**iSCSI SAN**

When using SAN solutions, two scenarios are common. First, there is the type

of SAN that just has the purpose of separating storage from the physical server.

Another approach for using a SAN is in environments where high-availability

clusters are used. In a high-availability cluster, two nodes can access the same

storage simultaneously. Using a high-availability cluster needs to be supported

by the file system on the shared storage, though. If multiple computers are

going to write to the same XFS file system simultaneously, things will go wrong

for sure because the nodes will not be aware of what is happening on the other

node.

To use SAN storage in a clustered environment, where multiple nodes are

writing to the storage at the same time, you need a clustered file system such

as GFS2. When you are using a clustered file system, the different nodes that

are accessing the storage are aware of operations happening on the other node,

which prevents errors from happening.

To perform the tasks described in this chapter, you need two servers: one that is

configured as the iSCSI SAN, and the other one that is configured as the node that

is accessing the storage devices shared by the SAN. In the exercises in this chapter,

two servers are used:

■ Server1 is iSCSI target. On this server, you need disk space to set up an LVM

volume and a file. 100MB is enough for setting these up.

■ Server2 is the iSCSI initiator and does not need to meet any specific

requirements.

**Software Versus Hardware iSCSI SAN**

In this chapter, you learn how to configure an iSCSI SAN based on Red Hat Enterprise

Linux. We explore setting up a software iSCSI SAN solution. Apart from software

SAN solutions as described in this section, hardware iSCSI SAN solutions are

available as well.

No fundamental difference exists in the operation of software or hardware iSCSI

SAN. The main difference is that in a hardware iSCSI SAN, the hardware is optimized

to deliver the best possible performance. That does not necessarily mean that

a hardware iSCSI SAN is performing better than its software-based counterpart. It

all depends on the way it is optimized.

When installing a software-based iSCSI SAN, you may build a well-performing

solution. It all starts by selecting high-end hardware such as a fast network infrastructure

and fast disks to ensure that data can be written fast. Also, special iSCSI

optimized network cards can be used, often referred to as *host bus adapters* (HBAs).

These often include a hardware TCP offload engine that allows the iSCSI packets

to be handled on the HBA.

The software also plays an important role in optimizing iSCSI. There are many

ways to optimize the Linux iSCSI processes. This, in fact, can be a benefit as compared

to a hardware-based SAN solution. In a hardware SAN, you might have to

purchase additional license to be able to use advanced features or features that are

offering a better performance. In a software-based iSCSI SAN, all you need to create

the best possible SAN is just available as free software.

**iSCSI SAN Architecture**

When setting up an iSCSI SAN, you configure one server as the iSCSI target. This

is the server that offers access to the shared storage devices. When you configure

Red Hat Enterprise Linux as an iSCSI target, the shared storage devices typically

are LVM logical volumes, but they can be complete disks or partitions as well. You

can even configure the SAN to provide access to an image file (as you’ll see when

setting up the iSCSI SAN in the first exercise). This image file is just an empty file

used as the iSCSI storage backend, and it does not have to adapt to any specific

standard.

The other server is going to be used as the iSCSI initiator. This is the server that

connects to the SAN. After connecting to the SAN, the iSCSI initiator sees an additional

disk device.

When using a redundant network connection, the iSCSI initiator will see the SAN

device twice, once over each different path to the SAN. This might lead to a situation

where the same shared disk device is presented twice as well. To make sure that

in such a setup where redundant paths are available the SAN device is addressed

correctly, the iSCSI initiator should be configured to run the multipath driver. The

driver provides one interface that the initiator can talk through, which is set up for

redundancy and will ensure that the connection to the storage can just continue,

even if one of the paths to the SAN drops completely.

iSCSI Terminology

**Item Description**

IQN The iSCSI qualified name. A unique name that is used for identifying targets as

well as initiators.

Backend

storage

The storage devices on the iSCSI target that the iSCSI target component is

providing access to.

Target The service on an iSCSI server that gives access to backend storage devices.

Initiator The iSCSI client that connects to a target and is identified by an IQN.

