**Configure a system as either an iSCSI target or initiator that persistently mounts an iSCSI target.**

**Presentation**

In the **iSCSI** world, you’ve got two types of agents:

* an **iSCSI** **target** provides some storage (here called **server**),
* an **iSCSI** **initiator** uses this available storage (here called **client**).

As you already guessed, we are going to use two virtual machines, respectively called **server** and **client**. If necessary, the **server** and **client** virtual machines can be one and only one machine.

**iSCSI Target Configuration**

Most of the **target** configuration is done interactively through the **targetcli** command. This command uses a directory tree to access the different objects.

To create an **iSCSI** **target**, you need to follow several steps on the **server** virtual machine.

Install the following packages:

# **yum install -y targetcli**

Activate the **target** service at boot:

# **systemctl enable target**

Note: This is mandatory, otherwise your configuration won’t be read after a reboot!

Execute the **targetcli** command:

# **targetcli**

Warning: Could not load preferences file /root/.targetcli/prefs.bin.

targetcli shell version 2.1.fb34

Copyright 2011-2013 by Datera, Inc and others.

For help on commands, type 'help'.

/>

You’ve got two options:

* You can create a **fileio backstore** called **shareddata** of **100MB** in the **/opt** directory (don’t hesitate to use tab completion):
* /> **backstores/fileio/ create shareddata /opt/shareddata.img 100M**

Created fileio shareddata with size 104857600

Note: If you don’t specify **write\_back=false** at the end of the previous command, it is assumed **write\_back=true**. The **write\_back** option set to **true** enables the local file system cache. This improves performance but increases the risk of data loss. In production environments, it is recommended to use **write\_back=false**.

* You can create a **block backstore** that usually provides the best performance. You can use a block device like **/dev/sdb** or a logical volume previously created (# **lvcreate –name lv\_iscsi –size 100M vg**):
* /> **backstores/block/ create block1 /dev/vg/lv\_iscsi**

Created block storage object block1 using /dev/vg/lv\_iscsi.

Then, create an **IQN** (**I**scsi **Q**ualified **N**ame) called **iqn.2014-08.com.example** with a target named **t1** and get an associated **TPG** (**T**arget **P**ortal **G**roup):

/> **iscsi/ create iqn.2014-08.com.example:t1**

Created target iqn.2014-08.com.example:t1.

Created TPG 1.

Global pref auto\_add\_default\_portal=true

Created default portal listening on all IPs (0.0.0.0), port 3260.

Note: The **IQN** follows the convention of the [RFC 3270](http://tools.ietf.org/html/rfc3720) (see <http://en.wikipedia.org/wiki/ISCSI> to get more details).

Now, we can go to the newly created directory:

/> **cd iscsi/iqn.2014-08.com.example:t1/tpg1**

/iscsi/iqn.20...ample: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]

Below **tpg1**, three objects have been defined:

* **acls** (**a**ccess **c**ontrol **l**ists: restrict access to resources),
* **luns** (**l**ogical **u**nit **n**umber: define exported resources),
* **portals** (define ways to reach the exported resources; consist in pairs of IP addresses and ports).

**If you use a version pre-RHEL 7.1** (this step is now automatically done by the **iscsi/ create** command), you need to create a **portal** (a pair of IP address and port through which the target can be contacted by initiators):

/iscsi/iqn.20...ple:t1/tpg1> **portals/ create**

Using default IP port 3260

Binding to INADDR\_ANY (0.0.0.0)

Created network portal 0.0.0.0:3260.

**Whatever version**, create a **lun** depending on the kind of backstore you previously chose:

* **Fileio** backstore:
* /iscsi/iqn.20...ample:t1/tpg1> **luns/ create /backstores/fileio/shareddata**

Created LUN 0.

* **Block** backstore:
* /iscsi/iqn.20...ample:t1/tpg1> **luns/ create /backstores/block/block1**

Created LUN 0.

