Linux Academy

Live! Lab

Configure an iSCSI Target and Initiator

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Live! Lab page
- Username: linuxacademy
- Password: 123456
- Root Password: 123456

Related Courses

Linux Academy Red Hat Certified Engineer Prep

Related Videos

Configure a System as Either an iSCSI Target or Initiator That Persistently Mounts an iSCSI Target - Target Setup

Configure a System as Either an iSCSI Target or Initiator That Persistently Mounts an iSCSI Target - Initiator Setup

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In the past, iSCSI was seen as too unstable to run on production environments, but due to improvements in both the iSCSI technology and networking overall, it is not just for virtual infrastructure anymore. This lab covers both creating an iSCSI target and initiator because both may be addressed on the RHCE exam.

The first server works as the iSCSI target, which provides storage that can be remotely mounted. The second server acts as the initiator, which mounts the storage.

iSCSI Target: Server

Log into your first server, and download the needed `targetcli` package:

```
[root@isci-target ~]# yum install -y targetcli
```

Enable the package, so it persists across reboots:

```
[root@isci-target ~]# systemctl enable target
Created symlink from /etc/systemd/system/multi-user.target.wants/target.
service to /usr/lib/systemd/system/target.service.
```

Before we start the service, we need to configure a storage mount. This can be either a *FILEIO backstore*, which creates an image of a defined size on the filesystem, or *block-level storage*, which is either a physical device or logical volume and provides better performance. We are using the BLOCKIO storage type for this lab.

Review the available drives:

```
[root@isci-target ~]# fdisk -l
WARNING: fdisk GPT support is currently new, and therefore in an
experimental phase. Use at your own discretion.
Disk /dev/xvda: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt

#         Start          End    Size  Type            Name
 1         2048         4095      1M  BIOS boot parti
 2         4096     20971486     10G  Microsoft basic
Disk /dev/xvdg: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

A second drive, either *xvdf* or *xvdg*, about 1 GB in size, is available.

Start the `target` application:

```
[root@isci-target ~]# targetcli
Warning: Could not load preferences file /root/.targetcli/prefs.bin.
targetcli shell version 2.1.fb41
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.
/>
```

We are working through the `target` CLI.

Create the first block in the storage device:

```
/> backstores/block/ create testblock1 /dev/xvdg
Created block storage object testblock1 using /dev/xvdg.
```

This creates a new storage object at `/dev/xvdg` by the name of *testblock1*.

Use the iSCSI command to create a qualified name. iSCSI qualified names are how target disks are referred when using iSCSI. It needs to follow the RTC 3270 naming convention. We gave ours the standard name of *iqn.2016-05.com.mylabserver:t1*, with the *t1* standing for *target 1*.

```
/> iscsi/ create iqn.2016-05.com.mylabserver:t1
Created target iqn.2016-05.com.mylabserver:t1.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
```

To review the created target group, treat iSCSI as a directory and move into the new target disk:

```
/> cd iscsi/iqn.2016-05.com.mylabserver:t1/tpg1
```

Review the contents using `ls`; we can see that we have no ACLs or LUNs set, and only one portal that listens on 3260 for attempted connections.

```
/iscsi/iqn.20...erver:t1/tpg1> ls
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]
```

We now want to create a LUN, or device associated with the mountable disk:

```
/iscsi/iqn.20...erver:t1/tpg1> luns/ create /backstores/block/testblock1
Created LUN 0.
```

Ensure that you are referring to the block using the name we originally gave it. This created the first LUN,

LUN 0.

The system needs to know how to refer to this via the clients. This can be done by referring to the IQN, and is called the *initiator name*.

Create the initiator name:

```
/iscsi/iqn.20...erver:t1/tpg1> acls/ create iqn.2016-05.com.
mylabserver:client
Created Node ACL for iqn.2016-05.com.mylabserver:client
Created mapped LUN 0.
```

Optionally, we can also provide a username and password for the client to authenticate itself. Move to the ACL's IQN directory:

```
/iscsi/iqn.20...erver:t1/tpg1> cd acls/iqn.2016-05.com.
mylabserver:client/
```

Set the user ID and password:

```
/iscsi/iqn.20...server:client> set auth userid=lunuser
Parameter userid is now 'lunuser'.
/iscsi/iqn.20...server:client> set auth password=password
Parameter password is now 'password'.
```

Remember to choose a more secure password in actual practice. The password for the *lunuser* is stored in plain text in the `/etc/` directory. Ensure that this password is not used for other services.

Check your configuration by returning to the main directory and using the `ls` command:

```
/iscsi/iqn.20...server:client> cd ../..
/iscsi/iqn.20...erver:t1/tpg1> ls
o- tpg1 ..................................... [no-gen-acls, no-auth]
  o- acls ................................................... [ACLs: 1]
  | o- iqn.2016-05.com.mylabserver:client ............... [Mapped LUNs: 1]
  |   o- mapped_lun0....................... [lun0 block/testblock1 (rw)]
  o- luns ................................................... [LUNs: 1]
  | o- lun0 ............................... [block/testblock1 (/dev/xvdg)]
  o- portals ............................................. [Portals: 1]
    o- 0.0.0.0:3260 ..................................... [OK]
```

This process generates a JSON file in the `/etc/target` directory. Exit the target CLI to review the JSON file:

```
/iscsi/iqn.20...erver:t1/tpg1> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup.
```

```
Configuration saved to /etc/target/saveconfig.json
[root@isci-target ~]# cd /etc/target/
[root@isci-target target]# ls
backup  saveconfig.json
```

The JSON file is an easy-to-read configuration file describing the configuration we set up in the CLI.

We now need to ensure the *3260* port is open to accepting connections:

```
[root@isci-target target]# firewall-cmd --permanent --add-port=3260/tcp
success
[root@isci-target target]# firewall-cmd --reload
success
```

Start the target system:

```
[root@isci-target target]# systemctl start target
```

This completes the configuration for the iSCSI target system.

Client Configuration

The client server works as an *initiator*, which persistently mounts an iSCSI device. The initiator does not have to be a remote system but is in this lab.

Install the iSCSI client utilities:

```
[root@isci-initiator ~]# yum install -y iscsi-initiator-utils
```

We now need to change a configuration setting located in the */etc/iscsi/* directory:

```
[root@isci-initiator ~]# cd /etc/iscsi/
```

This directory contains the file *initiatorname.iscsi*, wherein we need to add the initiator name that we configured on the server. Open the file in your chosen text editor, and replace the default name:

```
InitiatorName=iqn.2016-05.com.mylabserver:client
```

Save and exit the file.

Since we added a user and password for the client, we also need to update the *iscsid.conf* file, located in the same directory. Open the file, and navigate to the `#node.session.auth.authmethod = CHAP`

line. Uncomment the line. We also need to update the `#node.session.auth.username` and `#node.session.auth.password` lines with the username and password created earlier. Uncomment the lines.

```
# To enable CHAP authentication set node.session.auth.authmethod
# to CHAP. The default is None.
node.session.auth.authmethod = CHAP

#
# To set a CHAP username and password for initiator
# authentication by the target(s), uncomment the following lines:
node.session.auth.username = lunuser
node.session.auth.password = password
```

Save and exit.

Start the iSCSI service:

```
[root@isci-initiator iscsi]# systemctl start iscsi
[root@isci-initiator iscsi]# systemctl enable iscsi
```

We want to use the iSCSI administrative command to set up discovery mode using the IP address of our target ("server 1"). For this lab, use the *private IP* of the server. Use the `iscsiadm` tool to discover the available target.

```
[root@isci-initiator iscsi]# iscsiadm --mode discovery --type
sendtargets --portal 10.0.0.100
10.0.0.100:3260,1 iqn.2016-05.com.mylabserver:t1
```

Now that we know the target is these, we need to use the command to run the node and connect to the device:

```
[root@isci-initiator iscsi]# iscsiadm --mode node --target iqn.2016-05.
com.mylabserver:t1 --portal 10.0.0.100 --login
Logging in to [iface: default, target: iqn.2016-05.com.mylabserver:t1,
portal: 10.0.0.100,3260] (multiple)
Login to [iface: default, target: iqn.2016-05.com.mylabserver:t1,
portal: 10.0.0.100,3260] successful.
```

This pulls in the login information we provided via configuration file.

We can now check our configuration to ensure that we have the ability to mount the files.