ACL The access control list that is based on the iSCSI initiator IQNs and used to

provide access to a specific target. While setting up the iSCSI target on RHEL

7, creating an ACL is mandatory. This ACL is based on the IQN of the iSCSI

initiator that should be granted access.

**Setting Up the iSCSI Target**

Throughout different versions of Linux, different iSCSI target packages have been

used. In Red Hat Enterprise Linux 7, the LIO (Linux I/O) target is used. LIO is the

standard iSCSI target solution since Linux kernels 2.6.38, and because of its native

support in OpenStack Cloud, it has become an attractive storage solution that has

rapidly replaced alternative iSCSI target solutions in many Linux distributions. The

default interface to manage the LIO target is the **targetcli** command. This command

uses familiar Linux commands, such as **cd** , **ls** , **pwd** , and **set** to configure the

target.

The **targetcli** command was developed as a very intuitive command. Working with

it is like working in a directory structure where items need to be created in the different

directories. It has excellent command completion as well as a very good help

function, which makes working with the iSCSI target very intuitive.

When using the targetcli utility, you’ll go through a few steps to set up the target as

follows:

**1.** Create the backing storage devices.

**2.** Create the IQN and default target portal group (TGP).

**3.** Configure one or more ACLs for the TPG.

**4.** Create LUNs to provide access to the backing storage devices.

**5.** Create a portal to provide a network interface that iSCSI initiators can connect

to.

**6.** Verify and commit the configuration.

26.1 walks you through the procedure of setting up an iSCSI target using

**targetcli** .

**Exercise 26.1 Setting Up the Target with the targetcli Utility**

In this exercise, you set up an iSCSI target on server1. This exercise assumes that on

server1 that you have an LVM volume group available with the name vgsan. Within

the volume group, you need free disk space that allows you to create two LVM logical

volumes.

**1.** Open a root shell on server1. Type **vgs** to verify the name of the LVM volume

group and the amount of available disk space. If you do not have a volume

group with available disk space, you should create a volume group first.

**2.** Type **lvcreate -L 200M -n lvsan1 /dev/vgsan** and **lvcreate -L 200M -n**

**lvsan2 /dev/vgsan** to provide the backing storage needed for setting up your

iSCSI target.

**3.** Enter **yum -y install targetcli** .

**4.** Type **targetcli** . This opens the targetcli interface, which looks like a shell

prompt. Type **ls** to show the default interface

The first step in setting up an iSCSI target is configuring the backstore. Type

**cd /backstores** to enter the backstores branch of targetcli, which allows you to

specify which backing storage is going to be used.

**6.** Type **block/ create block1 /dev/vgsan/lvsan1**. This assigns the LVM logical

volume that you have created earlier as the backstore in the iSCSI target.

Repeat for lvsan2 by typing **block/ create block2 /dev/vgsan/lvsan2** . Notice

the way the command works; it starts with **block/** , which brings you to the contents

of the directory /block, and from there the **create** commands are issued.

Also notice that despite the fact that you are using the **create** command, it does

not really create the backing storage device; it assigns an already existing storage

device as the backing storage.

**7.** Now let’s create a file-backed block device as well: **fileio/ create file1 /root/**

**diskfile1 100M** . This command creates a sparse file with a size of 100 MiB and

assigns it as a backing storage device.

From

**8.** At this time, type **ls** to get an overview of the current configuration. It should

look like Figure 26.3 and show you two block devices and one file-backed storage

device.

**9.** Now that the block backstores are taken care of, you can start configuring the

unique identifier for your iSCSI target, the iSCSI IQN, which will also create

the default TGP. To start, use **cd /iscsi/** to get to the iscsi branch of the

configuration.

**10.** Now type **create iqn.2015-04.com.example:target** to create the IQN. Notice

that the name of the IQN starts with year-month (in YY-MM notation) and is

followed by the inversed DNS domain name. Make sure you respect this naming

standard; otherwise, the IQN will not be created.

**TIP** Notice that the IQN naming standard is very strict. If you specify the month

as one digit instead of two, for instance, you’ll get a “WWN not valid” message,

and creation will fail!

