to prepare *node2,* we provide you with a ready to use bash script that automates the same previous commands we run on *node1* to be executed on *node2*. Ssh to *node2* (you can find the public IP address from step-8 Exercice-1), escalate to *root* privileges. Then copy, explore, make executable and run "*prepare-gluster-node.sh*".

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE2-PubIP

[node2]$ sudo -s

[node2]# wget https://raw.githubusercontent.com/Microsoft-
OpenSource-Labs/glusterfs-azure-lab/master/prepare-gluster-
node.sh

[node2]# less prepare-gluster-node.sh

[node2]# chmod +x prepare-gluster-node.sh

[node2]# ./ prepare-gluster-node.sh

[node2]# exit
```

3. Use *gluster* command to connect the two nodes and create a Trusted Pool (Storage Cluster). You don't have to run the same command on the other node

```
[node2]$ sudo gluster peer probe node1
peer probe: success.
```

4. Verify the cluster peer:

```
[node2]$ sudo gluster peer status
Number of Peers: 1

Hostname: node1
Uuid: 17de2959-20f5-4107-a33a-3b169ee8adbf
State: Peer in Cluster (Connected)
[node2]$ exit
```

EXERCISE4: HIGH AVAILABILITY GLUSTERFS VOLUMES

Once the bricks are in place, a *GlusterFS* volume can be created; the volume combines the capacity from each node. GlusterFS Volume works with Gluster File System which is a logical collection of XFS bricks. The following table shows dependencies between volume types and sizes, assuming 1G bricks:

GlusterFS Volume types	Volume space
Distributed (for maximum space)	1G + 1G = 2G
Replicated (for high availability)	1G + 1G = 1G
Striped (for large files)	1G + 1G = 2G
Distributed and Replicated	(1G+1G) + (1G+1G) = 2G
Distributed and Striped	(1G+1G) + (1G+1G) = 4G
Distributed, Replicated and Stripped	[(1G+1G)+(1G+1G)] + [(1G+1G)+(1G+1G)] = 4G

Table: GlusterFS volume types

The two most common volume types are *distributed* and *distributed-replicated*. A distributed volume has no fault-tolerance but has the maximum capacity. A distributed-replicated volume has node-level fault-tolerance but has reduced capacity. In the next section, we will configure two *GlusterFS volumes,* replicated *glustervol1* and distributed, *glustervol2*.

1. First, create two sub-directory mount points, */bricks/brick1/repvol* and */bricks/brick1/disvol* in both *node1* and *node2*

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP


[node1]$ sudo mkdir /bricks/brick1/repvol
/bricks/brick2/distvol && exit
```

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE2-PubIP

[node2]$ sudo mkdir /bricks/brick1/repvol
/bricks/brick2/distvol
```

2. Use the */bricks/brick1* XFS partition on both nodes to create a highly available replicated volume, *glustervol1*. You don't have to run the same command on *node1*:

```
[node2]$ sudo gluster volume create glustervol1 replica
2 transport tcp node1:/bricks/brick1/repvol
node2:/bricks/brick1/repvol
volume create: glustervol1: success: please start the
volume to access data

[node2]$ sudo gluster volume start glustervol1
volume start: glustervol1: success
```

3. Use the */bricks/brick2* XFS partition on both nodes to create a big distributed volume, *glustervol2*. You don't have to run the same command on *node2*:

```
[node2]$ sudo gluster volume create glustervol2
transport tcp node1:/bricks/brick2/distvol
node2:/bricks/brick2/distvol
volume create: glustervol2: success: please start the
volume to access data

[node2]$ sudo gluster volume start glustervol2
volume start: glustervol1: success
```

4. Verify the newly created GlusterFS Volumes:

```
[node2]$ sudo gluster volume info all
Volume Name: glustervol1
Type: Replicate
Volume ID: 6ce0b2e0-696a-4deb-8f3a-6b11dfd5ad85
```

```
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: node1:/bricks/brick1/repvol
Brick2: node2:/bricks/brick1/repvol
Options Reconfigured:
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on