```
[root@isci-initiator iscsi]# lsblk --scsi
NAME HCTL       TYPE VENDOR   MODEL             REV TRAN
sda  2:0:0:0    disk LIO-ORG  testblock1       4.0  iscsi
```

Ensure that it is *not* in read-only mode:

```
[root@isci-initiator ~]# lsblk | egrep "NAME|sda"
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda       8:0    0   1G  0 disk
```

What we're looking for is the value under **RO**. The `0` denotes that it is *not* in read-only mode.

Now we need to create the filesystem on this server:

```
[root@isci-initiator iscsi]# mkfs.ext4 /dev/sda
mke2fs 1.42.9 (28-Dec-2013)
/dev/sda is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=32 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:        32768, 98304, 163840, 229376
Allocating group tables: doneWriting inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Normally, we would not write the whole `/dev/sda` disk, but in this instance, the partition takes up the entire space of the disk.

Now, use the UUID to mount the system. This can be discovered using the `blkid` command, and grepping for */dev/sda*:

```
[root@isci-initiator ~]# blkid | grep "/dev/sda"/dev/sda:
UUID="0f071c9f-a67b-4911-8000-e7478009700e" TYPE="ext4"
```

Now edit the */etc/fstab*:

```
#
# /etc/fstab
# Created by anaconda on Fri Oct 17 18:33:48 2014
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more
info
```

```
#
UUID=668dbd02-c201-44bc-be76-f606fc9ab8db /                       xfs
defaults        1 1
# iscsi filesystem
UUID=0f071c9f-a67b-4911-8000-e7478009700e /mnt/iscsi              ext4
_netdev         0 0
```

_netdev is used instead of *defaults* so the system knows the iSCSI filesystem is a remote (or *network*) device, and does not try to load the filesystem until after the network is up.

Save and exit.

Create the */mnt/iscsi/* directory:

```
[root@isci-initiator ~]# mkdir /mnt/iscsi
```

Mount the filesystem:

```
[root@isci-initiator ~]# mount -a
```

Navigate into the iSCSI filesystem, and create a test file:

```
[root@isci-initiator ~]# cd /mnt/iscsi/
[root@isci-initiator iscsi]# echo “My first iSCSI filesystem” >
testfile.txt
```

We can additionally use the `iscsiadm` command to discover more information about our iSCSI mount:

```
[root@isci-initiator ~]# iscsiadm -m session -P 3
iSCSI Transport Class version 2.0-870
version 6.2.0.873-33.2
Target: iqn.2016-05.com.mylabserver:t1 (non-flash)
        Current Portal: 10.0.0.100:3260,1
        Persistent Portal: 10.0.0.100:3260,1
                **********
                Interface:
                **********
                Iface Name: default
                Iface Transport: tcp
                Iface Initiatorname: iqn.2016-05.com.mylabserver:client
                Iface IPaddress: 10.0.0.101
                Iface HWaddress: <empty>
                Iface Netdev: <empty>
                SID: 21
                iSCSI Connection State: LOGGED IN
                iSCSI Session State: LOGGED_IN
                Internal iscsid Session State: NO CHANGE
                *********
```

```
Timeouts:
*********
Recovery Timeout: 120
Target Reset Timeout: 30
LUN Reset Timeout: 30
Abort Timeout: 15
*****
CHAP:
*****
username: lunuser
password: ********
username_in: <empty>
password_in: ********
************************
Negotiated iSCSI params:
************************
HeaderDigest: None
DataDigest: None
MaxRecvDataSegmentLength: 262144
MaxXmitDataSegmentLength: 262144
FirstBurstLength: 65536
MaxBurstLength: 262144
ImmediateData: Yes
InitialR2T: Yes
MaxOutstandingR2T: 1
************************
Attached SCSI devices:
************************
Host Number: 22 State: running
scsi22 Channel 00 Id 0 Lun: 0
        Attached scsi disk sda          State: running
```