Сервисы хранения данных. iSCSI

iSCSI (Internet Small Computer System Interface) - это протокол передачи блочных данных, который позволяет использовать сеть TCP/IP для передачи SCSI-команд и данных между устройствами хранения (например, хранилищами данных) и серверами.
Основная идея iSCSI заключается в том, что он позволяет серверам обращаться к удаленным блочным устройствам хранения, таким как жесткие диски или массивы хранения данных, как если бы они были локальными устройствами. iSCSI-устройство хранения, называемое iSCSI-целевым устройством (iSCSI target), предоставляет доступ к блочным данным через сеть TCP/IP.
Конфигурация iSCSI включает в себя следующие основные компоненты:

1. **iSCSI-целевое устройство (iSCSI target):** Это устройство хранения, которое предоставляет доступ к блочным данным по протоколу iSCSI. Оно может быть физическим сервером или специальным устройством хранения, подключенным к сети.
2. **iSCSI-инициатор (iSCSI initiator):** Это клиентское устройство или сервер, которое инициирует соединение с iSCSI-целевым устройством и использует блочные данные, предоставляемые целевым устройством.
3. **IP-сеть:** iSCSI-соединение устанавливается через IP-сеть, обычно с использованием сети Ethernet.

Преимущества использования iSCSI включают:

- Гибкость и масштабируемость: iSCSI позволяет использовать сеть TCP/IP для доступа к удаленным устройствам хранения, что делает его более гибким и масштабируемым решением.
- Низкая стоимость: Использование существующей IP-сети и Ethernet-инфраструктуры позволяет снизить затраты на оборудование по сравнению с традиционными SAN (Storage Area Network).
- Удобство администрирования: iSCSI устройства хранения могут быть централизованно управляемыми и настраиваемыми, что облегчает администрирование и поддержку.
- Возможность работы на больших расстояниях: iSCSI позволяет обеспечить доступ к удаленным хранилищам данных через интернет или другие сети на больших расстояниях.

iSCSI является широко используемым протоколом для доступа к удаленным блочным устройствам хранения и нашел широкое применение в различных областях, включая виртуализацию серверов, резервное копирование и хранилища данных.

TargetCLI (Target Command Line Interface) - это интерактивный инструмент командной строки для настройки iSCSI-целевого устройства в операционных системах Linux. Он предоставляет удобный способ управления iSCSI-целями и настройкой параметров iSCSI в Linux.

TargetCLI является частью проекта LIO (Linux-IO), который является реализацией iSCSI-целевого устройства в ядре Linux. С помощью TargetCLI можно создавать и настраивать iSCSI-цели, логические блочные устройства (LUN) и другие параметры iSCSI.

Вот некоторые основные команды, которые могут быть использованы в TargetCLI:

- **ls** или **ls /backstores:** Показывает список доступных хранилищ (backstore).
- **create:** Создает новый объект, такой как iSCSI-цель или LUN.
- **delete:** Удаляет указанный объект.
- **cd:** Переходит в указанный контекст, например, внутрь iSCSI-цели или LUN.
- **set:** Устанавливает значение указанного параметра объекта.
- **saveconfig:** Сохраняет текущую конфигурацию в файл.
- **exit:** Выходит из интерфейса TargetCLI.