Volume Name: glustervol2
Type: Distribute
Volume ID: 9b96e301-9aa7-47fc-a387-65c61e7d2bb6
Status: Started
Snapshot Count: 0
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: node1:/bricks/brick2/distvol
Brick2: node2:/bricks/brick2/distvol
Options Reconfigured:
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on
```

The following diagram, explain the logical architecture of the implemented solution, so far.

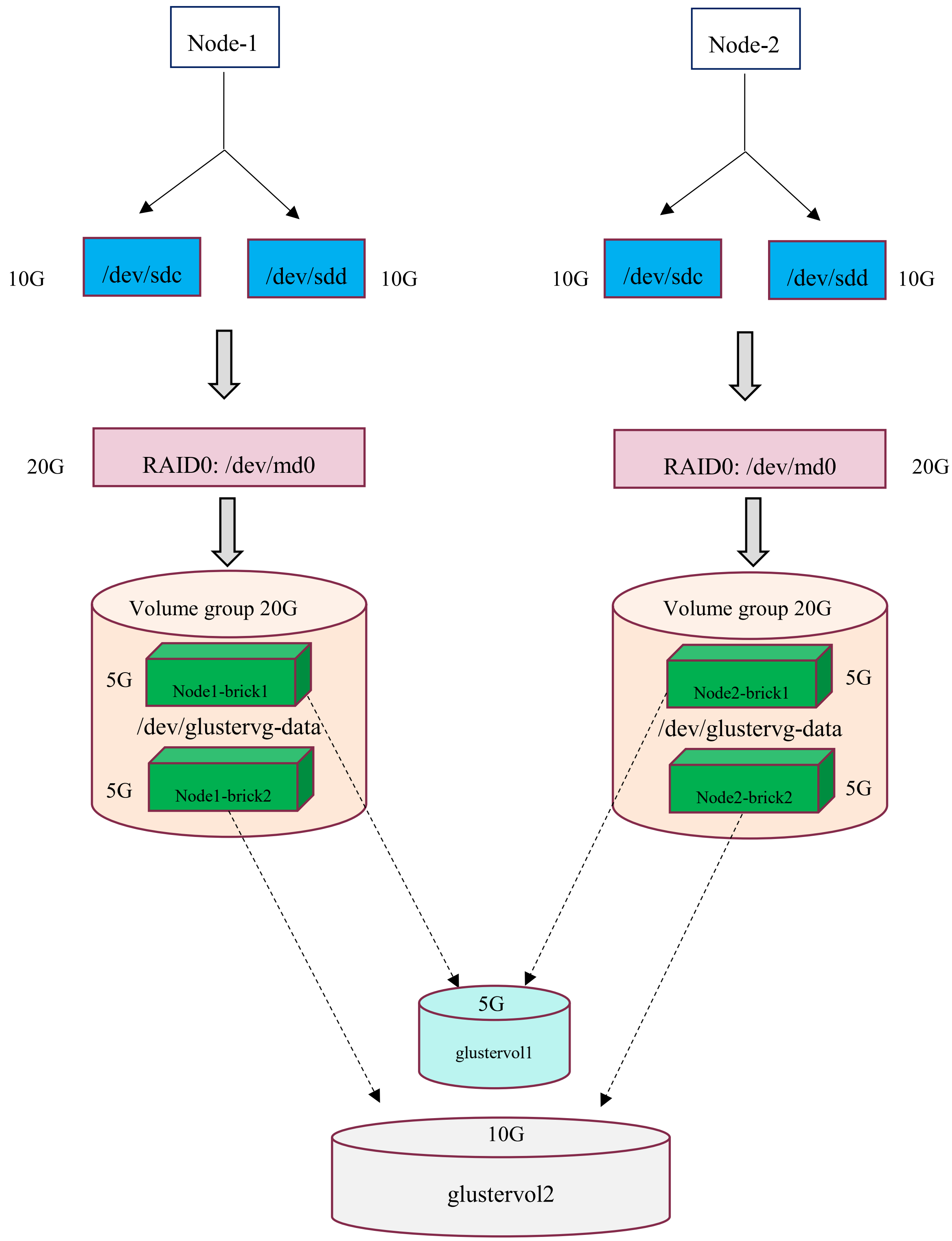


Fig3: Logical architecture

EXERCISE5: GLUSTERFS CLIENTS CONFIGURATION

Now that we have created the type GlusterFS volumes, we need to verify that the exported storage could be mounted by various operating systems. In a typical use case, we could have a cluster of multiple VMs sharing the exported storage as illustrated by the following figure. For instance, the cluster could be created by Azure scale sets. With such architecture, Red Hat storage / GlusterFS will provide highly available, persistent, elastic storage to be shared among the nodes.

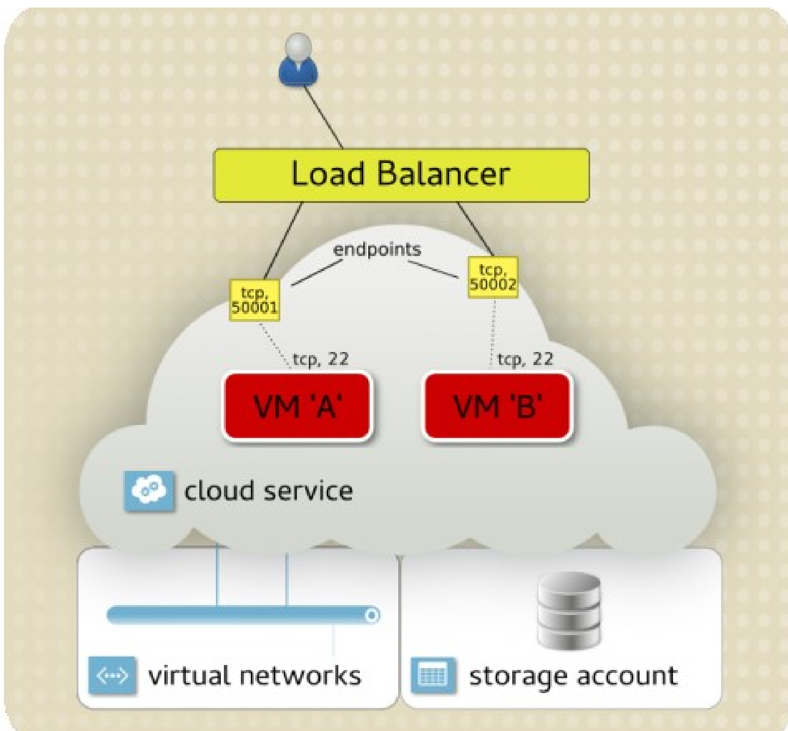


Fig4: GlusterFs as a backend to compute cluster

GlusterFS could be mounted on Linux systems using the native *glusterfs* client, or as an *NFS* or *samba* share. On windows, the filesystem could be exported with *samba* service and mounted as CIFS.

For simplicity reasons, we will deploy single Linux and Windows VMs. Then we will mount the created volumes on each of them.

Access from Linux via GlusterFS Native Client

1. Provision a Linux CentOS vm, *node3*. Replace CHANGEME by the string you used in the previous exercises.

```
[Azure]$ az vm create -g glusterfsRG -n node3 -l
northeurope --vnet-name gluster-vnet --subnet gluster-
snet --public-ip-address node3-pub --public-ip-address-
dns-name node3glusterCHANGEME --image centos --admin-
username azureuser --ssh-key-value .ssh/id_rsa.pub --
availability-set glusteras
```

2. Ssh into node3 and install glusterfs native client tools and some additional packages. All required packages are available by default in the CentOS 7 Base repository.

```
[Azure]$ az vm list-ip-addresses -g GLUSTERFSRG -o table
| grep node3

[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP

[node3]$ sudo yum install glusterfs-fuse attr nfs-utils
httpd -y
```

3. From azure portal, find node3 and associate *nic-node3* with a network security group that opens port 80 to the world.
4. Create a mount point and mount GlusterFS Volumes on node 3:

```
[node3]$ sudo mkdir -p /shared/big

[node3]$ sudo mount -t glusterfs node1:/glustervol1
/var/www/html

[node3]$ sudo mount -t glusterfs node1:/glustervol2
/shared/big
```

5. Report the size of the shared file systems and explain the difference in capacity:

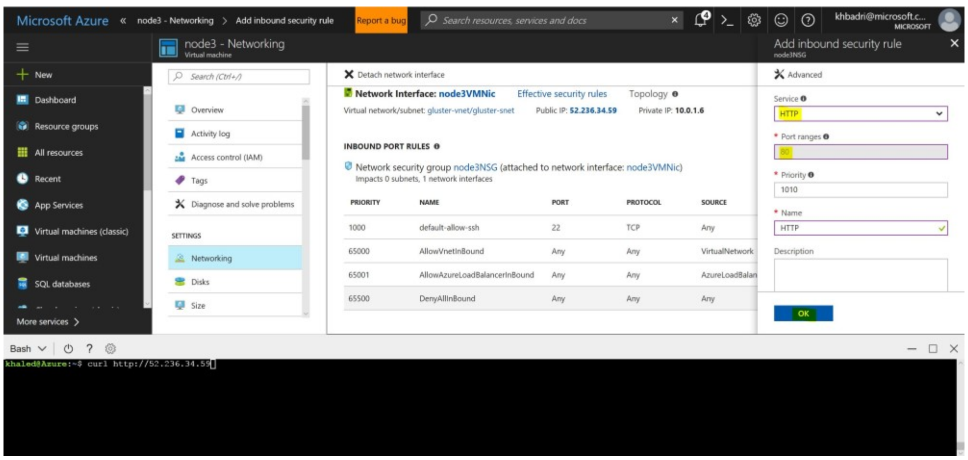
```
[node3]$ sudo df -h /var/www/html/ /shared/big
```

```
node1:/glustervol1  5.0G  33M  5.0G  1% /var/www/html
node1:/glustervol2   10G  66M   10G  1% /shared/big
```

6. Start Apache on *node3*

```
[node3]$ sudo systemctl start httpd && exit
```

7. Change the security groups to allow inbound http traffic on port 80 on node3



8. Point your web browser to the public IP of *node3* or use *curl* from the bash session in your local machine to confirm that the website on node3 is active.

```
[Azure]# curl http:// NODE3-PubIP
```

9. Copy some content to the shared volume

```
[Azure]# ssh -i .ssh/id_rsa azureuser@NODE3-PubIP
[node3]$ sudo cp /etc/passwd /shared/big && exit
```

10. Stop *node1*. Is the website still available? Can you list the contents of /share/big? Can you copy in some new contents? Can you explain what happened?

```
[Azure]$ az vm stop -g glusterfsRG -n node1
{
  "endTime": "2017-09-29T22:44:27.633160+00:00",
  "error": null,
  "name": "2ffe62db-83ac-49e4-90f0-425f76ce044f",
  "startTime": "2017-09-29T22:44:05.976484+00:00",
  "status": "Succeeded"
}
[Azure]$ curl://http:node3(pubIP)

[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP

[node3]$ sudo ls /shared/big

[node3]$ sudo cp /etc/shadow /shared/big/
cp: cannot create regular file '/shared/big/shadow':
Transport endpoint is not connected

[Azure]$ exit
```

11. Start node1 and wait for few seconds. Repeat the previous steps. Can you explain what just happened?

```
[Azure]$ az vm start -g glusterfsRG -n node1
[Azure]$ curl://http:node3(pubIP)
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP

[node3]$ sudo cp /etc/shadow /shared/big/
[node3]$ ls -l /shared/big/
total 3
-rw-r--r--. 1 root root 1573 Dec 28 17:15 passwd
----------. 1 root root  736 Dec 28 17:22 shadow
```

12. Stop *Apache* and unmount the shared file system.

```
[node3]$ sudo systemctl stop httpd

[node3]$ sudo umount /shared/big /var/www/html && exit
```

Access from Linux via GlusterFS via NFS

NB: GlusterFS NFS server only supports version 3 of NFS protocol.

13. Enable NFS access to glustervol2 and verify the volumes configuration.

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP

[node1]# sudo gluster volume set glustervol2 nfs.disable
off

[node1]# sudo gluster volume status
Status of volume: glustervol1
Gluster process          TCP Port  RDMA Port  Online  Pid
--------------------------------------------------------
Brick node1:/bricks/brick1/repvol 49154 0      Y    1120
Brick node2:/bricks/brick1/repvol 49156 0      Y    1108
NFS Server on localhost           2049  0      Y    1097
Self-heal Daemon on localhost     N/A   N/A    Y    1106
NFS Server on node1               2049  0      Y    1105
Self-heal Daemon on node1         N/A   N/A    Y    1111

Task Status of Volume glustervol1
--------------------------------------------------------
There are no active volume tasks

Status of volume: glustervol2
Gluster process          TCP Port  RDMA Port  Online  Pid
--------------------------------------------------------
Brick node1:/bricks/brick2/distvol 49155  0    Y    1127
Brick node2:/bricks/brick2/distvol 49157  0    Y    1120
```

```
NFS Server on localhost              2049    0     Y     1097
NFS Server on node1                  2049    0     Y     1105

Task Status of Volume glustervol2
------------------------------------------------------------------------------
There are no active volume tasks
```

14. Add the following line to */etc/nfsmount.conf* on both *node1* and *node2*. It is recommended to reboot all glusterfs nodes (*node1* and *node2*) before continuing.

```
[node1]$ sudo -s

[node1]# echo "Defaultvers=3" >> /etc/nfsmount.conf

[node1]# reboot && exit

[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE2-PubIP

[node2]$ sudo -s

[node2]# echo "Defaultvers=3" >> /etc/nfsmount.conf

[node2]# reboot && exit
```

15. Wait for a minute until *node1* and *node2* are up again, then mount GlusterFS Volumes via NFS:

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP

[node3]$ sudo mount -t nfs node1:/glustervol2
/shared/big

[node3]$ sudo mount

[node3]$ sudo df -h
```

Access from Windows/Linux machines via CIFS

16. Ssh to *node1* and *node2*. Install/update the *samba* required packages on both cluster nodes and start/enable Samba services:

PS: *node2* was pre-configured by the script *prepare-gluster-node.sh*

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP

[node1]$ sudo yum install samba samba-client samba-
common samba-vfs-glusterfs -y

[node1]$ sudo systemctl start smb.service

[node1]$ sudo systemctl enable smb.service

[node1]$ sudo systemctl start nmb.service

[node1]$ sudo systemctl enable nmb.service
```

Once a new GlusterFS Volume is created/started, it is added to the Samba configuration file, automatically as *gluster-<Volume_name>* file share.

