# Practice M3: Distributed and Shared Storage (Ubuntu)

This practice assumes that you are working in an on-premise environment

All tasks can be achieved under different configurations (host OS and/or virtualization solution) with the appropriate adjustments

This practice is oriented towards **Ubuntu 21.04 Server**

Chart, waterfall chart

Description automatically generated

## Part 1: Shared Storage (Samba)

### Samba

#### Install and configure the server

First, refresh the package information and then install the required packages

**sudo apt update**

**sudo apt install samba smbclient**

Now, we can check if the services are running

**systemctl status nmbd smbd**

Add firewall exception for **Samba**

**sudo ufw allow samba**

#### Create a public share

Make sure that you have this folder

**sudo mkdir -p /storage/samba/public**

Set the ownership of the folder

**sudo chown nobody:nogroup /storage/samba/public**

Set the permissions

**sudo chmod 1777 /storage/samba/public**

Now, open the main configuration file (**/etc/samba/smb.conf**) for **Samba** and add the following at the end

**[public]**

**comment = Public demo share**

**path = /storage/samba/public**

**writable = yes**

**browseable = yes**

**public = yes**

Save and close the file

Execute the following to test the configuration changes

**sudo testparm**

Reload the **Samba** services

**sudo systemctl restart smbd nmbd**

Let’s check for the available shares locally

**sudo smbclient -L //localhost**

When prompted for the **root** password just hit **Enter**

Now, let’s connect to our public share locally

**sudo smbclient -N //localhost/public**

We will be prompted with FTP-like interface

Here we can use **help** to check what commands are available

Then use the **mkdir** command to create a folder

**mkdir test**

And then list the catalog **ls**

Create a local file with

**!echo "Hello" > hello.txt**

List the local folder

**!ls**

Copy a local file a folder on the share

**put hello.txt**

List again the share with **ls**

And finally quit the share with **quit**

#### Install client tools

Log on to another machine (**ubuntu02**)

First, install the required packages

**sudo apt update**

**sudo apt install smbclient cifs-utils**

#### Connect from client

Check what is provided by the server

**smbclient -NL //ubuntu01**

Try to connect to the **public** share

**smbclient -N //ubuntu01/public**

List the contents with **ls**

Try to create a new folder named **client** with

**mkdir client**

And list again with **ls**

Finally, quit the Samba client with **quit**

#### Mount public share

Prepare the mountpoint

**sudo mkdir -p /mnt/samba/public**

Now, that we know the share is working, let’s try to mount it

**sudo mount -o guest,noperm //ubuntu01/public /mnt/samba/public**

We can verify the successful mount with either **df** or **mount**

Let’s try to create a file

**vi /mnt/samba/public/test.txt**

Should we want to mount the share on boot, then we must add it to the **/etc/fstab** with a line like:

**//ubuntu01/public /mnt/samba/public cifs guest,noperm 0 0**

#### Create a protected share

Return on the server

Create a dedicated user

**sudo useradd sambauser**

Create a special folder

**sudo mkdir -p /storage/samba/protected**

Change the ownership

**sudo chown sambauser:sambauser /storage/samba/protected**

Change the permissions

**sudo chmod 775 /storage/samba/protected**

Now, open the main configuration file (**/etc/samba/smb.conf**) for **Samba** and add the following at the end

**[protected]**

**comment = Protected demo share**

**path = /storage/samba/protected**

**create mask = 0664**

**directory mask = 0775**

**valid users = sambauser**

**write list = sambauser**

**writable = yes**

**public = yes**

**hosts allow = <client-ip-address>**

Close and save the file

Test the new configuration with

**sudo testparm**

Create **Samba** password for the dedicated user

**sudo smbpasswd -a sambauser**

To verify the new user creation, execute

**sudo pdbedit -Lv**

Finally, reload the **Samba** services

**sudo systemctl reload smbd nmbd**

Check what shares are offered by the server

**sudo smbclient -NL //localhost**

#### Access and mount a protected share

Log on to the client machine

List the available shares

**smbclient -L ubuntu01 -U sambauser%Password1**

Okay, this is not very secure. We can omit the password, and we will be asked to enter it