Create an **acl** with the previously created **IQN** (here **iqn.2014-08.com.example**) and an identifier you choose (here **client**), together creating the future **initiator name**:

/iscsi/iqn.20...ample:t1/tpg1> **acls/ create iqn.2014-08.com.example:client**

Created Node ACL for iqn.2014-08.com.example:client

Created mapped LUN 0

**Optionally**, set a **userid** and a **password**:

/iscsi/iqn.20...ample:t1/tpg1> **cd acls/iqn.2014-08.com.example:client/**

/iscsi/iqn.20...xample:client> **set auth userid=usr**

Parameter userid is now 'usr'.

/iscsi/iqn.20...xample:client> **set auth password=pwd**

Parameter password is now 'pwd'.

Now, to check the configuration, type:

/iscsi/iqn.20...om.example:d1> **cd ../..**

/iscsi/iqn.20...ple:tgt1/tpg1> **ls**

o- tpg1 ................................................. [no-gen-acls, no-auth]

  o- acls ............................................................ [ACLs: 1]

  | o- iqn.2014-08.com.example:client ......................... [Mapped LUNs: 1]

  |   o- mapped\_lun0 ............................. [lun0 fileio/shareddata (rw)]

  o- luns ............................................................ [LUNs: 1]

  | o- lun0 .......................... [fileio/shareddata (/opt/shareddata.img)]

  o- portals ...................................................... [Portals: 1]

    o- 0.0.0.0:3260 ....................................................... [OK]

Finally, you can quit the **targetcli** command:

/iscsi/iqn.20...ple:tgt1/tpg1> **exit**

Global pref auto\_save\_on\_exit=true

Last 10 configs saved in /etc/target/backup.

Configuration saved to /etc/target/saveconfig.json

Note: The configuration is automatically saved to the **/etc/target/saveconfig.json** file.

Also, it can be useful to check the ports currently used:

# **netstat -ant**

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 0.0.0.0:22              0.0.0.0:\*               LISTEN

tcp        0      0 127.0.0.1:25            0.0.0.0:\*               LISTEN

tcp        0      0 0.0.0.0:3260            0.0.0.0:\*               LISTEN

tcp        0      0 192.168.1.81:22         192.168.1.81:33584      ESTABLISHED

tcp6       0      0 :::22                   :::\*                    LISTEN

tcp6       0      0 ::1:25                  :::\*                    LISTEN

Finally, open the **3260 tcp** port in the firewall configuration:

# **firewall-cmd --permanent --add-port=3260/tcp**

Success

Note1: With **RHEL 7.2** ([RHBZ#1150656](https://bugzilla.redhat.com/show_bug.cgi?id=1150656)), there is now a **firewalld** configuration file for the **iscsi-target** service. So you can type: # **firewall-cmd –permanent –add-service iscsi-target**  
Note2: In the new **/usr/lib/firewalld/services/iscsi-target.xml** configuration file, two lines are specified for the ports: **TCP 3260** and **UDP 3260**. As everything was working fine until now with the **TCP 3260** argument, I suppose that you can run **iSCSI** on top of **UDP** but it’s not the default option (I didn’t find any details in the [RFC7143](https://tools.ietf.org/html/rfc7143) on this point).

Reload the firewall configuration:

# **firewall-cmd --reload**

Success

**iSCSI Initiator Configuration**

To create an **iSCSI** **initiator**, you need to follow several steps on the **client** virtual machine.

Install the following package:

# **yum install -y iscsi-initiator-utils**

Edit the **/etc/iscsi/initiatorname.iscsi** and replace the content with the initiator name that you previously configured as acl on the target side:

**InitiatorName=iqn.2014-08.com.example:client**

If you previously set up a **userid** and a **password** on the **server**, edit the **/etc/iscsi/iscsid.conf** file and paste the following lines:

**node.session.auth.authmethod = CHAP**

**node.session.auth.username = usr**

**node.session.auth.password = pwd**

Start the **iscsi** service:

# **systemctl start iscsi**

**Caution**: This action is mandatory to be able to unmount the remote resource when rebooting. Don’t confuse **iscsid** and **iscsi** services!

Execute the **iscsiadm** command in **discovery** mode with the **server** ip address (here **192.168.1.81**):

# **iscsiadm --mode discovery --type sendtargets --portal 192.168.1.81**

192.168.1.81:3260,1 iqn.2014-08.com.example:t1

Note1: If you don’t specify any port, the default port is **3260**.  
Note2: Don’t mention a **DNS** entry as your portal address (here **192.168.1.81**), this would be a bad idea causing you a lot of trouble.