17. Find the GlusterFS shares in */etc/samba/smb.conf*

```
[node1]$ sudo cat /etc/samba/smb.conf
... output omitted ...
 [gluster-glustervol1]
comment = For samba share of volume glustervol1
vfs objects = glusterfs
glusterfs:volume = glustervol1
glusterfs:logfile = /var/log/samba/glusterfs-
glustervol1.%M.log
glusterfs:loglevel = 7
path = /
read only = no
guest ok = yes
```

```
[gluster-glustervol2]
comment = For samba share of volume glustervol2
vfs objects = glusterfs
glusterfs:volume = glustervol2
glusterfs:logfile = /var/log/samba/glusterfs-
glustervol2.%M.log
glusterfs:loglevel = 7
path = /
read only = no
guest ok = yes
```

18. Use your preferred text editor to add a new parameter kernel **share modes = No** to the GlusterFS samba configuration on both *node1* and *node2*.

```
[node1]$ sudo -s

[node1]# nano /etc/samba/smb.conf
[gluster-glustervol2]
kernel share modes = No

[node1]# exit && exit

[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE2-PubIP

[node2]$ sudo -s

[node2]# nano /etc/samba/smb.conf
[gluster-glustervol2]
kernel share modes = No

[node2]$ exit && exit
```

19. Prepare the *glustervol2* GlusterFS Volume for Samba:

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP
```

```
[node1]$ sudo gluster volume set glustervol2 stat-
prefetch off
volume set: success

[node1]$ sudo gluster volume set glustervol2
server.allow-insecure on
volume set: success

[node1]$ sudo gluster volume set glustervol2
storage.batch-fsync-mode sudo lazy-open 0
volume set: success
```

20. Use your preferred editor to add the following line to */etc/glusterfs/glusterd.vol* before the line *#end-volume*, on *node1* and *node2*:

```
[node1]$ nano /etc/glusterfs/glusterd.vol
option rpc-auth-allow-insecure on
[node1]$ exit
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE2-PubIP
[node2]$ nano /etc/glusterfs/glusterd.vol
option rpc-auth-allow-insecure on
[node2]$ exit
```

21. Restart *glusterfs* service:

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP
[node1]$ sudo systemctl restart glusterd
```

22. Add a new samba user to *node1*:

```
[node1]$ sudo adduser sambauser

[node1]$ sudo smbpasswd -a sambauser
New SMB password:
Retype new SMB password:
```

```
Added user sambauser.
```

23. Restart Samba and turn SELinux to permissive mode on both nodes:

```
[node1]$ sudo systemctl restart smb.service

[node1]$ sudo systemctl restart nmb.service

[node1]$ sudo setenforce 0 && exit
```

24. Repeat steps 21 and 23 on *node2.*
25. On *node3*, mount GlusterFS Volume via CIFS (Samba) and verify the file system:

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP

[node3]$ yum install cifs-utils -y

[node3]$ mount -t cifs \\\\node1\\gluster-glustervol2
/mnt/ -o user=sambauser

[node3]$ sudo mount
\\10.0.1.5\gluster-glustervol2 on /share/big type cifs
(rw,relatime,vers=1.0,cache=strict,username=sambauser,do
main=NODE1,uid=0,noforceuid,gid=0,noforcegid,addr=10.0.1
.4,unix,posixpaths,serverino,acl,rsize=1048576,wsize=655
36,actimeo=1)

[node3]$ sudo df -h /mnt && exit
Filesystem                  Size Used Avail Use% Mounted on
\\node1\gluster-glustervol2 10G   66M   10G   1% /mnt
```

26. Use Azure portal to create a new Windows 2008 VM, in the same resource group *glusterfsRG*
27. Mount *glustervol2* on Windows by starting a new command interface, running and submitting the sambauser's password:

```
c:\>net use Z: \\node1\gluster-glustervol2
/user:sambauser

The command completed successfully.
```

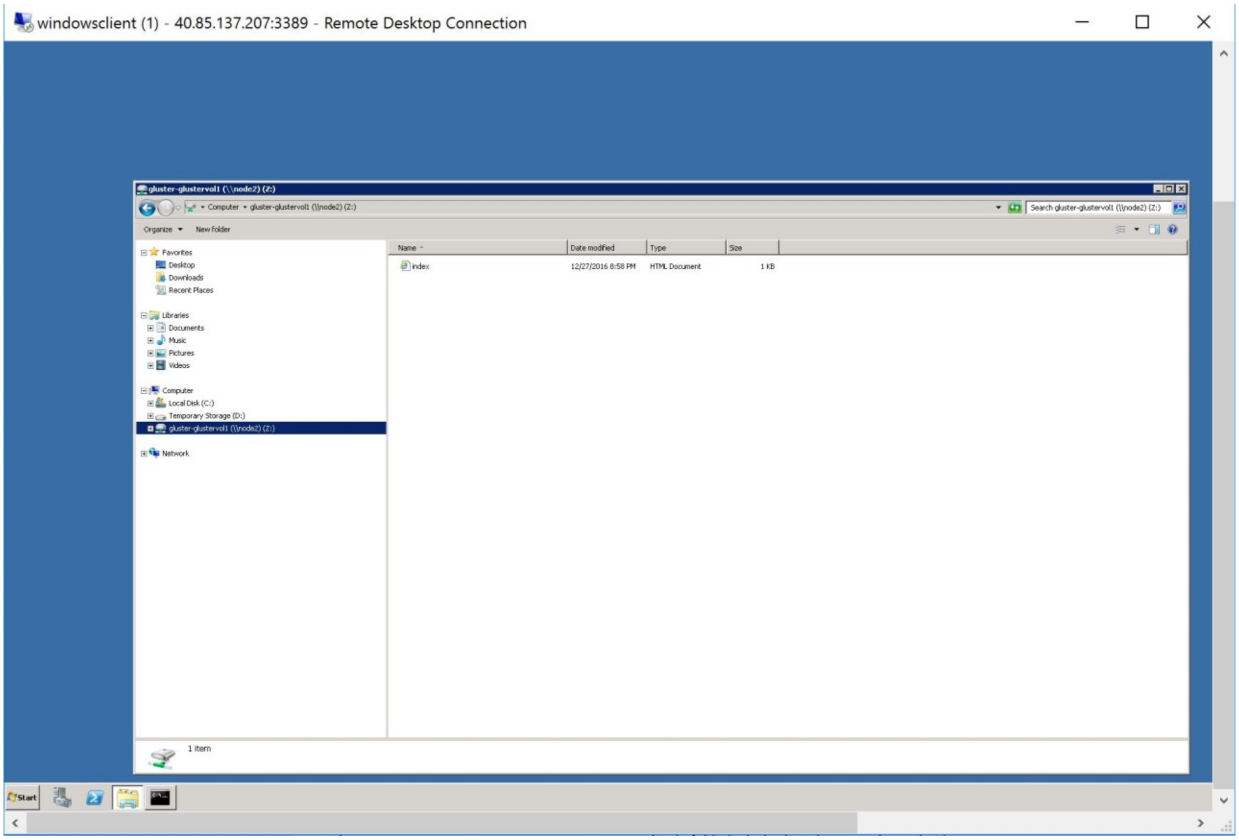


Fig5: Adding glusterfs share from windows client

EXERCISE6: EXTEND GLUSTERFS SYSTEM WITHOUT DOWNTIME

GlusterFS offers the option to extend the shared filesystem without down time. To do so, we need to add a number of bricks that is a multiple of the replica or stripe count. For example, to expand a distributed replicated volume with a replica count of 2, we need to add bricks in multiples of 2.

1. Show the volume parameters before the extension:

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP

[node1]$ sudo gluster volume info all
Volume Name: glustervol1
Type: Replicate
Volume ID: bf14d223-f8b7-43b3-8c6f-2cfc6cb40c93
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: gluster1.example.com:/bricks/brick1/repvol
Brick2: gluster2.example.com:/bricks/brick1/repvol
... output omitted ...
```

