Install and Configure iSCSI Ubuntu

iSCSI, an acronym for Internet Small Computer System Interface, is a Storage Area Network protocol that is used by the organizations to facilitate online storage management. It relies on TCP/IP networks to send SCSI commands between the initiator (client) and the target (server) that provide block-level access to the storage devices which can either be LVM logical volumes, complete disks, files or partitions.

iSCSI target is therefore, a server that offers access to the shared storage devices while iSCSI initiator functions like a client that connects to the target and access the shared storage.

- **IQN (iSCSI Qualified Name)** — these are the names used to identify both the target and the initiator.
- **Backend Storage** — defines storage device an iSCSI target is providing access to.
- **Target** — service on an iSCSI server that provides access to the backend storage devices.
- **Initiator** — iSCSI client that connects to the target.
- **ACL** — Access Control List that lists iSCSI clients to be granted access to the storage device.
- **LUN (logical unit number)** — backend storage devices (disks, partitions, logical volumes, files, or tape drives) shared through the target.
- **Portal** — an IP address and port used by the target or the initatior to establish a connection.
- **TPG (target portal group)** — group of IP addresses and TCP ports to which specific iSCSI target will listen.
- **Login** — Authentication that gives an initiator access to LUNs on the target.

Configure iSCSI target

Install the targetcli, administration tool for viewing and managing the target configurations.

```
# apt -y install targetcli-fb
```

On the targetcli administration tool is installed, proceed to configure the iSCSI target.

Create the backend storage device (backstore)

To configure the backend storage launch the admin tool, targetcli. When run, it launches an interactive prompt.

```
# targetcli
Warning: Could not load preferences file /root/.targetcli/prefs.bin.
targetcli shell version 2.1.fb43
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/>
```

You can run the ls command to get an idea of what is under the current directory.

```
/> ls
o- /
.....
.................. [...]
  o- backstores
.....
........ [...]
  | o- block
.....
[Storage Objects: 0]
  | o- fileio
.....
[Storage Objects: 0]
  | o- pscsi
.....
[Storage Objects: 0]
  | o- ramdisk
.....
[Storage Objects: 0]
  o- iscsi
.....
...... [Targets: 0]
  o- loopback
.....
... [Targets: 0]
  o- vhost
.....
...... [Targets: 0]
/>
```

The first item listed above is the backstores. Backstores provides various storage levels, block, fileio, pscsi and ramdisk. Block and fileio are the commonly used ones with block backstore being the normal Linux block device such as a hard drive like /dev/sdb while fileio backstore is a file on the file system that has been created with a predefined size.

In our iSCSI server, we have an unused storage disks /dev/sdb and we will be using them as our backend block storage device We will also learn how to create file-based backend storage.

To create a backstore, you can cd to backstores/block or backstores/fileio and create the respective backstore. To create a block backstore, run the command below make the appropriate substitutions.

```
/> backstores/block create name=iscsi-disk01 dev=/dev/sdb
Created block storage object iscsi-disk01 using /dev/sdb.
/>
```

To create fileio backstore, let us first create a directory to store this storage.

```
# mkdir /tmp/iscsi_disks
```

Create a fileio backstore under the above directory of size 1G.

```
/> backstores/fileio create iscsi_file /tmp/iscsi_disks/disk01.img 1G
Created fileio iscsi_file with size 1073741824
/>
```

Run the ls command to see an overview of what we have created.

```
/> ls
o- /
.....................................................................
.................. [...]
  o- backstores
.....................................................................
........ [...]
  | o- block
.....................................................................
[Storage Objects: 1]
  | | o- iscsi-disk01 ..................................... [/dev/sdb
(2.0GiB) write-thru deactivated]
  | o- fileio
.....................................................................
[Storage Objects: 1]
  | | o- iscsi_file ....................... [/tmp/iscsi_disks/disk01.img
(1.0GiB) write-back deactivated]
  | o- pscsi
.....................................................................
[Storage Objects: 0]
  | o- ramdisk
.....................................................................
[Storage Objects: 0]
  o- iscsi
.....................................................................
...... [Targets: 0]
  o- loopback
.....................................................................
... [Targets: 0]
  o- vhost
.....................................................................
...... [Targets: 0]
/>
```