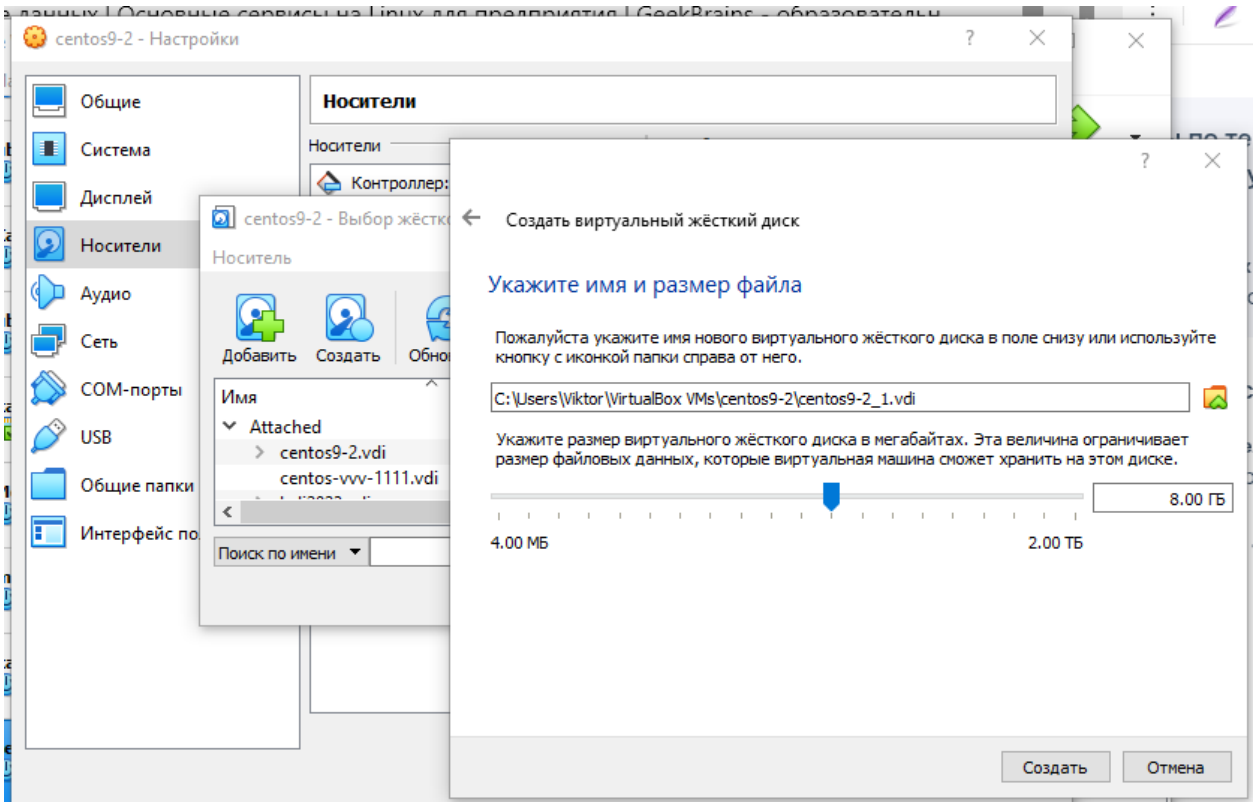
TargetCLI предоставляет много возможностей для настройки и управления iSCSI-целями и LUN в Linux. Он позволяет создавать, настраивать и удалять объекты, а также устанавливать различные параметры для оптимизации работы iSCSI.

Примечание: Для использования TargetCLI вам может потребоваться установить пакет targetcli, если он не установлен в вашей операционной системе.

