Утвержден

РДЦП.10001-01-УД

ПК СВ «БРЕСТ»

Руководство администратора. Часть 2

Средства обеспечения отказоустойчивости, масштабирования и виртуализации дисковых пространств

РДЦП.10001-01 95 01-2

Листов 39

2016

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

Настоящий документ является руководством администратора программного комплекса «Средства виртуализации «Брест» (ПК СВ «Брест») РДЦП.10001-01 (далее по тексту — ПК).

В документе приведено описание основных компонентов ПК «Брест» в части обеспечения отказоустойчивости, масштабирования и виртуализации дисковых пространств. Описаны шаги по их установки, настройки и порядка применения с учетом особенностей операционной системы специального назначения «Astra LInux Special Edition» РУСБ.10015-01 (далее по тексту — ОС СН), под управлением которой функционирует ПК.

СОДЕРЖАНИЕ

1. СРЕДСТВА ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ

1.1. Pacemaker и Corosync

Pacemaker и Corosync — это набор программного обеспечения для построения кластерных систем высокой доступности. Основные особенности:

– обнаружение и восстановление сбоев на уровне узлов и сервисов;

– независимость от подсистемы хранения;

– независимость от типов ресурсов: все что может быть заскриптовано, может быть кластеризовано;

– поддержка кластеров любого размера;

– поддержка и кворумных и ресурсозависимых кластеров;

– поддержка практически любой избыточной конфигурации;

– автоматическая репликация конфига на все узлы кластера;

– возможность задания порядка запуска ресурсов, а также их совместимости на одном узле;

– поддержка расширенных типов ресурсов: клонов (запущен на множестве узлов) и с дополнительными состояниями (master/slave и т.п.).

1.1.1. Установка

Для установки Pacemaker и Corosync необходимо выполнить следующее:

1) На каждом сервере отказоустойчивого кластера необходимо установить следующий пакет:

```
apt-get install pacemaker
```

2) На каждом сервере необходимо разрешить автозапуск corosync. Для этого в конфигурационном файле /etc/default/corosync следует указать параметр:

```
START=yes
```

1.1.2. Пример реализации

Настройка Pacemaker и Corosync будет рассмотрена на примере двух серверов с ОС СН. Для настройки необходимо выполнить следующий порядок действий:

1) На первом (главном) сервере нужно создать ключ и поменять на него права:

```
corosync-keygen
chown root:root /etc/corosync/authkey
chmod 400 /etc/corosync/authkey
```

2) Скопировать ключ /etc/corosync/authkey на другой сервер кластера в папку /etc/corosync.

3) Привести конфигурационный файл /etc/corosync/corosync.conf на всех серверах к одинаковому виду. Ниже представлен пример файла конфигурации для

кластера, состоящего из двух серверов: server-1 и server-2:

```
totem {
version: 2

# How long before declaring a token lost (ms)
token: 3000

# How many token retransmits before forming a new configuration
token_retransmits_before_loss_const: 10

# How long to wait for join messages in the membership protocol (ms)
join: 60

# How long to wait for consensus to be achieved before starting a new
# round of membership configuration (ms)
consensus: 3600

# Turn off the virtual synchrony filter
vsftype: none

# Number of messages that may be sent by one processor on receipt
# of the token
max_messages: 20

# Limit generated nodeids to 31-bits (positive signed integers)
clear_node_high_bit: yes

# Disable encryption
secauth: off

# How many threads to use for encryption/decryption
threads: 0

# Optionally assign a fixed node id (integer)
# nodeid: 1234

# This specifies the mode of redundant ring, which may be none,
# active, or passive.
```

```
rrp_mode: active
transport: udpu

interface {
member {
memberaddr: 192.168.122.10
}

member {
memberaddr: 192.168.122.11
}

ringnumber: 0
bindnetaddr: 192.168.122.0
mcastport: 5405
ttl: 1
}

interface {
member {
memberaddr: 192.168.25.8
}

member {
memberaddr: 192.168.25.9
}

ringnumber: 1
bindnetaddr: 192.168.25.0
mcastport: 5407
ttl: 1
}
}

amf {
mode: disabled
}
```

```
service {
# Load the Pacemaker Cluster Resource Manager
ver:       0
name:      pacemaker
}

aisexec {
user:   root
group:  root
}

logging {
syslog_priority: warning

fileline: off
to_stderr: no
to_logfile: yes
logfile: /var/log/corosync/corosync.log
logfile_priority: notice

to_syslog: no
syslog_facility: daemon
debug: off
timestamp: on
logger_subsys {
subsys: AMF
debug: off
tags: enter|leave|trace1|trace2|trace3|trace4|trace6
}
}
```

В указанной конфигурации, в секции `interface`, описаны по два сетевых интерфейса каждого сервера.

4) На каждом сервере необходимо запустить сервис corosync:

```
/etc/init.d/corosync start
```

Для управления кластером `pacemaker` имеется утилита `crm`. Например, проверить статус кластера можно с помощью команды:

```
crm status
```

Посмотреть текущую конфигурацию можно с помощью команды:

```
crm configure show
```

Результат проверки статуса кластера в примере будет таким:

```
============
Last updated: Thu Jun 16 10:21:43 2016
Last change: Wed Jun  8 12:56:36 2016 via cibadmin on server-2
Stack: openais
Current DC: server-1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
0 Resources configured.
============

Online: [ server-1 server-2 ]
```

5) Для нашего кластера, состоящего из 2 узлов, необходимо выставить следующие базовые настройки, выполнив команды:

```
crm configure property stonith-enabled="false"
crm configure property symmetric-cluster="false"
crm configure rsc_defaults resource-stickiness="110"
crm configure rsc_defaults migration-threshold=3
crm configure property no-quorum-policy=ignore
```

Настройка кластера завершена. Теперь необходимо настроить отказоустойчивость того или иного сервиса.