Connect to the share

**smbclient //ubuntu01/protected -U sambauser%<user-password>**

Create a folder or two, list the contents and then quit the client

Let’s create a mount point

**sudo mkdir -p /mnt/samba/protected**

Mount the protected share

**sudo mount -o username=sambauser,password=<user-password> //ubuntu01/protected /mnt/samba/protected**

Check with either **df** or **mount**

#### Mount protected share on boot

Unmount the share

**sudo umount /mnt/samba/protected**

We can create a record in our **/etc/fstab** file

Unfortunately, it will contain the password, which is not very secure

Alternatively, we can create a credentials file which is not reachable or readable by the regular users

Let’s create one in **/root/protected.cred**

It must have the following content

**username=sambauser**

**password=<user-password>**

Save and close the file

Open the **/etc/fstab** file and add the following at the end

**//ubuntu01/protected /mnt/samba/protected cifs credentials=/root/protected.cred 0 0**

Save and close the file

Test and mount the share with

**sudo mount -a**

Everything should work as expected

#### Windows client

Switch to the **Windows** client machine

Open **File Explorer**

In the address bar type **\\ubuntu01**

When asked for credentials type **sambauser** and **<user-password>**

Then click **OK**

Once you manage to enter the shared resource, try to create file or folder

Everything should work as expected

## Part 2: Shared Storage (NFS)

We will continue with the infrastructure used during the first part

### Network File System

#### Install and configure server

First, install the required packages

**sudo apt update**

**sudo apt install nfs-kernel-server**

The two configuration files **/etc/default/nfs-common** and **/etc/default/nfs-kernel-server** do not offer much optinos to change

Now, we can check what versions of the protocol are supported

**sudo cat /proc/fs/nfsd/versions**

And also, which ports are occupied with

**rpcinfo -p localhost | grep nfs**

For the full picture of listening services and ports, we can execute

**sudo ss -4ltp**

#### Export shares

Let’s check if we have any default exports available

**sudo exportfs**

There shouldn’t be any

Let’s prepare a folder to be shared first

Create a child folder under **/storage/nfs**

**sudo mkdir -p /storage/nfs/share**

Now, let’s provide some content

**sudo find /usr/share/doc -type f -name '\*.txt' -exec cp {} /storage/nfs/share/ \;**

And check what we have now

**ls -al /storage/nfs/share**

In order to share our folder, we must edit the **/etc/exports** file

**sudo vi /etc/exports**

Enter the following to share the folder with everyone with read and write permissions

**/storage/nfs/share \*(rw)**

We can execute the following to apply the changes

**sudo exportfs -rav**

And if we want more information about the options of the share, we can execute just

**sudo exportfs -v**

We can check in one more place for available exports

**cat /var/lib/nfs/etab**

#### Firewall settings

We can examine the state of our firewall by

**sudo ufw status**

Now, if the firewall is active, allow the NFS communication

**sudo ufw allow from 192.168.81.0/24 to any port nfs**

#### Install client tools

Log on to another machine (**ubuntu02**)

Install the required packages

**sudo apt install nfs-common**

Let’s create a mount point

**sudo mkdir -p /mnt/nfs/share**

With client tools installed, let’s try to mount the exported folder

**sudo mount -t nfs4 ubuntu01:/storage/nfs/share /mnt/nfs/share**

And check if it mounted successfully

**sudo mount -t nfs4**

Now browse it and explore some of the files

Try to create a folder

**mkdir /mnt/nfs/share/t1**

No success. Try with **sudo**, again no success

Return on the server and check the folder’s permissions

**ls -al /storage/nfs**

Change them to allow everyone to be able to do anything

**sudo chmod -R 777 /storage/nfs/share**

Return on the client

Try to create a folder with

**sudo mkdir /mnt/nfs/share/t1**

This time it is successful. Check the result with

**ls -al /mnt/nfs/share**

The folder is owned by **nobody**. Why? (*perhaps because we used* ***sudo*** *and there is the* ***root\_squash*** *option set*)