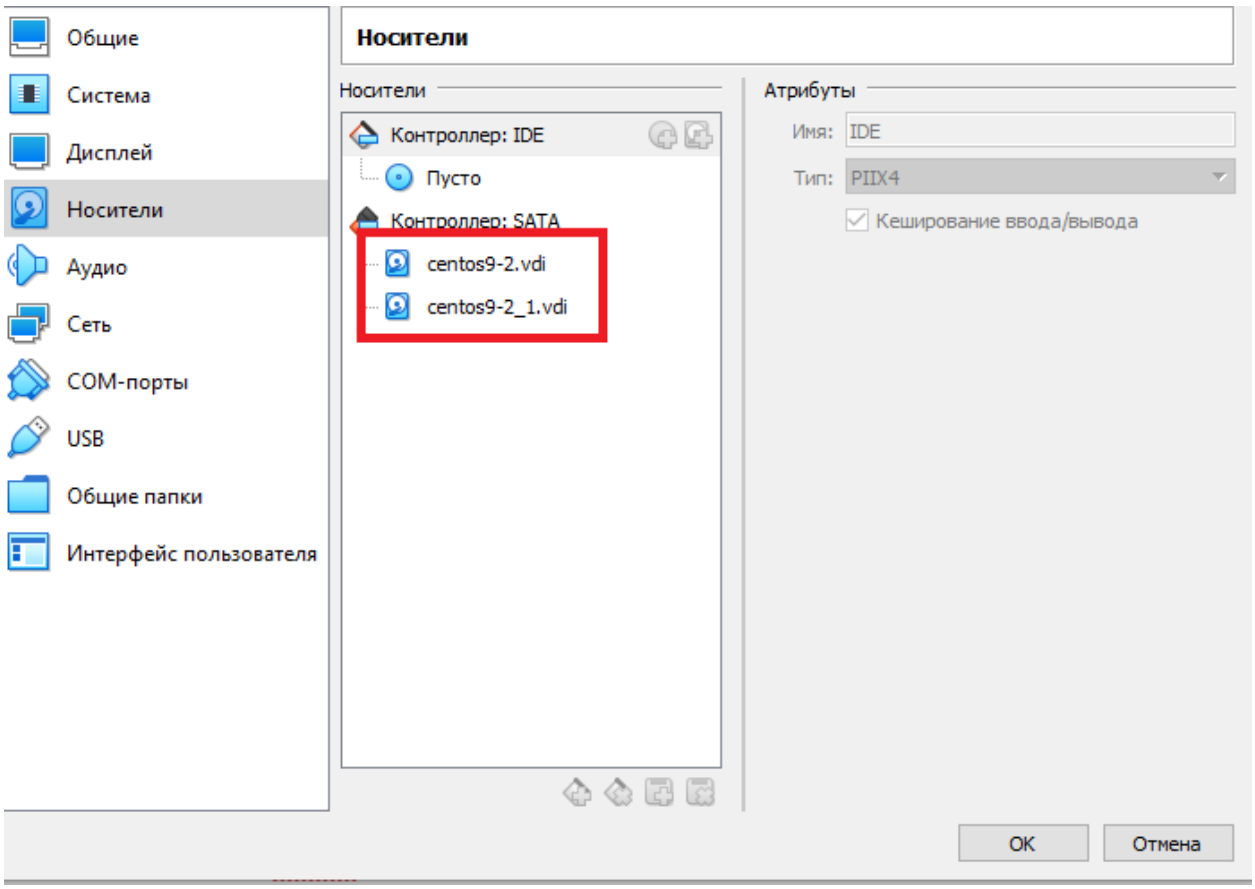
Пример настройки целевого устройства и инициализатора.

Понадобятся две виртуальные машины с ОС Centos 9 Stream.

На целевом устройстве нужно добавить еще один виртуальный жесткий диск:



Теперь имеются два диска, один системный, второй будем “расшаривать”:



Приступаем:

1. **Создание целевого устройства (iSCSI таргета) на linux с помощью targetcli.**

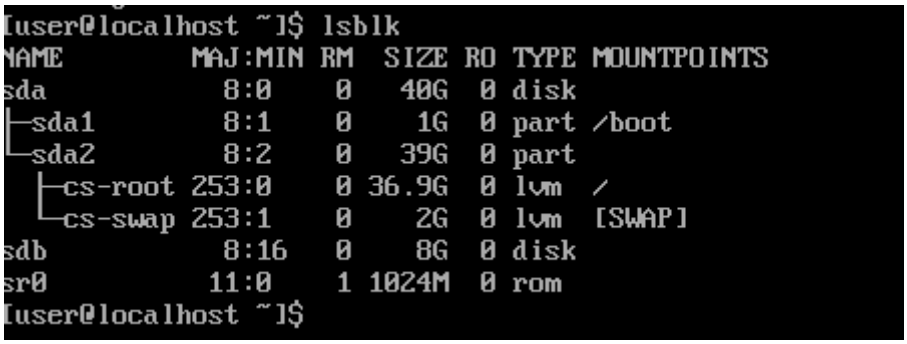
Установка пакета targetcli:

```
[user@localhost ~]$ sudo dnf install -y targetcli
[sudo] password for user:
Last metadata expiration check: 0:02:22 ago on Sun 25 Feb 2024 02:00:02 PM +06.
Dependencies resolved.
================================================================================
 Package                 Arch        Version                Repository      Size
================================================================================
Installing:
 targetcli               noarch      2.1.57-2.el9           appstream       75 k
Installing dependencies:
 python3-configshell     noarch      1:1.1.30-1.el9         baseos          72 k
 python3-kmod            x86_64      0.9-32.el9             baseos          84 k
 python3-pyparsing       noarch      2.4.7-9.el9            baseos         150 k
 python3-rtslib          noarch      2.1.76-1.el9           appstream      100 k
 python3-urwid           x86_64      2.1.2-4.el9            baseos         837 k
 target-restore          noarch      2.1.76-1.el9           appstream       14 k
```