2. Now, we will leverage the free capacity in the volume group *glusterfsvg* from each node to create an additional brick *brick3* on *node1* and another one on *node2*.

```
[node1]$ sudo lvcreate -L 5G -n brick3 glustervg-data

[node1]$ sudo mkfs.xfs /dev/glustervg-data/brick3

[node1]$ sudo mkdir /bricks/brick3

[node1]$ sudo mount /dev/glustervg-data/brick3
/bricks/brick3/
```

```
[node1]$ sudo -s

[node1]# echo "/dev/glustervg-data/brick3
/bricks/brick3    xfs     defaults" >> /etc/fstab

[node1]# mount |grep brick3
/dev/mapper/vg_gluster-brick3 on /bricks/brick3 type xfs
(rw,relatime,seclabel,attr2,inode64,noquota)
```

3. Create a necessary the sub-directory mount point */bricks/brick3/distrepvol*.

```
[node1]# mkdir /bricks/brick3/distrepvol
```

4. Repeat steps 2 and 3 on node2.
5. Use the two XFS bricks, newly created, to extend the GlusterFS Volume without any downtime:

```
[node1]# gluster volume add-brick glustervol1
node1:/bricks/brick3/distrepvol
node2:/bricks/brick3/distrepvol force
volume add-brick: success
```

6. Verify the Volume:

```
[node1]# gluster volume info glustervol1
Volume Name: glustervol1
Type: Distributed-Replicate
Volume ID: bf14d223-f8b7-43b3-8c6f-2cfc6cb40c93
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: node1:/bricks/brick1/repvol
Brick2: node2:/bricks/brick1/repvol
```

```
Brick3: node1:/bricks/brick3/distrepvol
Brick4: node2:/bricks/brick3/distrepvol
Options Reconfigured:
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: off
[node1]# exit && exit
```

7. Mount and verify the newly extended volume.

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP


[node3]$ sudo mount -t glusterfs node1:/glustervol1
/var/www/html


[node3]$ df -h /var/www/html
Filesystem           Size   Used Avail Use% Mounted on
node1:/glustervol1   10G    66M   10G   1% /var/www/html
```

Now the Volume is extended with two bricks and became **Distributed-Replicate**.

EXERCISE7: INSTALLING AND CONFIGURING THE GRAPHICAL CONSOLE

The *oVirt* upstream project, provides a graphical management console that can be used to manage the GlusterFS cluster. Let's install it on *node3* and explore it.

1. Subscribe the server to the *oVirt* project yum repository and install *ovirt-engine* by running

```
[node3]$ sudo yum install
http://resources.ovirt.org/pub/yum-repo/ovirt-
release36.rpm -y


[node3]$ sudo yum install ovirt-engine -y
```

2. Once the installation is complete, set up *ovirt* with *gluster*. The installer will take you through a series of interactive questions. Accept the default values. But don't setup the firewall. When prompted for the application mode, choose *Gluster.* Ignore the warning about the RAM resources.

```
[node3]$ sudo engine-setup
......
   --== CONFIGURATION PREVIEW ==--


          Application mode                     : gluster
          Default SAN wipe after delete        : False
          Update Firewall                      : False
          Host FQDN                            : node3
          Engine database secured connection : False
          Engine database host                 : localhost
          Engine database user name            : engine
          Engine database name                 : engine
          Engine database port                 : 5432
          Engine database host name validation   : False
          Engine installation                    : True
```

```
          PKI organization                          : Test
          Configure local Engine database           : True
          Set application as default page           : True
          Configure Apache SSL                      : True
          Configure VMConsole Proxy                 : True
          Engine Host FQDN                          : node3
          Configure WebSocket Proxy                 : True
[node3]$ exit
```

3. Now, from your local host graphical environment, browse through the following URL “https://<node3-ip>/ovirt-engine. Accept the self-signed certificate. Provide the user name *admin* and the password you chose at setup.
4. Install *ovirt* service on *node1*

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE1-PubIP

[node1]$ sudo yum install centos-release-ovirt40
```

5. Configure *node1* to allow ssh root access. In a real scenario, this operation should be taken more carefully. Azure network security groups could be used to only allow *ssh* access from *node3*.

   Key based authentication is possible, but for demonstration purposes we will just use password authentication. Use your favorite text editor to find and modify the following parameters from the config file */etc/ssh/sshd_config*. This will enable *root* login on *node1*.

```
[node1]$ nano /etc/ssh/ssh_config
PermitRootLogin yes
PasswordAuthentication yes
```

6. Set a new login password for *root* and restart *sshd* service to apply the new configuration

```
[node1]$ sudo -s

[node1]# passwd

[node1]# systemctl restart sshd && exit && exit
```

7. Repeat steps 4, 5 and 6 on *node2*.

8. Add both nodes *node1* and *node2* as new hosts to the default cluster. Note, that *Ovirt* detects the previously created volumes and automatically shows them in the interface. Explore features like adding new/ importing existing cluster, creating/deleting volumes, adding/deleting bricks, set/reset volume options, optimize volume for virt store, rebalance, remove brick ...