Execute the **iscsiadm** command in **node** mode with the **server** ip address (here **192.168.1.81**):

# **iscsiadm --mode node --targetname iqn.2014-08.com.example:t1 --portal 192.168.1.81 --login**

Logging in to [iface: default, target: iqn.2014-08.com.example:t1, portal: 192.168.1.81,3260] (multiple)

Login to [iface: default, target: iqn.2014-08.com.example:t1, portal: 192.168.1.81,3260] successful.

Note: As before, if you don’t specify any port, the default port is **3260**. Use of **DNS** entry as portal address only brings problems.

To check the configuration, type:

# **lsblk --scsi**

NAME HCTL       TYPE VENDOR   MODEL             REV TRAN

sda  2:0:0:0    disk LIO-ORG  shareddata       4.0  iscsi

To be sure that your resource is **not** in **read-only** mode (**1**=**read-only** mode), type:

# **lsblk | egrep "NAME|sda"**

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 100M **0** disk

Now, you can create a file system:

# **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=1024 (log=0)

Fragment size=1024 (log=0)

Stride=0 blocks, Stripe width=4096 blocks

25688 inodes, 102400 blocks

5120 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=33685504

13 block groups

8192 blocks per group, 8192 fragments per group

1976 inodes per group

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729

Allocating group tables: done

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

Retrieve the **UUID** of this disk:

# **blkid | grep "/dev/sda"**

/dev/sda: UUID="4a184c70-20ad-4d91-a0b1-c2cf0eb1986f" TYPE="ext4"

Add the disk **UUID** to the **/etc/fstab** file:

# **echo "UUID=..." >> /etc/fstab**

Note: Be **very** careful to type **>>** and not **>**, otherwise this will **destroy** all your configuration!  
Make a copy of the **/etc/fstab** file **before** doing this operation if you don’t want to take any risk.

Edit the **/etc/fstab** file and add the mount point (here **/mnt**), the file system type (here **ext4**) and the mount options (**\_netdev**):

**UUID=... /mnt ext4 \_netdev 0 0**

Note: The **\_netdev** mount option is mandatory to postpone the mount operation after the network initialization. If you don’t do it, the initiator boot process will be stopped after a timeout in maintenance mode (more information about the **\_netdev** option [here](http://codingberg.com/linux/systemd_when_to_use_netdev_mount_option)).

To check your configuration, type:

# **mount /mnt**

# **touch /mnt/testFile**

Note: A best practice is to execute the **mount -a** command, each time you change something in the **/etc/fstab** file to detect any boot problem before it occurs.

Optionally, you can dump all the initiator configuration (**3**=max output, **0**=min output):

# **iscsiadm -m session -P 3**

iSCSI Transport Class version 2.0-870

version 6.2.0.873-28

Target: iqn.2014-08.com.example:t1 (non-flash)

Current Portal: 192.168.1.81:3260,1

Persistent Portal: 192.168.1.81:3260,1

\*\*\*\*\*\*\*\*\*\*

Interface:

\*\*\*\*\*\*\*\*\*\*

Iface Name: default

Iface Transport: tcp

Iface Initiatorname: iqn.2014-08.com.example:client

Iface IPaddress: 192.168.1.10

Iface HWaddress:

Iface Netdev:

SID: 1

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: usr

password: \*\*\*\*\*\*\*\*

username\_in:

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: 2 State: running

scsi2 Channel 00 Id 0 Lun: 0

Attached scsi disk sda State: running

Source: **targetcli** man page and [Linux-iSCSI wiki](http://linux-iscsi.org/wiki/ISCSI).

**Useful Tips**

Before rebooting, [set up a virtual console](https://www.certdepot.net/rhel7-access-virtual-machines-console/), this can be helpful!

If you need to shut down target and initiator, shut down the initiator first. If you shut down the target first, the initiator won’t be able to unmount the remote resource and will be stuck in the shutdown process.

During the exam, as an extra precaution, unmount the remote resource before rebooting the initiator, you will avoid any bad surprise.