С точки зрения кластера все сущности, которые используются — сервисы, службы, точки монтирования, тома и разделы — это ресурсы, поэтому в данном руководстве под словом «ресурс» понимается все, что находится под управлением кластера. Ниже по тексту будут представлены конкретные примеры организации таких отказоустойчивых сервисов, как `Samba` и `NFS`. В качестве основы используется два сервера с четырьмя сетевыми интерфейсами, причем два из них объединены в один логический интерфейс (bonding). Имя первого сервера — server-1, имя второго — server-2. Оба сервера «видят» друг друга по имени (должен быть настроен DNS или быть соответствующие записи в файлах `/etc/hosts`). Настройка отказоустойчивых Samba и NFS состоит из 4 этапов, причем первые 3 одинаковы:

1) Настройка сети.
   а) На оба сервера необходимо установить пакет `ifenslave-2.6` (для обеспечения бондинга):

   ```
   apt-get install ifenslave-2.6
   ```

   б) Указать настройки для сетевых интерфейсов (`/etc/network/interfaces`):

Первый сервер:

```
auto lo
iface lo inet loopback


auto eth0
iface eth0 inet static
      address 10.10.10.99
netmask 255.255.255.0
gateway 10.10.10.2

auto eth1
iface eth1 inet static
     address 192.168.25.8
     netmask 255.255.255.0

auto bond0
iface bond0 inet static
     address 192.168.122.10
         netmask 255.255.255.0
        gateway 192.168.122.1
     bond_mode balance-rr
     bond_miimon 100
     bond_downdelay 200
     bond_updelay 200
     slaves eth2 eth3
```

Второй сервер:

```
auto lo
iface lo inet loopback


auto eth0
iface eth0 inet static
      address 10.10.10.100
netmask 255.255.255.0
gateway 10.10.10.2

auto eth1
```

```
iface eth1 inet static
     address 192.168.25.9
     netmask 255.255.255.0

auto bond0
iface bond0 inet static
     address 192.168.122.11
         netmask 255.255.255.0
        gateway 192.168.122.1
     bond_mode balance-rr
     bond_miimon 100
     bond_downdelay 200
     bond_updelay 200
     slaves eth2 eth3
```

Интерфейс `eth0` служит для доступа к отдельным узлам кластера, `eth1` — это heartbeat-интерфейс (по которому будет осуществляться доступ к сервису), а `bond0` — это логический интерфейс (бондинг) на базе физических `eth2`, `eth3` используется для синхронизации DRBD-ресурсов.

в) На обоих серверах необходимо выполнить следующие команды:

```
modprobe bonding
echo 'bonding' >> /etc/modules
init 6
```

2) Установка и настройка DRBD. Описание действий указано в пунктах 3.1.1.1 и 3.1.1.2 данного руководства.

3) Установка и настройка Pacemaker и Corosync. Описание действий указано в пунктах 1.1.1 и 1.1.2 данного руководства

4) Настройка отказоустойчивости сервиса (Samba/NFS).

а) Отказоустойчивый Samba-сервер

Перед настройкой ресурсов кластера на обоих серверах необходимо убрать из автозагрузки скрипты, которые в дальнейшем будут контролироваться corosync:

```
update-rc.d -f drbd remove
update-rc.d -f samba remove
```

Для настройки отказоустойчивости Samba-сервера, необходимо создать определенные ресурсы, объединить их в группы и определить порядок их запуска. Все действия нужно делать на первом сервере.

1) Необходимо создать ресурс DRBD раздела:

```
crm configure primitive drbd_main ocf:linbit:drbd params \
```

```
drbd_resource="main" op monitor interval="60s" op start \
interval="0" timeout="240s" op stop interval="0" timeout="240s" \
op monitor role=Master interval="10s" op monitor role=Slave \
interval="30s"
```

2) Следует указать определение Мастер/Слейв для созданного drbd-ресурса:

```
crm configure ms ms_drbd_main drbd_main meta master-max="1" \
master-node-max="1" clone-\ max="2" clone-node-max="1" \
notify="true"
```

3) Нужно создать ресурс точки монтирования drbd-раздела. В примере используем папку `/opt/main`:

```
crm configure primitive fs_main ocf:heartbeat:Filesystem params \
device="/dev/drbd0"\  directory="/opt/main" fstype="ext4" \
options="noatime"
```

4) Необходимо создать ресурс отказоустойчивого ip-адреса (по нему будет идти обращение к Samba-серверу):

```
crm configure primitive failover_ip ocf:heartbeat:IPaddr2 params \
ip=192.168.25.10 nic=eth1
```

5) Необходимо создать ресурс samba:

```
crm configure primitive samba lsb:samba
```

6) Нужно организовать созданные ресурсы служб/сервисов в группы:

```
crm configure group group_main fs_main failover_ip samba
```

7) Необходимо указать правило монтирования — монтироваться ресурсы будут на том узле, где drbd-ресурсы находятся в состоянии Мастер. Назовем этот ресурс `cluster_resources`:

```
crm configure colocation cluster_resources \
inf: fs_main ms_drbd_main:Master
```

8) Нужно указать порядок старта сервисов: стартовать после старта соответственных drdb-разделов. Назовем этот ресурс `all_after_drbd`:

```
crm configure order all_after_drbd \
inf: ms_drbd_main:promote group_main:start
```

9) Необходимо установить предпочтение запуска ресурсов Мастер/Слейв

```
crm configure location prefer_main_server-1 \
ms_drbd_main 100: server-1
crm configure location prefer_main_server-2 \
ms_drbd_main 50: server-2
crm configure location prefer_group_main_server-1 \
group_main 100: server-1
crm configure location prefer_group_main_server-2 \
```

```
group_main 50: server-2
```

Из вышеуказанных настроек видно, что основным сервером для запуска службы Samba является server-1.