   The following screenshots illustrate some of the functionalities of *Ovirt*:

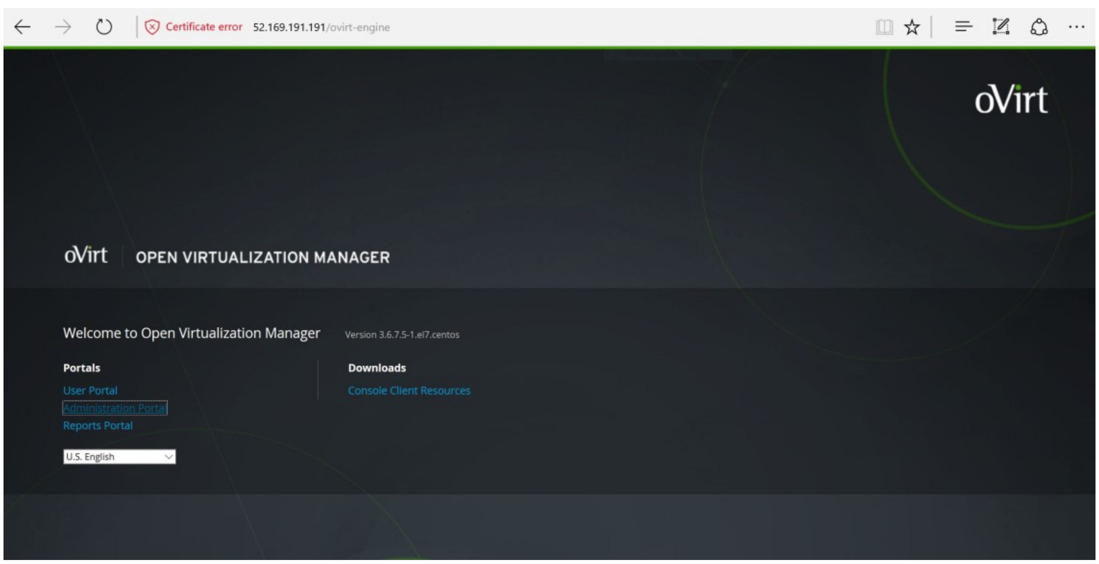


Fig6: oVirt login portal

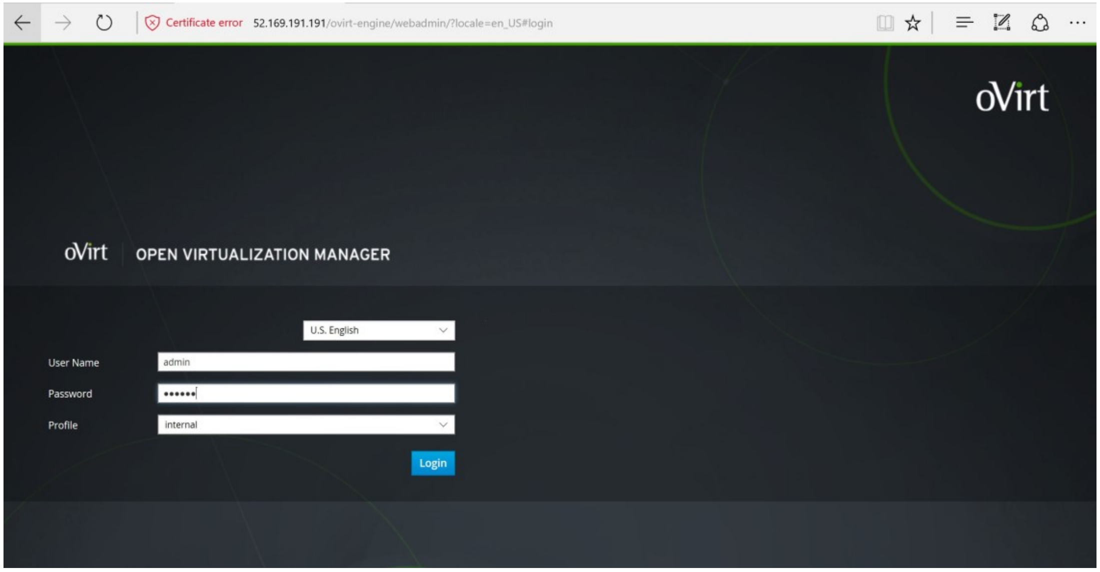


Fig7: Administration portal login page

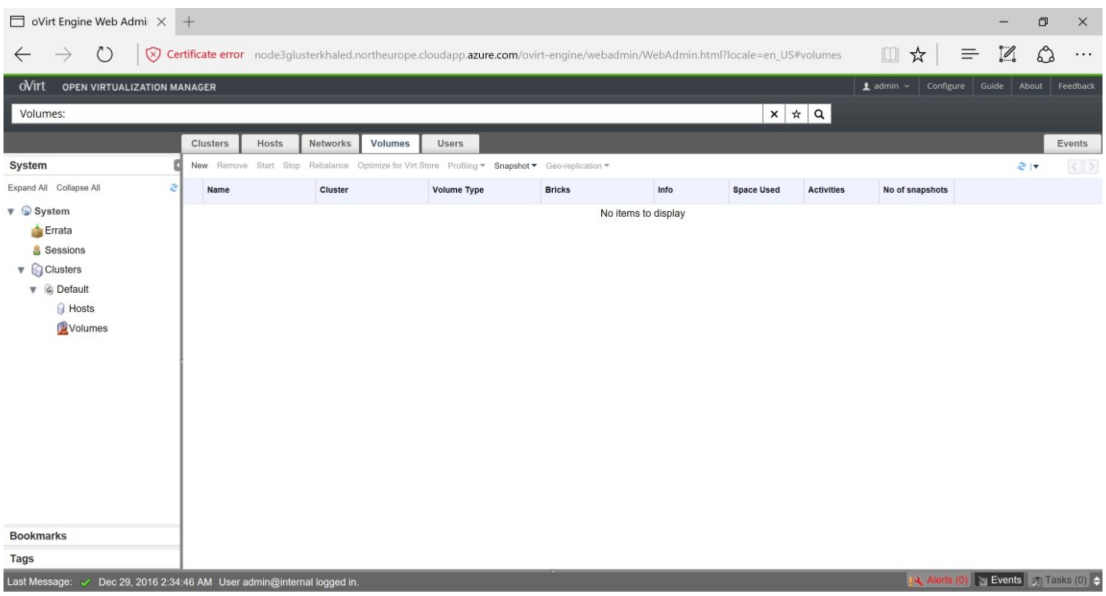


Fig8: Administration portal

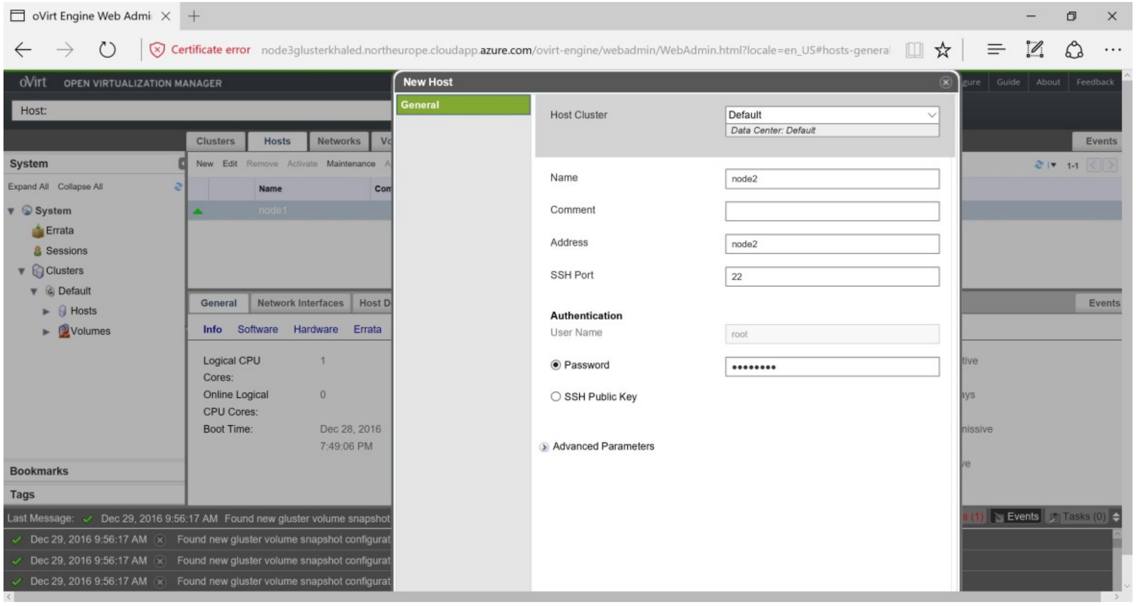


Fig9: Adding a node to glusterfs cluster

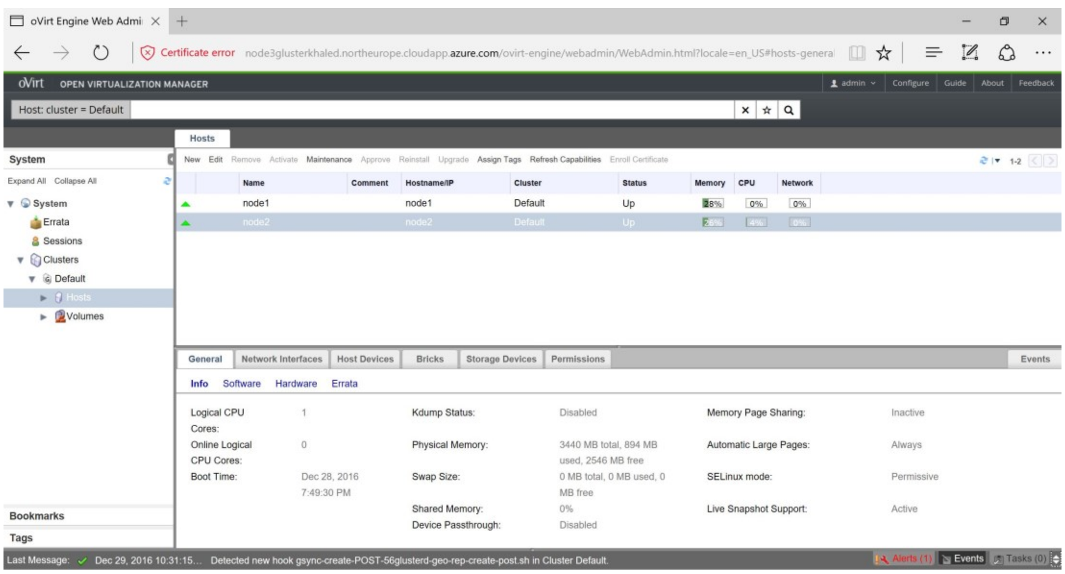


Fig10: nodes information

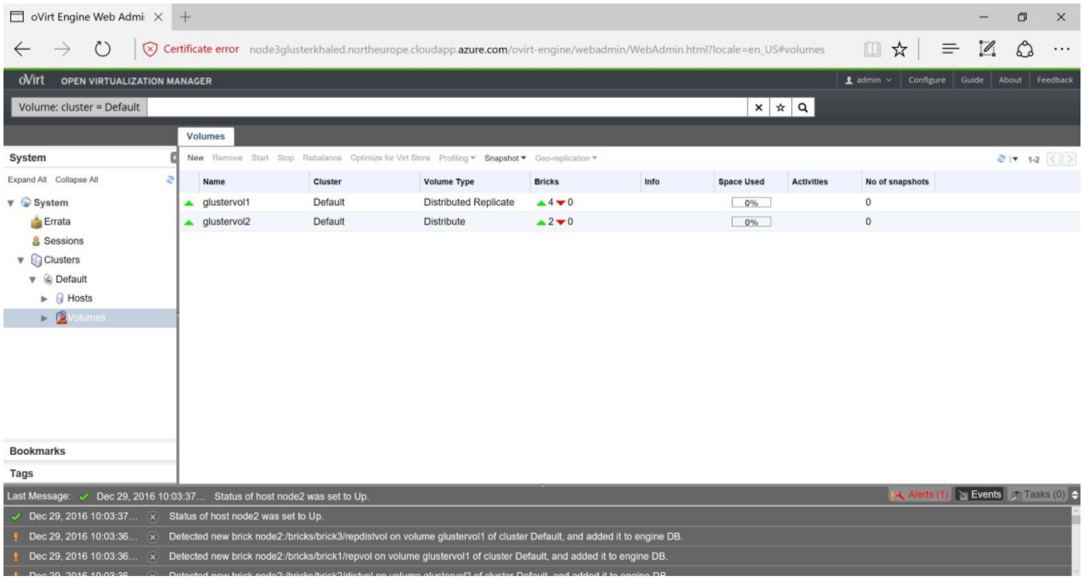


Fig11: volumes information

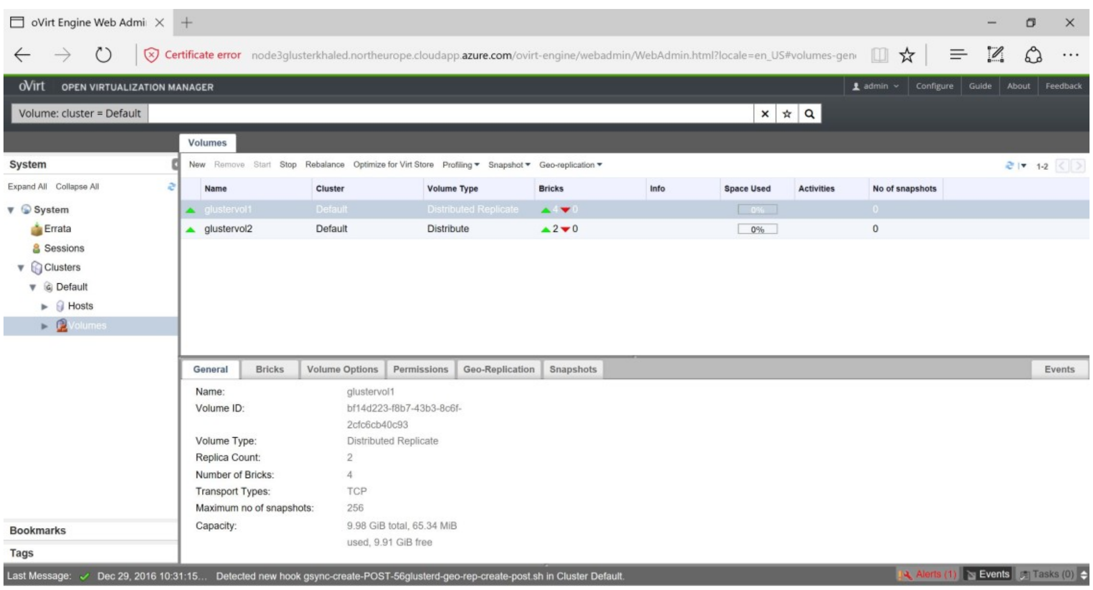


Fig12: Volume details

9. In addition to the management and visualization capabilities, *oVirt* provides a suite of pre-configured reports that enable you to monitor the system. The reports module is based on *JasperReports* and *JasperServer*. *JasperReports* is an open source Java reporting tool that can