Create the iSCSI Target and Portal

To create an iSCSI target, navigate to scsi directory run the create command specifying the IQN of the target followed by the name of the target itself as shown below. NOTE the naming rule : **[iqn.(year)-(month).(reverse of domain name):(name of your choice)]** However, if are you lazy to type all these, just run the create command and it will automatically assign an IQN as well as the target name.

```
/> iscsi/
/iscsi> create iqn.2018-08.com.example.storagesrv01.target01
```

```
Created target iqn.2018-08.com.example.storagesrv01.target01.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
/iscsi>
```

As you can see on the last line, a default portal listening on TCP port 3260 is created when the target is created. To verify this, run ls command.

```
/iscsi> ls
o- iscsi
.....
........ [Targets: 1]
  o- iqn.2018-08.com.example.storagesrv01.target01
.......................................... [TPGs: 1]
    o- tpg1
.............................................................. [no-
gen-acls, no-auth]
      o- acls
.....................................................................
...... [ACLs: 0]
      o- luns
.....................................................................
...... [LUNs: 0]
      o- portals
.....................................................................
[Portals: 1]
        o- 0.0.0.0:3260
.....................................................................
. [OK]
/iscsi>
```

Set up the LUN

LUN is needed to associate a block device with a specific TPG. To create the LUN, naviate to the target portal group (tpg1) created above and create a LUN for both the block and fileio backstores created above.

```
/iscsi> iqn.2018-08.com.example.storagesrv01.target01/tpg1/
/iscsi/iqn.20...target01/tpg1> luns/ create /backstores/block/iscsi-disk01
Created LUN 0.
/iscsi/iqn.20...target01/tpg1> luns/ create /backstores/fileio/iscsi_file
Created LUN 1.
/iscsi/iqn.20...target01/tpg1>
```

Set up Access Control Lists (ACL)

ACL is used to specify the clients (iSCSI intiators) allowed to access the iSCSI target backstores. To cteate an ACL for an intiator, run create command with IQN of the initiator just like as did above.

```
/iscsi/iqn.20...target01/tpg1> acls/
/iscsi/iqn.20...t01/tpg1/acls> create
iqn.2018-08.com.example.srv01.initiator01
Created Node ACL for iqn.2018-08.com.example.srv01.initiator01
Created mapped LUN 1.
Created mapped LUN 0.
/iscsi/iqn.20...t01/tpg1/acls>
```

com.example.srv01 is a reversed domain name for out iSCSI client.

Set User ID and password for iSCSI initiator authentication

Set the userid and password as shown below and exit the targetcli admin console. When you exit the console, all the changes are saved automatically.

```
/iscsi/iqn.20...t01/tpg1/acls> iqn.2018-08.com.example.srv01.initiator01/
/iscsi/iqn.20...1.initiator01> set auth userid=username
Parameter userid is now 'username'.
/iscsi/iqn.20...1.initiator01> set auth password=strongpassword
Parameter password is now 'strongpassword'.
/iscsi/iqn.20...1.initiator01> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/rtslib-fb-target/backup.
Configuration saved to /etc/rtslib-fb-target/saveconfig.json
```

Once the changes are saved, start the target service and it should now be listening on port 3206. If firewall is running, allow iscsi target through it.

```
# ufw allow 3260
# ufw reload
# ss -napt | grep 3260
LISTEN   0     256           0.0.0.0:3260                0.0.0.0:*
```

Now that all is well, let proceed to configure the iSCSI initiator.

Configure iSCSI Initiator

Install iSCSI initiator packages

```
apt -y install open-iscsi
```

Edit the **/etc/iscsi/initiatorname.iscsi** and set the IQN of the initiator to be just the same as the one we created on the target above you can comment out the existing and add a new one as shown below. **iqn.2018-08.com.example.srv01.initiator01**

```
# vim /etc/iscsi/initiatorname.iscsi
```

```
#InitiatorName=iqn.1993-08.org.debian:01:02068a9161c
InitiatorName=iqn.2018-08.com.example.srv01.initiator01
```

Edit the **/etc/iscsi/iscsid.conf** configuration file to set the authentication method and specify the username and password defined above, under the CHAP settings

```
# vim /etc/iscsi/iscsid.conf
```

```
51 # CHAP Settings
52 # **************
53
54 # To enable CHAP authentication set node.session.auth.authmethod
55 # to CHAP. The default is None.
56 node.session.auth.authmethod = CHAP   <-------- uncomment this line
57
58 # To set a CHAP username and password for initiator
59 # authentication by the target(s), uncomment the following lines:
60 node.session.auth.username = username   <---- uncomment and set the right
user
61 node.session.auth.password = password     < - uncomment and set the right
password
```