**11.** Type **ls** . This shows the contents of the iscsi branch, where you now see the

IQN you just created, as well as the TPG tpg1 that was created automatically

while creating the IQN

**12.** At this point, you can create an ACL in the TPG. Remember, you need to create

ACLs as well to access the iSCSI target, because without ACLs, any iSCSI

initiator will be denied access. Any new LUN that you create will be mapped

to each ACL that is associated with the TPG. This is because of the auto\_add\_

mapped\_luns feature, which is on by default. To create the ACL, first enter the

IQN that you just created. Type **cd iqn.[Tab]** to enter the IQN. Notice that in

the **cd** command, you are using the Tab key for command-line completion.

**13.** Now type **tpg1/acls/ create iqn.2015-04.com.example:server1** , which creates

a node ACL that allows server1 to access the IQN you just created. In this

command, the initiatorname as used in /etc/iscsi/initiatorname.iscsi on the

iSCSI initiator server is used. After completing this procedure, make sure that

the contents of this file on the iSCSI initiatorname matches. (See also the section

“Setting the Initiatorname” later in this chapter for more details.) If so

required, repeat this command for all other iSCSI initiators that need access to

this iSCSI target.

**14.** Now that you have created the ACL, you need to create the LUNs. The LUNs

are needed to associate a block device with a specific TPG. To do this, type the

following commands:

tpg1/luns/ create /backstores/block/block1

tpg1/luns/ create /backstores/block/block2

tpg1/luns/ create /backstores/fileio/file1

**TIP** While creating LUNs, you can specify additional parameters. For instance,

from the **tpg1/luns/** context, type **create lun=10 storage\_object=/backstores/**

**block/block1** to assign the LUN number 10 to the backing storage device that is

specified.

**15.** Type **ls** to verify what you have created so far.

**16.** At this point, you can create the portal. This connects the iSCSI configuration

to the specific IP address on the iSCSI target server. To make this work, you

need to ensure that the iSCSI target is on a fixed IP address. If that IP address

is 192.168.4.210, the command to use is **tpg1/portals/ create 192.168.4.210** .

Notice that this step is required only if you want to have the iSCSI target offering

its services on only a specific IP address. If you do not create a portal, a

default portal is used that binds to the IP address 0.0.0.0, which represents all

IP addresses on your server.

**17.** The configuration has now completed. Get back to the root of the configuration

tree by using **cd /** and type **ls** to get a complete overview of the configuration.

It should look like Figure 26.6 . You can now type **exit** to close the

configuration interface. This automatically writes the configuration file.

At this point, the iSCSI target configuration is written, but the target is not yet

operational. The configuration is written to the file /etc/target/saveconfig.json.

This is a file in the Java JSON format and is not really meant to be edited directly.

While you are saving the configuration, the iSCSI target service is also started automatically

and listening on port 3260 of the specified portal IP address. Listing 26.1

shows partial contents of the JSON file.

**Opening the Firewall**

Now that the iSCSI target has been configured, you need to make sure that it can

be accessed through the firewall and that the service is started automatically. To

start, type **systemctl enable target; systemctl start target** . To open port 3260 in

the firewall, type **firewall-cmd --add-port=3260/tcp --permanent; firewall-cmd**

**--reload** . No further action is required to run the iSCSI target.

**Finalizing the iSCSI Target Configuration**

**1.** On server1, type **systemctl start target** followed by **systemctl enable target** .

**2.** Type **systemctl status target** and verify that the target currently is active and

that the status is set to enabled

**3.** Open the firewall, using **firewall-cmd --add-port=3260/tcp --permanent** , followed

by **firewall-cmd --reload** .

**4.** Type **firewall-cmd --list-all** to verify the firewall configuration. The results

should look like

**Setting Up the iSCSI Initiator**

Now that the iSCSI target server is configured and operational, you can move on to

the next step and configure the iSCSI initiator. You can see this as the ISCSI client

because it will request access to storage offered by the iSCSI target server. To do

this, you need to go through a few steps:

**1.** Set the iSCSI initiatorname.

**2.** Use **iscsiadm** to discover available targets.

**3.** Use **iscsiadm** to log in to the target.

**Setting the iSCSI Initiatorname**

Access to the iSCSI target is based on the initiatorname. The initiatorname is

configured on iSCSI nodes where the iscsi-initiator-utils RPM package has been

installed. The initiatorname is stored in the file /etc/iscsi/initiatorname.iscsi.