Настройка завершена, проверить состояние можно командой `crm status`. Результат проверки будет примерно таким:

```
============
Last updated: Thu Jun 16 10:58:33 2016
Last change: Thu Jun 16 10:58:10 2016 via cibadmin on server-2
Stack: openais
Current DC: server-1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
5 Resources configured.
============

Online: [ server-1 server-2 ]

 Master/Slave Set: ms_drbd_main [drbd_main]
     Masters: [ server-1 ]
     Slaves: [ server-2 ]
 Resource Group: group_main
     fs_main    (ocf::heartbeat:Filesystem):  Started server-1
     failover_ip    (ocf::heartbeat:IPaddr2): Started server-1
     samba    (lsb:samba):    Started server-1
```

10) Осталось настроить Samba-сервер. В примере — на каждом сервере указать настройки для директории `/opt/main` в файле `/etc/samba/smb.conf`:

```
  [share-drbd]
       comment = Astra Linux Samba Share
      path = /opt/main
       browsable = yes
       guest ok = yes
       read only = no
       create mask = 0755
```

б) Отказоустойчивый NFS-сервер

Перед настройкой ресурсов кластера на обоих серверах необходимо убрать из

автозагрузки скрипты, которые в дальнейшем будут контролироваться corosync:

```
update-rc.d -f drbd remove
update-rc.d -f nfs-common remove
update-rc.d -f nfs-kernel-server remove
```

1) Необходимо создать ресурс DRBD раздела:

```
crm configure primitive drbd_main ocf:linbit:drbd params \
drbd_resource="main" op monitor interval="60s" op start \
interval="0" timeout="240s" op stop interval="0" timeout="240s" \
op monitor role=Master interval="10s" op monitor role=Slave \
interval="30s"
```

2) Необходимо указать определение Мастер/Слейв для созданного drbd-ресурса:

```
crm configure ms ms_drbd_main drbd_main meta master-max="1" \
master-node-max="1" clone-\ max="2" clone-node-max="1" \
notify="true"
```

3) Нужно создать ресурс точки монтирования drbd-раздела. В примере используем папку `/opt/main`:

```
crm configure primitive fs_main ocf:heartbeat:Filesystem \
params device="/dev/drbd0"\  directory="/opt/main" \
fstype="ext4" options="noatime"
```

4) Необходимо создать ресурс отказоустойчивого ip-адреса (по нему будет идти обращение к NFS-серверу):

```
crm configure primitive failover_ip ocf:heartbeat:IPaddr2 \
params ip=192.168.25.10 nic=eth1
```

5) Следует создать ресурсы nfs:

```
crm configure primitive nfs-kernel-server lsb:nfs-kernel-server
crm configure primitive nfs-common lsb:nfs-common
```

6) Нужно организовать созданные ресурсы служб/сервисов в группы:

```
crm configure group group_main fs_main failover_ip \
nfs-kernel-server nfs-common
```

7) Необходимо указать правило монтирования: монтироваться ресурсы будут на том узле, где drbd-ресурсы находятся в состоянии Мастер. Назовем этот ресурс `cluster_resources`:

```
crm configure colocation cluster_resources \
inf: fs_main ms_drbd_main:Master
```

8) Нужно указать порядок старта сервисов: стартовать после старта соответственных drdb-разделов. Назовем этот ресурс `all_after_drbd`:

```
crm configure order all_after_drbd \
```

```
inf: ms_drbd_main:promote group_main:start
```

9) Необходимо установить предпочтение запуска ресурсов Мастер/Слейв:

```
crm configure location prefer_main_server-1 \
ms_drbd_main 100: server-1
crm configure location prefer_main_server-2 \
ms_drbd_main 50: server-2
crm configure location prefer_group_main_server-1 \
group_main 100: server-1
crm configure location prefer_group_main_server-2 \
group_main 50: server-2
```

Из вышеуказанных настроек видно, что основным сервером для запуска службы NFS является server-1.

Настройка завершена, проверить состояние можно командой `crm status`. Результат проверки будет примерно таким:

```
============
Last updated: Sun Jun 19 21:20:06 2016
Last change: Sun Jun 19 21:18:23 2016 via cibadmin on
server-2
Stack: openais
Current DC: server-1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
6 Resources configured.
============
Online: [ server-1 server-2 ]

 Master/Slave Set: ms_drbd_main [drbd_main]
     Masters: [ server-1 ]
     Slaves: [ server-2 ]
 Resource Group: group_main
     fs_main    (ocf::heartbeat:Filesystem):    Started server-1
     failover_ip        (ocf::heartbeat:IPaddr2):  Started server-1
     nfs-kernel-server  (lsb:nfs-kernel-server):   Started server-1
     nfs-common (lsb:nfs-common):       Started server-1
```

10) Осталось настроить сам NFS-сервер. Для этого необходимо на каждом сервере указать настройки для директории `/opt/main` в файле `/etc/exports`:

```
  /opt/main *(ro,sync)  # Только чтение для всех
```

1.2. Приложение pksvbrest

В ПК входит вспомогательный инструмент под названием «pksvbrest», позволяющий развернуть кластер высокой доступности виртуальных машин на базе Pacemaker и Corosync с использованием `virt-amanger` и `virsh` (речь о которых пойдет в следующей части руководства). Он позволяет без особых усилий организовать отказуостойчивость виртуальной машины.

Для его использования на всех узлах необходимо установить соответствующий пакет:

```
apt-get install pksvbrest
```

ВНИМАНИЕ! Предварительные требования:

– пользователь, от которого будет запускаться приложение, должен быть добавлен в группы `kvm` и `libvirt-admin`. Также должна быть возможность выполнять команды с помощью `sudo`;

– перед использованием `pksvbrest` необходимо произвести настройку кластера `pacemaker` (см. 1.1);

– перед добавлением отказоустойчивости ВМ, необходимо предоставить доступ на виртуальную машину пользователю `root`;

– в режим отказоустойчивости можно перевести только ВМ, образ которой доступен на всех узлах кластера.

Использование:

– Для просмотра списка доступных операций, необходимо выполнить в терминале: `pksvbrest`

– Создания кластера pksvbrest начинается с вызова команды `pksvbrest -c`. После успешного выполнения всех запрашиваемых действий, будет создан кластер из двух машин.

– Для добавления третьего узла необходимо воспользоваться командой `pksvbrest -a nodename`, где `nodename` — имя узла кластера `pacemaker` (добавляемый узел предварительно должен быть добавлен в кластер pacemaker, 1.1).