produce reports and export them to PDF, HTML, Microsoft Excel, RTF, ODT, Comma-separated values and XML files. It generates the reports from an XML or .jasper file. *JasperReports* server is a reporting server for JasperReports. It allows to generate, organize, secure and deliver interactive reports.

10. Install ovirt-engine-reports package and set up the reports engine. Accept all the default options except enabling the firewall and ignore the warning about the recommended RAM.

```
[node3]# yum install ovirt-engine-reports
[node3]# engine-setup
```

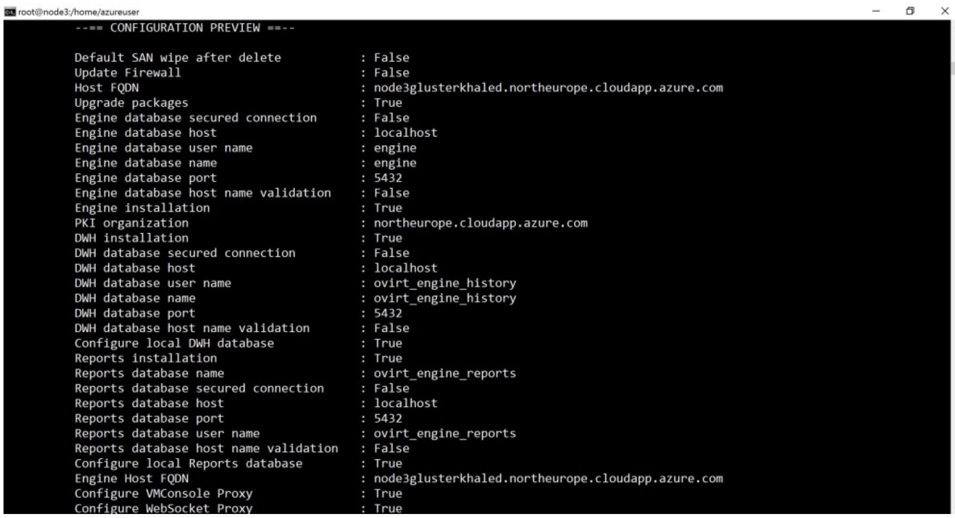


Fig13: engine reports installation

11. From the login page of the graphical console, select "Reports Portal", login with your username and password, navigate built-in reports and explore creating your own ones. Note that, reports require the system to run for few days before you can generate meaningful reports.

12. Well done! Now that you have accomplished all the required steps in this lab, you deserve a treat Too Too! ☺

```
[Azure]$ ssh -i .ssh/id_rsa azureuser@NODE3-PubIP

[node3]$ yum install epel-release -y

[node3]$ sudo yum install sl -y

[node3]$ sl

[node3]$ exit
```

13. Clean up the environment by deleting the resource group *glusterfsRG*

```
[Azure]# az group delete glusterfsRG
```

REFERENCES

USEFUL LINKS

https://access.redhat.com/documentation/en/red-hat-storage/

https://access.redhat.com/articles/using-gluster-with-azure

https://wiki.centos.org/HowTos/GlusterFSonCentOS

https://www.ovirt.org/blog/2016/08/up-and-running-with-ovirt-4-0-and-gluster-storage/

MICROSOFT AND RED HAT PARTNERSHIP

http://openness.microsoft.com/2016/04/15/microsoft-red-hat-partnership-accelerating-partner-opportunities/

https://www.redhat.com/en/microsoft