After you install the iscsi-initiator-utils package, your nodes will have a default initiatorname

that is based on the Red Hat IQN and used as a UUID number. It may

look like InitiatorName=iqn.1994-05.com.redhat:76f76a97c56e224. There is nothing

wrong with using this default initiatorname, unless it is hard to read. That is why

you may consider using an initiatoname that is easier to read, such as iqn.2015-04.

com.example:server1.

While setting an initiatorname manually, make sure that you respect the syntax

restrictions. Initiatornames must start with iqn, followed by a four-digit year, followed

by a two-digit month and the inversed DNS name of your domain, which is

followed by a unique identifier for the initiator.

After changing the iSCSI initiatorname, do not forget to restart the iscsid service by

using **systemctl restart iscsid** !

**TIP** Notice that on the iSCSI initiator two services are needed. The iscsid service

is the main service that accesses all configuration files involved. The iscsi service is

the service that establishes the iSCSI connections.

**Performing the Discovery**

To connect to an iSCSI SAN, you start by exploring which configuration is available.

The iscsiadm command has different modes, and in discovery mode the command shows available connections. When using iSCSI discovery, you need three

different arguments:

■ **--type sendtargets** This tells the discovery mode how to find the iSCSI

targets. In some configurations, an iSNS service can be configured to make

discovery easier, but setting up iSNS is not an RHCE requirement. Therefore,

you’ll find the targets you need using --type sendtargets.

■ **--portal** This argument tells the **iscsiadm** command which IP address and

port to address to perform the discovery. You can use an IP address or node

name as the argument, and optionally, you can specify a port as well. If no port

is specified, the default port 3260 is used.

■ **--discover** This argument tells the iscsid service to perform a discovery.

Using these command arguments, you can perform an iSCSI discovery with the following

command:

iscsiadm --mode discovery --type sendtargets --portal 192.168.4.210

--discover

Note that instead of the **--mode discovery** option, you can also use **--mode**

**discoverydb** . This discovery mode uses a database and allows for some additional

commands to be used. For the RHCE objectives, you can consider the discovery and

discoverydb modes as more or less equivalent to one another.

After a successful discovery, you can request more information about the target that

was discovered, using the **-P** option. This option can be used in any iscsiadm mode,

and it will show details about the current mode. In all modes, the print levels 0 and

1 are supported. In some modes, you can go beyond that to display more additional

information. Listing 26.4 shows the result of the **iscsiadm --mode discovery -P 1**

command.

**Making the Connection**

Based on the name that you found when performing the iSCSI discovery and assuming

that you have set the initiatorname correctly, you can now log in to the iSCSI

target and make the actual connection. To do this, use a command that looks like

the following:

iscsiadm --mode node --targetname iqn.2014-11.com.exaple:target1

--portal 192.168.4.21-:3260 --login

In this command, a few options are used:

■ **--mode node** This specifies iscsiadm to enter “node” mode. This is the

mode in which the actual connection with the target can be established.

■ **--targetname** This specifies the name of the target as discovered when using

the iSCSI discovery process.

■ **--portal** This is the IP address and port on which the target is listening.

■ **--login** This authenticates to the target and will store credentials as well to

ensure that on reboot the connection can be reestablished again.

After logging in, a session with the iSCSI target is established. Both the session and

the node connection can be monitored, using the **-P** option. Use **iscsiadm --mode**

**node -P 1** to see node connection details, and **iscsiadm --mode session -P [1-3]**

to get information about the current iSCSI session

After making the connection to the iSCSI target, you’ll see the new SCSI devices as

offered by the target. A convenient command to list these commands is **lsscsi.**

**Making iSCSI Connections Persistent**

After configuring an iSCSI connection, you need to make sure that it comes back

after a restart. You also need to make sure that iSCSI disks can be mounted automatically

on reboot. In this section, you learn how to do that.