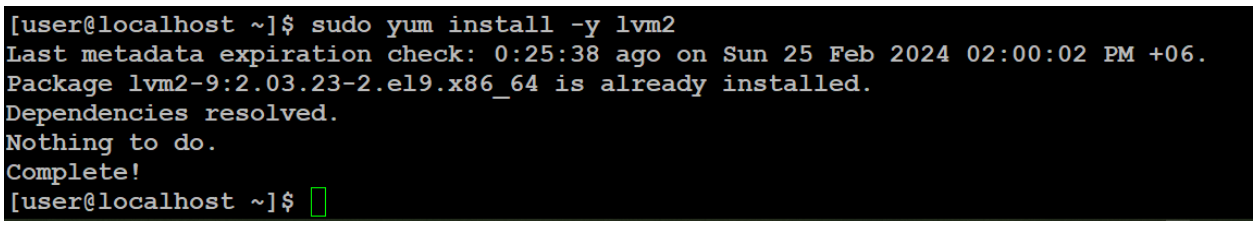
```
Installed:
  python3-configshell-1:1.1.30-1.el9.noarch          python3-kmod-0.9-32.el9.x86_64
  python3-pyparsing-2.4.7-9.el9.noarch               python3-rtslib-2.1.76-1.el9.noarch
  python3-urwid-2.1.2-4.el9.x86_64                   target-restore-2.1.76-1.el9.noarch
  targetcli-2.1.57-2.el9.noarch

Complete!
```

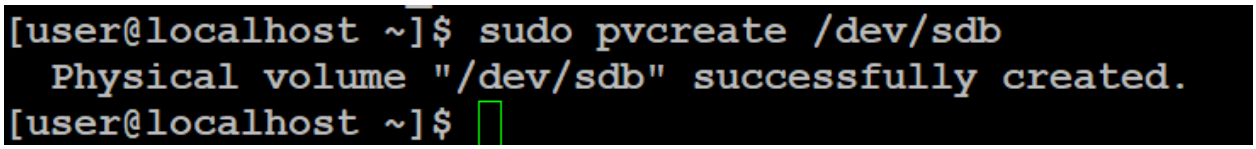
Проверим имеющиеся блочные устройства, sda – системный диск, sdb - дополнительный



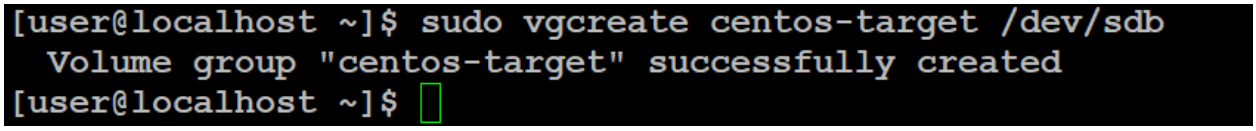
Подготовка диска. Дополнительный диск нужно отформатировать, для этого установим lvm2 (программа для управления логическими томами в операционных системах Linux):



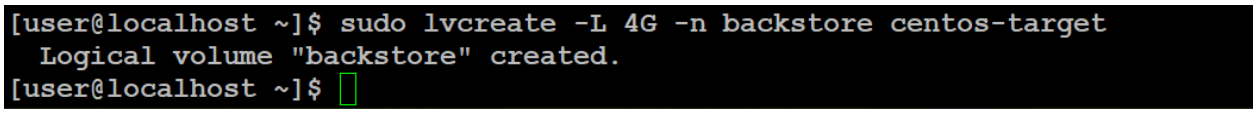
Создание физического том (PV) на дополнительном жестком диске:

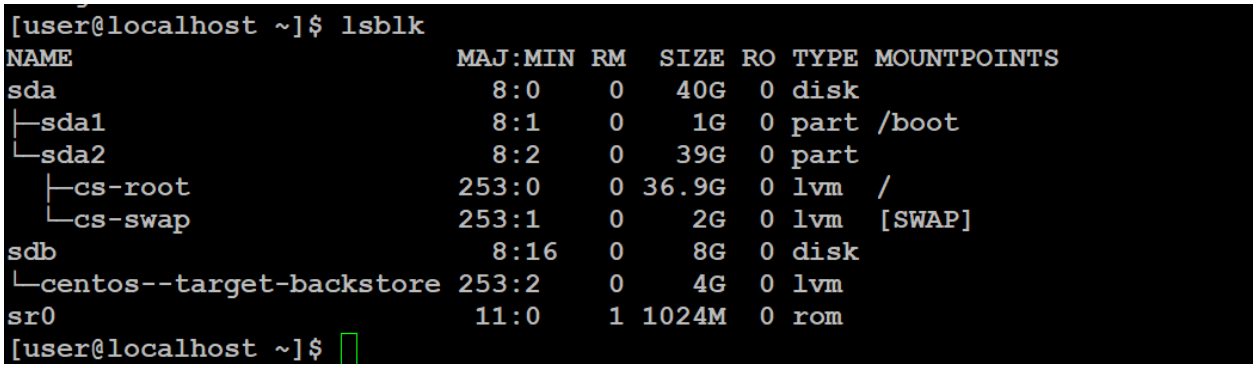


Создание группы томов (centos-target) и добавление физического тома в нее:



Создание логического тома (backstore) внутри группы томов(centos-target):





backstore - это абстрактный слой программного обеспечения, который отвечает за хранение данных и предоставляет интерфейс для доступа к ним через протокол iSCSI.

Backstore выполняет преобразование запросов iSCSI в операции ввода-вывода на устройстве хранения. Он может использовать различные механизмы для хранения данных, такие как файлы, блочные устройства, облачные хранилища и другие.

Запуск targetcli:

```
[user@localhost ~]$ sudo targetcli
targetcli shell version 2.1.57
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> 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]
/>
*              /              backstores/    iscsi/         loopback/      bookmarks
cd             clearconfig    exit           get            help           ls
pwd            refresh        restoreconfig  saveconfig     sessions       set
status         version
```

Создаем блочный backstore с именем "disk1", который будет использовать **/dev/centos-target/backstore** в качестве устройства хранения данных:

```
/> cd backstores/block
/backstores/block> create disk1 /dev/centos-target/backstore
Created block storage object disk1 using /dev/centos-target/backstore.
/backstores/block>
```

```
/backstores/block> cd ..
/backstores> cd ..
/> ls
o- / ..... [...]
  o- backstores ..... [...]
  | o- block ..... [Storage Objects: 1]
  | | o- disk1 ..... [/dev/centos-target/backstore (4.0GiB) write-thru deactivated]
  | |   o- alua ..... [ALUA Groups: 1]
  | |     o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
  | o- fileio ..... [Storage Objects: 0]
  | o- pscsi ..... [Storage Objects: 0]
  | o- ramdisk ..... [Storage Objects: 0]
  o- iscsi ..... [Targets: 0]
  o- loopback ..... [Targets: 0]
```

IQN (iSCSI Qualified Name) представляет собой уникальное имя, используемое для идентификации iSCSI-объекта в сети. Оно должно быть уникальным для каждого iSCSI-объекта, чтобы избежать конфликтов и позволить идентифицировать объект в сети.