– Для создания отказоустойчивой ВМ, необходимо выполнить команду `pksvbrest -f VMNAME`, где `VMNAME` — имя виртуальной машины.

ВНИМАНИЕ! При создании кластера `pksvbrest` из более 2 машин, добавление следующего узла необходимо производить на каждом узле кластера отдельно (команда `pksvbrest -a nodename`)

2. СРЕДСТВА ЭФФЕКТИВНОГО МАСШТАБИРОВАНИЯ

2.1. HAProxy

HAProxy — программное обеспечение для обеспечения высокой доступности и балансировки нагрузки для TCP и HTTP-приложений, посредством распределения входящих запросов на несколько обслуживающих серверов. Возможности:

– периодическая проверка доступности обслуживающих (back-end) серверов, на которые перенаправляются запросы пользователей;

– несколько алгоритмов определения доступности сервера: tcp-check, http-check, mysql-check;

– балансировка HTTP / HTTPS / TCP-запросов между «живыми» серверами;

– возможность закрепления определенных клиентов за конкретными обслуживающими серверам (stick-tables);

– поддержка: IPv6 и UNIX sockets, HTTP/1.1 сжатие (deflate, gzip, libslz), SSL-шифрование, полная поддержка постоянного HTTP-соединения;

– поддержка переменных блоков и Lua-скриптов в конфигурации сервера;

– веб-интерфейс с актуальным состоянием и статистикой работы программы;

2.1.1. Установка

На основном сервере, который будет принимать запросы и перераспределять их по рабочим узлам, из репозитория ПК необходимо установить пакет haproxy:

```
apt-get install haproxy
```

2.1.2. Пример реализации

Настройка производиться в конфигурационном файле `/etc/haproxy/haproxy.cfg`. Конфигурирование HAProxy состоит из:

– «global» раздел конфигурационного файла, который устанавливает параметры для всего процесса;

– раздел «defaults» определяет параметры по умолчанию для всех остальных разделов и являеться обязательным;

– раздел «frontend» описывает набор интерфейсов для принятия соединений от клиентов;

– раздел «backend» описывает набор серверов, к которым будет подключаться прокси для переадресации входящих соединений;

– раздел «listen» описывает полный прокси в одном разделе (объединенное описание «frontend» и «backend»). Как правило, используется только для TCP трафика.

В таблице 1 представлены основные примеры значений параметров конфигурации и их описание:

Таблица 1 – Параметры конфигурационного файла `/etc/haproxy/haproxy.cfg`

Раздел	Параметр	Описание
global	**log «address» «facility»** [max level [min level]] (log 127.0.0.1 local0 notice)	Добавляет сервер системного журнала. **«facility»** — должен быть одним из 24 стандартных типов журналирования (`kern user mail daemon auth syslog lpr news uucp cron auth2 ftp ntp audit alert cron2 local0 local1 local2 local3 local4 local5 local6 local7`)
	maxconn «number» (maxconn 10000)	Устанавливает максимальное число одновременных подключений для каждого процесса.
	nbproc «number» (nbproc 2)	Задает количество процессов. По умолчанию только один процесс будет создан.
	daemon	Устанавливает режим работы демоном
	user	Пользователь от которого работает процесс
	group	Группа от которой работает процесс
	chroot /var/lib/haproxy	Устанавливает окружение процесса
defaults	**log global**	Включает в журналирование информацию о трафике
	mode http	Режим работы HAProxy. Может быть одним из двух вариантов: – `http` — в этом режиме происходит анализ Layer 7, подходит для балансировки http трафика; – `tcp` — балансировка любого трафика
	option dontlognull	Отключает логирование пустых конекшинов
	retries 3	Количество попыток определить состояние бэкенда после обрыва соединения
	option redispatch	Перераспределяет запросы после обрыва связи с одним из бэкендомм
	option httpclose	Закрывает пассивные соединения
	option forwardfor	Включает X-Forwarded-For для передачи IP клиента бэкенду
frontend	**frontend http**	Задает имя фронтенда.
	bind *:80	Указывает на каком IP и порту будет слушаться запросы.

Продолжение таблицы 1

Раздел	Параметр	Описание
backend	**backend sitecluster**	Указывает название бэкенда.
	balance (roundrobin/leastconn/static-rr/uri/source)	Настройка алгоритма балансировки. Поддерживает следующие алгоритмы: – `Round Robin` направляет новые подключения к следующему месту назначения в циклическом списке, который видоизменяется при помощи так называемого веса сервера, на основании которого идет распределение запросов. – `Least Connected` направляет новые подключения к серверу с наименьшим числом доступных соединений. Можно включить данную опцию при помощи **balance leastconn**. – `Static Round Robin` направляет новые подключения к следующему месту назначения в циклическом списке, который видоизменяется при помощи так называемого веса сервера, на основании которого идет распределение запросов. В отличие от стандартной реализации round robin нельзя изменить вес сервера на «лету». Изменение веса сервера требует перезагрузки HAProxy. Можно включить данную опцию при помощи **balance static-rr**. – `Source`. Выбирает сервер исходя из хеша, построенного на основе IP пользователя. Таким образом, пользователь всегда обращается к одному и тому же серверу.
	server srv-1.3.my.com 21.86.21.20:80 cookie site113ha check inter 2000 fall 3 minconn 30 maxconn 70 weight 100	Описание рабочего сервера. Включает: – "Название "IP: порт"; – "cookie site113ha— задание кукиса необходимого для правильной балансировки сессий клиентов; – "check inter 2000 fall 3— проверка каждые 2с, при наличии 3 ошибок считать сервер недоступным; – "minconn 30 maxconn 70— организация очереди запросов, ограничение не более 70 одновременно обрабатывавшихся запросов; – "weight 100— вес сервера от 1 до 100.
	stats enable	Включает статистику
	fullconn 200	Максимальное значение одновременных подключений