Of course, we can remove or change the option

Let’s try another approach

Unmount the filesystem

**sudo umount /mnt/nfs/share**

Check the permissions of the mount point

**ls -al /mnt/nfs/**

And change them to

**sudo chmod 777 /mnt/nfs/share**

Mount again the filesystem

**sudo mount -t nfs4 ubuntu01:/storage/nfs/share /mnt/nfs/share**

Now, create another folder

**mkdir /mnt/nfs/share/t2**

And check the result

**ls -al /mnt/nfs/share/**

This time, the resulting permissions are different. Why? (*perhaps because we did not use* ***sudo*** *and there is the* ***no\_all\_squash*** *option set*)

#### Mount export on boot

While still on the client, let’s first unmount the export

**sudo umount -at nfs4**

Check if it did unmount

**sudo mount -t nfs4**

Now, let’s open the **/etc/fstab** file and add the following at the end

**ubuntu01:/storage/nfs/share /mnt/nfs/share nfs4 defaults 0 0**

Save and close the file

Let’s test it with

**sudo mount -va**

Now, if we reboot the export will be mounted automatically

#### Windows client

Switch to the **Windows** client machine with account that has administrative privileges

Open **Server Manager**, navigate to **Tools** and click **Add Roles and Features**

On the welcome screen click **Next**

Make sure that the **Role-based or feature-based installation** option is selected and click **Next**

Check that the correct server is selected and click **Next**

On the server roles screen click **Next**

On the features screen select **Client for NFS** and click **Next**

On the confirmation screen click **Install**

Once the process is done, click **Close**

Alternatively, you can open **PowerShell** session with **Run as Administrator** and execute

**Install-WindowsFeature NFS-Client**

No matter how you installed the client, let’s try to connect to the **NFS** server

Open a command line session and type

**mount -o nolock \\ubuntu01\storage\nfs\share Z:**

This should result in device **Z:** pointing to the exported folder

Please note that the **Windows** client is very picky and depending on the version of the **NFS** server you may not have success, especially for version 4

## Part 3: Distributed Storage

We will continue with the infrastructure used during the first part

### iSCSI

#### Target Preparation

Log on to **M1**

Install the required package

**sudo apt update**

**sudo apt install targetcli-fb**

Create a folder to store the iSCSI disk files

**sudo mkdir /var/lib/iscsi-images**

Start the administration tool

**sudo targetcli**

Check the available commands with **help**

Then execute the **ls** command

Switch to the **fileio** backend

**cd backstores/fileio**

Create an iSCSI disk

**create D1 /var/lib/iscsi-images/D1.img 10G**

Check the result with the **ls** command

Switch to the **iscsi** functions

**cd /iscsi**

Define a new target

**create iqn.2021-09.lab.lsaa:m1.tgt1**

Enter the target

**cd iqn.2021-09.lab.lsaa:m1.tgt1/tpg1/luns**

Create a LUN using the disk created earlier

**create /backstores/fileio/D1**

Check the result with the **ls** command

Adjust the access to the resource

**cd ../acls**

Register the initiator

**create iqn.2021-09.lab.lsaa:m2.init1**

Enter the record (if not there already)

**cd iqn.2021-09.lab.lsaa:m2.init1/**

Set user and password

**set auth userid=demo**

**set auth password=demo**

Exit the administrative tool

**exit**

Adjust the firewall if needed and active

Don’t forget to enable and start the following service in order the configuration to be loaded automatically on boot

**sudo systemctl enable --now rtslib-fb-targetctl.service**

#### Initiator Preparation

Log on to the **M2** machine

Install the initiator package

**sudo apt update**

**sudo apt install open-iscsi**

Open the initiator configuration file for editing

**sudo vi /etc/iscsi/initiatorname.iscsi**

Set the name to match to your situation, for example **iqn.2021-09.lab.lsaa:m2.init1**

Save and close the file

Adjust the authentication settings in **/etc/iscsi/iscsid.conf** file

**sudo vi /etc/iscsi/iscsid.conf**

Change the mode on line 46 to **automatic**

Uncomment **node.session.auth.authmethod** = CHAP (line 59)