Обычно IQN выглядит как строка в формате:

iqn.<год-месяц>.<обратный домен>:<имя_объекта>

Создание нового iSCSI-объекта (iqn нашего таргета):

```
/> cd iscsi
/iscsi> create iqn.2024-02.com.example:server1-disk1
Created target iqn.2024-02.com.example:server1-disk1.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
/iscsi>
```

```
/iscsi> ls
o- iscsi ..... [Targets: 1]
  o- iqn.2024-02.com.example:server1-disk1 ..... [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]
```

Настройка acl. Переходим в пространство имен iSCSI, далее в конкретный iSCSI-объект, для которого нужно настроить ACL. Указываем имя клиента, которому разрешен доступ, например, iqn.2024.com.example:client1

```
/iscsi> cd iqn.2024-02.com.example:server1-disk1/tpg1/acls
/iscsi/iqn.20...sk1/tpg1/acls> create iqn.2024-02.com.example:client1
Created Node ACL for iqn.2024-02.com.example:client1
```

LUN (Logical Unit Number) - это логический блок данных или виртуальный диск, предоставляемый хранилищем данных (например, хранилищем SAN или NAS) посредством протокола iSCSI.

Каждый LUN идентифицируется уникальным идентификатором LUN (LUN ID) и может быть доступен для подключения к удаленным устройствам через сеть. LUN является абстракцией физических накопителей и разделов, предоставляющих пространство для хранения данных удаленным устройствам, таким как серверы или виртуальные машины.

Настройка лунов (создание логического диска, который будет виден как блочное устройство у клиента):

```
/iscsi/iqn.20...sk1/tpg1/acls> cd ..
/iscsi/iqn.20...r1-disk1/tpg1> cd luns
/iscsi/iqn.20...sk1/tpg1/luns>
/          @last      bookmarks  cd         create     delete     exit       get
help       ls         pwd        refresh    set        status
/iscsi/iqn.20...sk1/tpg1/luns>
```

```
/iscsi/iqn.20...sk1/tpg1/luns> create /backstores/block/disk1
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.2024-02.com.example:client1
/iscsi/iqn.20...sk1/tpg1/luns>
```

```
/iscsi/iqn.20...sk1/tpg1/luns> cd /
/> ls
o- / ..... [...]
  o- backstores ..... [...]
  | o- block ..... [Storage Objects: 1]
  | | o- disk1 ..... [/dev/centos-target/backstore (4.0GiB) write-thru activated]
  | |   o- alua ..... [ALUA Groups: 1]
  | |     o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
  | o- fileio ..... [Storage Objects: 0]
  | o- pscsi ..... [Storage Objects: 0]
  | o- ramdisk ..... [Storage Objects: 0]
  o- iscsi ..... [Targets: 1]
  | o- iqn.2024-02.com.example:server1-disk1 ..... [TPGs: 1]
  |   o- tpg1 ..... [no-gen-acls, no-auth]
  |     o- acls ..... [ACLs: 1]
  |     | o- iqn.2024-02.com.example:client1 ..... [Mapped LUNs: 1]
  |     |   o- mapped_lun0 ..... [lun0 block/disk1 (rw)]
  |     o- luns ..... [LUNs: 1]
  |     | o- lun0 ..... [block/disk1 (/dev/centos-target/backstore) (default_tg_pt_gp)]
  |     o- portals ..... [Portals: 1]
  |       o- 0.0.0.0:3260 ..... [OK]
  o- loopback ..... [Targets: 0]
/>
```

Настройки таргета завершены, выходим:

```
/> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup/.
Configuration saved to /etc/target/saveconfig.json
```

Запускаем таргет и добавляем в автозагрузки:

```
[user@localhost ~]$ sudo systemctl start target
[sudo] password for user:
[user@localhost ~]$ sudo systemctl enable target
```

Смотрим статус, все хорошо:

```
[user@localhost ~]$ sudo systemctl status target
● target.service - Restore LIO kernel target configuration
     Loaded: loaded (/usr/lib/systemd/system/target.service; enabled; preset: disabled)
     Active: active (exited) since Sun 2024-02-25 14:59:41 +06; 21s ago
   Main PID: 1625 (code=exited, status=0/SUCCESS)
        CPU: 197ms

Feb 25 14:59:41 localhost.localdomain systemd[1]: Starting Restore LIO kernel target configurat>
Feb 25 14:59:41 localhost.localdomain systemd[1]: Finished Restore LIO kernel target configurat>
[user@localhost ~]$
```