Save the configuration file and restart and enable both iscsid and iscsi services.

```
# systemctl restart iscsid open-iscsi
# systemctl enable iscsid open-iscsi
```

Next, run target discovery against our iSCSI target server to find out the shared targets.

```
# iscsiadm -m discovery -t sendtargets -p 192.168.43.154
192.168.43.154:3260,1 iqn.2018-08.com.example.storagesrv01.target01
```

From the output above, we can see that the avaiable target is **iqn.2018-08.com.example.storagesrv01.target01** and in order to use it, we need to login to it. Run the command below to login to the target.

```
# iscsiadm -m node --login
Logging in to [iface: default, target:
iqn.2018-08.com.example.storagesrv01.target01, portal: 192.168.43.154,3260]
(multiple)
Login to [iface: default, target:
iqn.2018-08.com.example.storagesrv01.target01, portal: 192.168.43.154,3260]
successful.
```

You can also login to target by specifying the iqn as in:

```
# iscsiadm -m node -T iqn.2018-08.com.example.storagesrv01.target01 -l
```

Once logged in, you can run the following command to see the details of the established session.

```
# iscsiadm -m session -o show
tcp: [1] 192.168.43.154:3260,1 iqn.2018-08.com.example.storagesrv01.target01
(non-flash)
```

To increase the verbosity of the session details, pass option -P with the levels of verbosity, 0-3 with 0 meaning less verbose and 3 most verbose

```
# iscsiadm -m session -o show -P 1
Target: iqn.2018-08.com.example.storagesrv01.target01 (non-flash)
Current Portal: 192.168.43.154:3260,1
Persistent Portal: 192.168.43.154:3260,1
**********
Interface:
**********
Iface Name: default
Iface Transport: tcp
Iface Initiatorname: iqn.2018-08.com.example.srv01.initiator01
Iface IPaddress: 192.168.43.214
Iface HWaddress: <empty>
Iface Netdev: <empty>
SID: 1
iSCSI Connection State: LOGGED IN
iSCSI Session State: LOGGED_IN
Internal iscsid Session State: NO CHANGE
```

To check fileio and block disks shared from the iSCSI target, run the following command.

```
# lsblk --scsi
NAME HCTL       TYPE VENDOR   MODEL             REV TRAN
sda  2:0:0:0    disk ATA      VBOX HARDDISK     1.0 sata
sdb  3:0:0:0    disk LIO-ORG  iscsi-disk01      4.0 iscsi
sdc  3:0:0:1    disk LIO-ORG  iscsi_file        4.0 iscsi
sr0  0:0:0:0    rom  VBOX     CD-ROM            1.0 ata
sr1  1:0:0:0    rom  VBOX     CD-ROM            1.0 ata
```

And you can see that the that the block device and fileio targets shared are now available to the initiator as **sdb** and **sdc** respectively and can now be used as if they were locally mounted.

To make these devices usable, we need to partition them, create filesystems on them and mount them. To partition the devices, you can use any partitioning system you are comfortable with. In our case we used parted in a scripted format as shown below.

```
# parted -s /dev/sdb "mklabel msdos"
# parted -s /dev/sdb "mkpart primary 0% 100%"
```

```
# mkfs.ext4 /dev/sdb1
```

```
# parted -s /dev/sdc "mklabel msdos"
# parted -s /dev/sdc "mkpart primary 0% 100%"
# mkfs.ext4 /dev/sdc1
```

Create the mount points to mount the filesystems created above

```
# mkdir /media/{iscsi_disk01,iscsi_disk02}
```

```
# mount /dev/sdb1 /media/iscsi_disk01
# mount /dev/sdc1 /media/iscsi_disk02
```

Confirm the mounting

```
# df -h
Filesystem                  Size Used Avail Use% Mounted on
...
/dev/sdb1                   2.0G 6.0M 1.9G    1% /media/iscsi_disk01
/dev/sdc1                   992M 2.6M 923M    1% /media/iscsi_disk02
```

Finally, that is all it takes to configure iSCSI storage server and access this storage from a client. To automount these storages on system reboot, you can add their entries in the fstab. For more information on the same. check the man pages for **targetcli** admin tool and **iscsiadm** tool.