Managing iSCSI Connection Persistency

After logging in to an iSCSI target server, the connections are persistent automatically.

That means that on reboot, the **iscsid** and **iscsi** services are started on the

iSCSI client, and these services will read the iSCSI configuration that is locally

stored to automatically reconnect. Therefore, there is no need to put anything in

configuration files if you have successfully connected once to the iSCSI server.

All the relevant iSCSI configuration is stored in the directory /var/lib/iscsi. This

directory contains several subdirectories, of which the nodes subdirectory is the

most important one. In the nodes subdirectory, all previous connections are stored.

If you go into this directory, you can see a subdirectory that has the name of the

iSCSI target IQN. In this IQN subdirectory, you’ll find a subdirectory for each

known portal, which contains the file default that contains all session parameters.

If you need an iSCSI connection not to be restored after reboot, you first have to

log out to disconnect the actual session by using **iscsiadm --mode node**

**--targetname iqn.2014-11.com.example:target1 --logout** .

Next you need to delete the corresponding IQN subdirectory and all of its contents.

You can do this with the **rm** command or by using **iscsiadm --mode node**

**--targetname iqn.2014-11.com.example:target1 --op=delete** . This ensures that

all configuration is wiped and that you can make a clean restart.

**TIP** Stop the iscsi.service and remove all files under /var/lib/iscsi/nodes to clean

up all current configuration. After doing that, restart the iscsi.service and start the

discovery and login again.

Mounting iSCSI Devices

To mount an iSCSI device, you need to take care of a few things. First, the iSCSI

disk that now appears as /dev/sdb might appear as a different device name the next

time it is connected due to a topology change in your SAN configuration. For that

reason, it is not a smart idea to put a reference to /dev/sdb in the /etc/fstab file.

You should instead use a file system UUID. Every file system automatically gets

a UUID. To request the value of that UUID, you can use the **blkid** command.

Notice that this command shows results only if you have first created a file system

on the device that you want to use.

The second issue when making persistent iSCSI mounts is that normally the /etc/

fstab file is processed before the network is available. To make sure the iSCSI disk

can be mounted, you need to use the **netdev** mount option in /etc/fstab.

So to ensure that an iSCSI mount is configured persistently, put an entry in /etc/

fstab that looks like this:

UUID-XXXXXXXX-XXXX-XXXX-XXXXXXXX /iscsi xfs \_netdev 0 2

**Making an iSCSI Connection**

**1.** On server 2, open a root shell and type **yum -y install iscsi-initiator-utils**

**lsscsi** to install the software that you need to perform this exercise.

**2.** Type **iscsiadm --mode discovery --type sendtargets --portal 192.168.4.210**

**--discover** . This should return the name of the iSCSI target as you have configured

it in the previous exercise.

**3.** Next, type **iscsiadm --mode node --targetname iqn.2014-11.com.**

**example:target1 --portal 192.168.4.210:3260 --login** .

**4.** The iSCSI devices should now be available. Type **lsscsi** to show them. You

should see three LIO devices. Use **iscsiadm -m session -P3** , which also shows

all disks.

**5.** On the first iSCSI device (I’ll assume that it is /dev/sdb in this exercise, but it

can be a different device on your server depending on the configuration that is

used), type **mkfs.xfs /dev/sdb** .

**6.** Use **blkid /dev/sdb** to get the UUID that is set for the XFS file system that you

have just created on /dev/sdb.

**7.** Create a mount point for the iSCSI disk, using **mkdir /mnt/iscsi** .

**8.** Type **vim /etc/fstab** to open the /etc/fstab file in an editor and add a line that

looks like the following. (Make sure to replace the UUID with the UUID you

have found in Step 6 of this exercise.)

UUID=XXXXXXXX-XXXX-XXXX-XXXXXX /mnt/iscsi xfs \_

netdev 0 2

**9.** Type **mount -a** . This should mount the iSCSI disk. Type **mount** without any

arguments to verify.