Смотрим список открытых сетевых соединений и прослушивающих портов в системе. Порт 3260 «слушается» на всех адресах:

```
[user@localhost ~]$ netstat -tulpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name

tcp        0      0 0.0.0.0:3260            0.0.0.0:*               LISTEN      -

tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -

tcp6       0      0 :::22                   :::*                    LISTEN      -

udp        0      0 127.0.0.1:323           0.0.0.0:*                           -

udp6       0      0 ::1:323                 :::*                                -

udp6       0      0 fe80::a00:27ff:fec6:546 :::*                                -

[user@localhost ~]$
```

Файоволл может блокировать соединения к таргету, добавим новое правило (разрешение входящих TCP-соединений на порт 3260):

```
[user@localhost ~]$ sudo firewall-cmd --add-port=3260/tcp --permanent
[sudo] password for user:
success
[user@localhost ~]$ sudo systemctl reload firewalld
[user@localhost ~]$
```

```
[user@localhost ~]$ sudo systemctl restart firewalld
[user@localhost ~]$
```

Настройка клиента (Инициатора)

iscsi-initiator-utils - это набор утилит и библиотек для работы с iSCSI в Linux. Они предоставляют средства для инициации (инициатор) iSCSI-соединений, которые позволяют Linux-системе подключаться к удаленным iSCSI-устройствам (iSCSI target), таким как хранилища данных.

Устанавливаем нужную утилиту:

```
[user@localhost ~]$ sudo dnf install iscsi-initiator-utils
Last metadata expiration check: 1:10:10 ago on Sun 25 Feb 2024 02:00:02 PM +06.
Package iscsi-initiator-utils-6.2.1.9-1.gita65a472.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[user@localhost ~]$
```

Редактируем файл initiatorname.iscsi, указываем имя, которое добавили в acl при настройке таргета:

```
[user@localhost ~]$ sudo nano /etc/iscsi/initiatorname.iscsi
```

```
  GNU nano 5.6.1              /etc/iscsi/initiatorname.iscsi                Modified
InitiatorName=iqn.2024-02.com.example:client1
```

Запускаем службу, добавляем ее в автозапуск и проверяем статус:

```
[user@localhost ~]$ sudo systemctl start iscsi
[user@localhost ~]$ sudo systemctl enable iscsi
[user@localhost ~]$ sudo systemctl status iscsi
● iscsi.service - Login and scanning of iSCSI devices
     Loaded: loaded (/usr/lib/systemd/system/iscsi.service; indirect; preset: enabled)
     Active: active (exited) since Sun 2024-02-25 15:36:26 +06; 19s ago
       Docs: man:iscsiadm(8)
             man:iscsid(8)
   Main PID: 1587 (code=exited, status=21)
        CPU: 9ms

Feb 25 15:36:26 localhost.localdomain systemd[1]: Starting Login and scanning of iSCSI devices...
Feb 25 15:36:26 localhost.localdomain iscsiadm[1587]: iscsiadm: No records found
Feb 25 15:36:26 localhost.localdomain systemd[1]: Finished Login and scanning of iSCSI devices.
[user@localhost ~]$
```

iscsiadm - это утилита командной строки в Linux, предназначенная для управления iSCSI-инициатором и настройки iSCSI-соединений. Она позволяет администраторам настраивать и управлять iSCSI-инициатором, управлять сеансами и соединениями iSCSI, а также выполнять другие административные задачи.

Вот некоторые основные команды и параметры **iscsiadm**:

1. **iscsiadm -m discovery -t st -p <target_IP>**: Выполняет процедуру обнаружения iSCSI-целей (targets) на указанном IP-адресе.
2. **iscsiadm -m node -T <target_name> -p <target_IP> --login**: Подключается к указанной iSCSI-цели, предоставляя ее имя и IP-адрес, и выполняет процедуру входа в систему.