Uncomment and adjust **node.session.auth.username** and **node.session.auth.password** (lines 70 and 71)

Save and close

Initiate a target discovery with

**sudo iscsiadm -m discovery -t sendtargets -p ubuntu01**

Confirm what we have discovered

**sudo iscsiadm -m node -o show**

Login to the target

**sudo iscsiadm -m node --login**

Confirm the established session

**sudo iscsiadm -m session -o show**

We can check the available block devices either with **lsblk** or

**cat /proc/partitions**

You can restart the machine to check if the drives will be still there

Let’s create a partition on the **sdb** device

**sudo parted -s /dev/sdb -- mklabel msdos mkpart primary 16384s -0m**

Create a filesystem

**sudo mkfs.ext4 /dev/sdb1**

Prepare a mount point

**sudo mkdir -p /mnt/iscsi**

Mount it

**sudo mount /dev/sdb1 /mnt/iscsi**

Examine the free space by filesystem

**df -hT**

Should we want to mount the new filesystem at boot, we must get its partition UUID

**sudo blkid /dev/sdb1**

Then edit the **/etc/fstab** and add

**UUID="<copied-value>" /mnt/iscsi ext4 \_netdev 0 0**

### GlusterFS

For this part we will continue with the same infrastructure but this time we will use the third machine as well

#### Installation (Server)

Log on to the first node (**M1**)

Install the required packages

**sudo apt install glusterfs-server glusterfs-common**

Enable and start the service

**sudo systemctl enable --now glusterd**

Adjust the firewall settings (service **glusterfs**) if needed

Repeat this procedure on the second node (**M3**) as well

#### Configuration

On both nodes create a folder that will be used for the purpose of this exercise

**sudo mkdir -p /storage/glusterfs**

Return on the first node

Test if there is a communication with the second node

**sudo gluster peer probe ubuntu03**

Confirm the status

**sudo gluster peer status**

Create a volume

**sudo gluster volume create vol01 transport tcp ubuntu01:/storage/glusterfs ubuntu03:/storage/glusterfs**

As we are not using a separate partition, we must add **force** at the end of the above command

Get information about the volume

**sudo gluster volume info vol01**

We can check the status of the volume

**sudo gluster volume status vol01**

If the volume is not started, you can do it with

**sudo gluster volume start vol01**

#### Use GlusterFS

Log on to the third machine (**M2**) or **ubuntu02**

Install the required packages

**sudo apt install glusterfs-client**

Prepare a mount point

**sudo mkdir -p /mnt/glusterfs**

Mount the volume

**sudo mount -t glusterfs ubuntu01:/vol01 /mnt/glusterfs**

#### Testing

While on the client, test the **GlusterFS** by creating a few files

**sudo touch /mnt/glusterfs/file0{1..9}**

Log on to the first node and check if there are files

**ls -al /storage/glusterfs**

And then check on the other node

Some files reside on the first and others on the second

This is because of the volume type - **distributed**

This is not fault tolerant, so let’s change it

Being on the client, delete the files

**sudo rm -f /mnt/glusterfs/file\***

And unmount the filesystem

**sudo umount /mnt/glusterfs**

Return on the first server and stop the volume

**sudo gluster volume stop vol01**

Delete the volume

**sudo gluster volume delete vol01**

Finally, re-create if with the following command

**sudo gluster volume create vol01 replica 2 transport tcp ubuntu01:/storage/glusterfs ubuntu03:/storage/glusterfs force**

Get information about the volume

**sudo gluster volume info vol01**

Start the volume

**sudo gluster volume start vol01**

And check its status

**sudo gluster volume status vol01**

Return on the client machine and mount it again

**sudo mount -t glusterfs ubuntu01:/vol01 /mnt/glusterfs**

Re-create the set of test files

**sudo touch /mnt/glusterfs/file0{1..9}**

Now, if we check on both nodes, we will see that the files are **replicated**

To mount is during boot, we must add the following in **/etc/fstab**

**ubuntu01:/vol01 /mnt/glusterfs glusterfs defaults,\_netdev 0 0**