3. **iscsiadm -m node -T <target_name> -p <target_IP> --logout**: Отключается от указанной iSCSI-цели.
4. **iscsiadm -m session**: Показывает список текущих iSCSI-сеансов.
5. **iscsiadm -m session -P 3**: Показывает подробную информацию о текущих iSCSI-сеансах, включая параметры конфигурации и статус подключения.
6. **iscsiadm -m node**: Показывает список iSCSI-устройств (инициаторов), к которым система может подключаться.
7. **iscsiadm -m iface**: Показывает список доступных сетевых интерфейсов, которые могут использоваться для iSCSI.

Нужно обнаружить доступные таргеты, смотрим примеры в мануале:

```
[user@localhost ~]$ man iscsiadm
```

```
EXAMPLES
       Discover targets at a given IP address:

              sh# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.1.10 --dis
              cover

       Login, must use a node record id found by the discovery:

              sh#   iscsiadm   --mode   node   --targetname   iqn.2001-05.com.doe:test   --portal
              192.168.1.1:3260 --login
```

Проводим поиск по образцу. Таргет найден:

```
[user@localhost ~]$ sudo iscsiadm -m discoverydb -t st -p 192.168.1.115 -D
[sudo] password for user:
192.168.1.115:3260,1 iqn.2024-02.com.example:server1-disk1
```

Логинимся так же по образцу, успешно:

```
[user@localhost ~]$ sudo iscsiadm -m node -T iqn.2024-02.com.example:server1-disk1 -l
Logging in to [iface: default, target: iqn.2024-02.com.example:server1-disk1, portal: 192.168.1.115,3
260]
Login to [iface: default, target: iqn.2024-02.com.example:server1-disk1, portal: 192.168.1.115,3260]
successful.
[user@localhost ~]$
```

Проверяем список доступных блочных устройств. Добавился новый диск sdb:

```
[user@localhost ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   40G  0 disk
├─sda1        8:1    0    1G  0 part /boot
└─sda2        8:2    0   39G  0 part
  ├─cs-root 253:0    0 36.9G  0 lvm  /
  └─cs-swap 253:1    0    2G  0 lvm  [SWAP]
sdb           8:16   0    4G  0 disk
sr0          11:0    1 1024M  0 rom
```

Нужно его подготовить, устанавливаем dosfstools (пакет программного обеспечения для работы с файловыми системами в Linux.)

```
[user@localhost ~]$ sudo yum install dosfstools
Last metadata expiration check: 1:49:00 ago on Sun 25 Feb 2024 03:42:07 PM +06.
Package dosfstools-4.2-3.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Форматируем диск в ext4:

```
[user@localhost ~]$ sudo mkfs.ext4 /dev/sdb
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 1048576 4k blocks and 262144 inodes
Filesystem UUID: e4d536fb-5efa-43a5-bca8-8928b2608138
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[user@localhost ~]$
```

Создадим подкаталог newdisk в каталоге /mnt и смонтируем в него диск:

```
[user@localhost ~]$ sudo mkdir /mnt/newdisk
[user@localhost ~]$ sudo mount /dev/sdb /mnt/newdisk
[user@localhost ~]$ ls /mnt/newdisk
lost+found
```

Перейдем в диск, создадим файл, изменим его права доступа:

```
[user@localhost ~]$ cd /mnt/newdisk
[user@localhost newdisk]$ ls
lost+found
[user@localhost newdisk]$ touch newfile
[user@localhost newdisk]$ ls -l
total 16
drwx------. 2 root root 16384 Feb 26 17:40 lost+found
-rw-r--r--. 1 root root     0 Feb 26 17:43 newfile
[user@localhost newdisk]$ sudo chmod 666 newfile
[user@localhost newdisk]$ ls -l
total 16
drwx------. 2 root root 16384 Feb 26 17:40 lost+found
-rw-rw-rw-. 1 root root     0 Feb 26 17:43 newfile
```

Проверяем, что с файлом можно работать как обычно:

```
[user@localhost newdisk]$ cat > newfile
test
hello
[user@localhost newdisk]$ cat newfile
test
hello
```

Настройка iscsi завершена.