Окончание таблицы 1

Раздел	Параметр	Описание
listen	**listen stats-srv-3.my.com *:8180**	Описывает IP адрес и порт доступа к статистике.
	stats uri /stats	URL доступа к статистике
	stats realm Haproxy Statistics	Заголовок (title) страницы статистики
	stats show-legends	Отображает в статистике дополнительную информацию о параметрах
	stats refresh 5s	Указывает интервал автоматического обновления страницы статистики
	stats auth test:test	Устанавливает логин и пароль доступа к странице статистики

Ниже представлен пример конфигурационного файла для балансировщика сервера Apache:

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        maxconn 40000
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # Default ciphers to use on SSL-enabled listening sockets.
        # For more information, see ciphers(1SSL). This list is from:
        #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
        ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:
        ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:
        !aNULL:!MD5:!DSS
        ssl-default-bind-options no-sslv3
defaults
        log     global
        mode    http
```

```
        option  httplog
        option  dontlognull
        retries 3
        option  redispatch
        maxconn 2000
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

listen webcluster *:80
        mode    http
        balance leastconn
        option forwardfor
        cookie serv insert
        option httpclose

      server webserver1 192.168.25.11:80 cookie serv1 check
      server webserver2 192.168.25.12:80 cookie serv2 check
```

3. СРЕДСТВА ВИРТУАЛИЗАЦИИ ДИСКОВЫХ ПРОСТРАНСТВ

3.1. Организация сетевого RAID

ПК включает набор инструментов для организации сетевого RAID, в частности DRBD.

3.1.1. DRBD

DRBD (Distributed Replicated Block Device — распределенное реплицируемое блочное устройство) — это блочное устройство, предназначенное для построения отказоустойчивых кластерных систем. DRBD занимается полным отражением по сети всех операций с блочным устройством. DRBD - это сетевой RAID-1. Технология «DRBD» поддерживает только два узла, более сложные конструкции могут строиться с помощью использования drbd-устройства в качестве «локального» для еще одного drbd-устройства. Узлы могут работать в режиме первичного (primary) узла или вторичного (secondary). Вторичный хранит данные, но не позволяет осуществить к ним локальный доступ, первичный позволяет осуществить доступ. DRBD поддерживает режим «первичный — первичный», при котором возможен доступ к обоим узлам. DRBD работает локально на узле (то есть обеспечивает репликацию на удаленный узел содержимого локального блочного устройства). Для использования создается новое устройство, обычно /dev/drbdX (X — число). Для нормальной работы DRBD должен быть запущен на обоих узлах. Если узел имеет роль вторичного, то он имеет соответствующее drbd-устройство, но доступ к нему запрещен. Как только происходит повышение роли до первичного, доступ открывается. Большинство операций осуществляется с помощью утилиты drbdadm, хотя фактическая работа происходит на уровне ядра. Для повышения скорости и надежности работы, необходимо наличие отдельного сетевого интерфейса (или нескольких интерфейсов), с помощью которых будет происходить репликация между узлами.

3.1.1.1. Установка

Для установки поддержки технологии DRBD необходимо выполнить следующее:

1) Из репозитория ПК необходимо установить следующие пакеты:

```
apt-get install lvm2 drbd8-utils
```

2) После установки необходимо поднять модуль DRBD:

```
modprobe drbd
```

3.1.1.2. Пример реализации

Настройка DRBD будет рассмотрена на примере двух серверов с ОС СН. На каждом сервере имеется неиспользуемый раздел *sdb1*. Для настройки необходимо выполнить следующий порядок действий:

1) Подготовить разделы для синхронизации. Для этого воспользуемся технологией LVM. На каждом сервере необходимо создать логический том (назовем его *main*,

размер тома — 100 гигабайт):

```
pvcreate /dev/sdb1
vgcreate vg0 /dev/sdb1
lvcreate -L100G -n main vg0
```

2) На обоих серверах необходимо создать одинаковый файл конфигурации `/etc/drbd.d/main.res`. Ниже представлен пример файла настройки:

```
resource main {
protocol C;

disk {
fencing resource-only;
}

handlers {
fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
after-resync-target "/usr/lib/drbd/crm-unfence-peer.sh";
}

net {
after-sb-0pri discard-least-changes;
after-sb-1pri call-pri-lost-after-sb;
after-sb-2pri call-pri-lost-after-sb;
}
syncer {
rate 1000M;
}
on server-1
{
device /dev/drbd0;
disk /dev/vg0/main;
address 192.168.122.10:7794;
meta-disk internal;
}
on server-2
{
device /dev/drbd0;
disk /dev/vg0/main;
address 192.168.122.11:7794;
```

```
meta-disk internal;
}
}
```

В секции `syncer` указывается максимальная скорость синхронизации данных между узлами. В секциях `on server-1` и `on server-2` располагаются параметры каждого из серверов. В опции `disk` указываются синхронизируемые LVM-тома, а в опции `address` — ip-адрес и порт сервера, по которому будет проходить синхронизация.

3) Необходимо создать описанный ресурс DRBD, выполнив на обоих серверах следующие команды:

```
drbdadm create-md main
drbdadm up all
```

4) На первом сервере нужно сделать созданный DRBD-ресурс основным (*primary*):

```
drbdadm -- --overwrite-data-of-peer primary all
```

Проверить статус можно с помощью команды:

```
service drbd status
```

5) На первом сервере нужно создать файловую систему, выполнив команды:

```
mkfs.ext4 /dev/drbd0
tune2fs -m 0 /dev/drbd0
```

После этого необходимо дождаться синхронизации и блочное устройство `/dev/drbd0` готово к использованию.

3.2. Распределенные параллельные файловые системы с защитой от сбоев

Распределенные файловые системы, являющиеся параллельными и с защитой от сбоев, разделяют и реплицируют данные на многие сервера для высокой производительности и обеспечения целостности данных. Даже когда сервер дает сбой, данные не теряются. Данные файловые системы используются в высокоскоростных вычислениях и фокусируются на высокой доступности, масштабируемости и высокой производительности. ПК поддерживает следующие системы:

– Ceph;
– GlusterFS.

3.2.1. Ceph

Ceph — свободная программная объектная сеть хранения, обеспечивающая как файловый, так и блочный интерфейсы доступа. Может использоваться на системах, состоящих как из нескольких серверов, так и из тысяч узлов; встроенные механизмы продублированной репликации данных обеспечивают высокую живучесть системы, при добавлении или удалении новых узлов массив данных автоматически перебалансируется с учетом из-

менений. В Ceph обработка данных и метаданных разделена на различные группы узлов в кластере. Обработка производится на уровне пользователя, не требуя никакой особой поддержки от ядра операционных систем узлов. Ceph может работать поверх блочных устройств, внутри одного файла или используя существующую файловую систему узла (например, EXT4). Кластер хранения данных состоит из нескольких различных демонов программного обеспечения. Каждый из этих демонов отделен от других, заботится об уникальной функциональности Ceph и добавляет ценность для своих соответствующих компонент. Следующая диаграмма кратко выделяет функции каждого компонента Ceph:

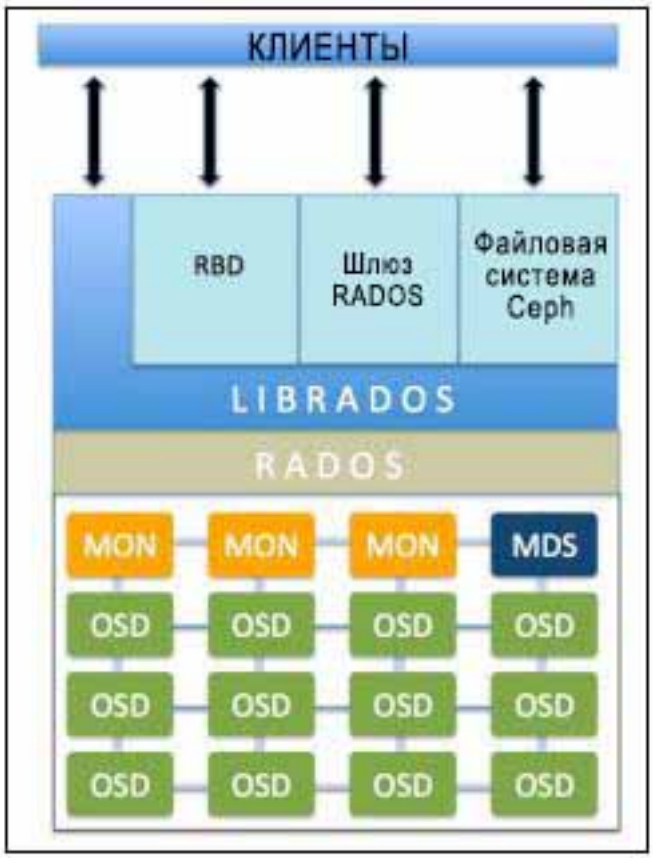

Рис. 1

Безотказное автономное распределенное хранилище объектов (**RADOS**, Reliable Autonomic Distributed Object Store) является основой хранения данных кластера Ceph. Все в Ceph хранится в виде объектов, а хранилище объектов RADOS отвечает за хранение этих объектов, независимо от их типа данных. Слой RADOS гарантирует, что данные всегда остается в согласованном состоянии и надежны. Для согласованности данных он выполняет репликацию данных, обнаружение отказов и восстановление данных, а также миграцию данных и изменение баланса в узлах кластера. Как только приложение выполняет операцию записи на кластер Ceph, данные сохраняются в виде объектов в устройстве хранения объектов Ceph (**OSD**, Object Storage Device). Это единственная составляющая кластера Ceph в которой хранятся фактические данные пользователя, и эти же данные получают-

ся клиентом, когда он выполняет операцию чтения. Как правило, один OSD демон связан с одним физическим диском кластера. Следовательно, обычно, общее число физических дисков в кластере Ceph совпадает с количеством демонов OSD, которые выполняют работу по хранению пользовательских данных в тесном взаимодействии со своим физическим диском.

Мониторы Ceph (**MON**, Ceph monitor) отслеживает состояние всего кластера путем хранения карты состояния кластера, которая включает в себя карты OSD, MON, PG и CRUSH. Все узлы кластера сообщают узлам монитора и делают общедоступной информацию обо всех изменениях в своих состояниях. Монитор поддерживает отдельную карту информации для каждого компонента. Монитор не хранит фактические данные; это является работой OSD.

Библиотека **librados** является удобным способом получения доступа к RADOS с поддержкой языков программирования PHP, Ruby, Python, C и C++. Она предоставляет собственный интерфейс для кластера хранения данных Ceph, RADOS, и является основанием для других служб, таких как RBD, RGW а также интерфейса POSIX для CephFS. librados API поддерживает прямой доступ к RADOS и позволяет создать свой собственный интерфейс к хранилищу кластера Ceph.

Блочное устройство Ceph (Ceph Block Device, известное также как RADOS block device, **RBD**) предоставляет блочное хранилище, которое может отображаться, форматироваться и монтироваться в точности как любой другой диск в сервере. Блочное устройство Ceph имеет функциональность корпоративных хранилищ, такую как динамичное выделение и моментальные снимки.

Сервер метаданных Ceph (**MDS**, Metadata Server отслеживает метаданные файловой иерархии и сохраняет их только для CephFS. Блочное устройство Ceph и шлюз RADOS не требуют метаданных, следовательно, они не нуждаются в демоне Ceph MDS. MDS не предоставляет данные непосредственно клиентам, тем самым устраняя единую точку отказа в системе.

Файловая система Ceph(**CephFS**, Ceph File System) предлагает POSIX- совместимую распределенную файловую систему любого размера. CephFS опирается на CephFS MDS, т.е. метаданные, для хранения иерархии.

3.2.1.1. Установка

Для установки Ceph, на каждом узле создаваемого кластера необходимо установить одноименный пакет:

```
apt-get install ceph
```

Кроме того на серверах требуется синхронизация времени:

```
apt-get install ntp ntpdate
```

3.2.1.2. Пример реализации

В данном подразделе содержится описание действий по настройке распределенного хранилища на базе Ceph на примере кластера из 3 узлов с ОС СН. Имена серверов соответственно: `astra-ceph1`, `astra-ceph2`, `astra-ceph3`. Все узлы должны видеть другу друга по имени, соответственно они должны быть прописаны на DNS сервере или же настроены соответствующие записи в файлах `/etc/hosts`. После установки необходимых пакетов, требуется соблюсти следующий порядок действий:

1) Произвести настройку синхронизации времени на серверах по протоколу `ntp`.

2) На каждом сервере создать единый конфигурационный файл `/etc/ceph/ceph.conf`. В примере файл настроек будет иметь следующий вид:

```
[global]
fsid = 912cfdf7-9ec9-4598-a14c-ea2553c3d916
auth cluster required = cephx
auth service required = cephx
auth client required = cephx

[osd]
osd journal size = 2000
osd mkfs type = ext4
osd mount options ext4 = rw,noatime

[mon.a]
host = astra-ceph1
mon addr = 192.168.25.11:6789

[mon.b]
host = astra-ceph2
mon addr = 192.168.25.12:6789

[mon.c]
host = astra-ceph3
mon addr = 192.168.25.13:6789

[osd.0]
host = astra-ceph1
devs = /dev/sdb1

[osd.1]
```

```
host = astra-ceph2
devs = /dev/sdb1

[osd.2]
host = astra-ceph3
devs = /dev/sdb1

[mds.a]
host = astra-ceph1
```

где `fsid = 912cfdf7-9ec9-4598-a14c-ea2553c3d916` — уникальный номер кластера Ceph. Его можно получить с помощью команды `uuidgen`. Как видно из конфигурации, на каждом узле используется раздел `sdb1`. Поэтому необходимо предварительно создать его на каждом сервере и отформатировать в EXT4.

3) На каждом узле из под пользователя root командой `ssh-keygen -t rsa` необходимо сгенерить ssh-ключи и настроить между тремя узлами взаимную авторизацию по ssh-ключам (проброс ключа на узел под именем «other-host» выполняется командой `ssh-copy-id root@other-host`).

4) Создать необходимые директории на каждом сервере:

```
mkdir /var/lib/ceph/osd/ceph-0
mkdir /var/lib/ceph/osd/ceph-1
mkdir /var/lib/ceph/osd/ceph-2
```

5) Выполнить команду создания хранилища (на первом узле):

```
mkcephfs -a -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.keyring --mkfs
```

6) `Созданный ключ /etc/ceph/ceph.keyring` скопировать на остальные узлы в `/etc/ceph/`.

7) На каждом узле проинициализировать свой osd:

- На astra-ceph1:

  ```
  ceph-osd -i 0 -c /etc/ceph/ceph.conf
  ```

- На astra-ceph2:

  ```
  ceph-osd -i 1 -c /etc/ceph/ceph.conf
  ```

- На astra-ceph3:

  ```
  ceph-osd -i 2 -c /etc/ceph/ceph.conf
  ```

8) На первом узле запустить сервис:

```
service ceph -a start
```

9) Проверить статус сервиса можно с помощью команды `ceph -s`.

10) Перегрузить сервера. Настройка завершена.

Для создания блочного устройства RBD, к примеру размером 2 гигабайта, необходимо воспользоваться командой:

```
rbd create ceph-rbd1 --size 2048
```

Посмотреть список созданных блочных устройств можно с помощью команды:

```
rbd ls
```

Для использования CephFS и RBD на клиенте, необходимо иметь установленные пакеты `ceph` и `ceph-fs-common`. Также нужно скопировать на клиента файл настроек `/etc/ceph/ceph.conf` и файл ключей `/etc/ceph/ceph.keyring`. Затем выполнить следующие действия:

```
ceph-authtool --name client.admin /etc/ceph/ceph.keyring --print-key |
tee /etc/ceph/admin.secret
```

Монтирование CephFS (к примеру в папку `/nmt/cephfs`) производится командой:

```
mount -t ceph astra-ceph1:6789,astra-ceph2:6789,astra-ceph3:6789:/ \
/mnt/cephfs -o name=admin,secretfile=/etc/ceph/admin.secret,noatime
```

3.2.2. GlusterFS

GlusterFS — это распределенная, параллельная, линейно масштабируемая файловая система с возможностью защиты от сбоев. GlusterFS может объединить хранилища данных, находящиеся на разных серверах, в одну параллельную сетевую файловую систему. GlusterFS работает в пользовательском пространстве при помощи технологии `FUSE`, поэтому не требует поддержки со стороны ядра операционной системы и работает поверх существующих файловых систем (`ext3`, `ext4`, `XFS` и т.п.). В отличие от Ceph, для работы GlusterFS не требуется отдельный сервер для хранения метаданных.

GlusterFS разделена на серверную и клиентскую части. На каждом сервере работает демон `glusterfsd` который делает доступным для клиентов локальное хранилище в качестве тома. Клиентский процесс `glusterfs` соединяется с одним или несколькими серверами посредством `TCP/IP` или `InfiniBand` и объединяет все доступные серверные тома в один, используя расширяемые трансляторы (функциональные модули системы). Получившийся том монтируется на клиентском хосте при помощи механизма Filesystem in Userspace (FUSE). Большая часть функциональности GlusterFS реализована в виде трансляторов (модулей). Использование необходимых трансляторов и их настройка позволяет гибко конфигурировать режим работы системы. Трансляторы реализуют следующую функциональность:

– синхронная репликация между серверами (нельзя расширить уже существующий том, добавив сервер для репликации);
– чередование порций данных между серверами (Striping);
– распределение файлов между серверами;

– балансировка нагрузки;

– восстановление после отказа узла (в ручном режиме с помощью опроса файлов (ls -lR или find на смонтированом томе));

– опережающее чтение (read-ahead) и запаздывающая запись (write-behind) для увеличения быстродействия;

– дисковые квоты.

Сервер GlusterFS реализован довольно просто: он предоставляет в пользование клиенту свое хранилище данных, оставляя за клиентом право решать каким образом организовать хранение. Все клиенты одного кластера должны быть настроены одинаково, во избежание проблем с консистентностью данных. Такая архитектура позволяет масштабировать GlusterFS до хранилищ, общий объем которых может измеряться петабайтами данных, используя аппаратное обеспечение средней производительности. Также, архитектура GlusterFS позволяет избежать узких мест, которые свойственны распределенным системам с более тесной модульной интеграцией.

Для работы GlusterFS не требуется отдельный сервер метаданных, что улучшает масштабируемость и надежность системы. Метаданные хранятся вместе с данными (в расширенных атрибутах файлов).

Поддерживаемые типы томов GlusterFS:

– **Distributed** — распределенный том. Тип тома, при котором данные распределяются равномерно (в произвольном порядке) по всем под томам. Например первый файл будет записан на первый сервер а второй файл — на третий. Тома такого типа очень хорошо и легко масштабируются, но никак не защищены средствами GlusterFS. Надежность Distributed тома необходимо обеспечить отдельно на аппаратном или программном уровне. В случае выхода из строя сервера или его дисков, данные находящиеся на нем будут не доступны. Самое интересное в этой схеме, что совершенно непредсказуемо, какие именно данные будут потеряны.

– **Replicated** — тома с репликацией. Аналогично `RAID 1`. В такой конфигурации одни и те же данные записываются минимум на два подтома. Более детальное разъяснение будет дано с примерами.

– **Striped** — том с чередованием. Аналогично `RAID 0`, наиболее производительный и одновременно самый ненадежный тип. Все поступающие данные разбиваются на части и параллельно пишутся на разные подтома на разных серверах. При считывании данные в обратном порядке собираются и отдаются клиенту. В результате выхода из строя одного сервера или его диска, том приходит в негодность до восстановления сбойного узла.

– **Distributed Striped** — распределение с чередованием. Здесь как и в случае с

Distributed-томом данные распределяются между разными серверами при этом они еще разбиваются на части между несколькими Striped-томами.

– **Distributed Replicated** — то же, что и Distributed Striped, только вместо чередования будет использоваться репликация. Этот вариант предоставляет такую же масштабируемость как и простой Distributed том, но при этом обладает повышенной надежностью за которую придется платить вдвое большим количеством серверов/дисков. Такая конфигурация рекомендуется разработчиками для высокопроизводительных сред, с повышенными требованиям к надежности.

3.2.2.1. Установка

На каждом узле кластера GlusterFS необходимо установить следующий пакет:

```
apt-get install glusterfs-server
```

3.2.2.2. Пример реализации

В данном подразделе содержится описание действий по настройке распределенного хранилища на базе GlusterFS на примере кластера из 3 узлов с ОС СН. Имена серверов соответственно: astra-gluster1, astra-gluster2, astra-gluster3. Все узлы должны видеть другу друга по имени, соответственно они должны быть прописаны на DNS сервере или же настроены соответствующие записи в файлах /etc/hosts.

После установки основного пакета, требуется выполнить следующее:

1) С первого узла необходимо подключится ко второму и третьему и тем самым инициализировать кластер:

```
gluster peer probe astra-gluster2
gluster peer probe astra-gluster3
```

Проверить статус сервера gluster можно с помощью команды.

```
gluster peer status
```

Результат команды, выполненной на первом сервере, будет таким:

```
Number of Peers: 2

Hostname: astra-gluster2
Uuid: 0a8b1faf-cf05-4241-926d-95d209cd7f56
State: Peer in Cluster (Connected)

Hostname: astra-gluster3
Uuid: 390e2c54-223a-4d01-b2e0-d7279b8c2dc0
State: Peer in Cluster (Connected)
```

2) Необходимо создать том. Для этого нужно предварительно подготовить место на винчестере. В примере, на каждом сервере используем отдельные разделы диска, смонтированные в /mny/dist. В качестве типа выберем реплицируемый (кол-во

реплик — 3,) и назовем его volumedata:

```
gluster volume create volumedata replica 3 transport tcp \
astra-gluster1:/mnt/dist astra-gluster2:/mnt/dist astra-gluster3:/mnt/dist
```

3) Необходимо запустить созданный том:

```
gluster volume start volumedata
```

Настройка завершена, получить информацию о `volume` можно с помощью команды:

```
gluster volume info
```

Результат будет таким:

```
Volume Name: volumedata
Type: Replicate
Volume ID: 5bb58232-70c2-4c26-9651-9ac4e32a64aa
Status: Started
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: astra-gluster1:/mnt/dist
Brick2: astra-gluster2:/mnt/dist
Brick3: astra-gluster3:/mnt/dist
```

Для использования созданного хранилища на клиенте, необходимо поставить соответствующий пакет:

```
apt-get install glusterfs-client
```

Оперативно смонтировать хранилище (к примеру, в папку /mnt/data) можно с помощью команды:

```
mount -t glusterfs astra-gluster1:/volumedata, \
astra-gluster2:/volumedata,astra-gluster3:/volumedata /mnt/data
```

Если же хранилище будет использоваться постоянно, можно его прописать в `/etc/fstab`

```
astra-gluster1:/volumedata,astra-gluster2:/volumedata,
astra-gluster3:/volumedata /mnt/data glusterfs rw,nosuid,
nodev,exec,nouser,async,_netdev  0 0
```

Теперь при записи в папку `/mnt/data`, данные будут реплицироваться на всех узлах кластера.

3.3. Средства интеграции и дифференциации блочных устройств

3.3.1. DRBD

Подробная информация по работе с DRBD представлена в разделе 3.1.1.

3.3.2. ISCSI

iSCSI — протокол, базирующийся на TCP/IP и разработанный для установления взаимодействия и управления системами хранения данных, серверами и клиентами. iSCSI описывает:

– транспортный протокол для SCSI, который работает поверх TCP>;

– механизм инкапсуляции SCSI команд в IP сети;

– протокол для нового поколения систем хранения данных, которые будут использовать «родной» TCP/IP.

Системы на основе iSCSI могут быть построены на любой достаточно быстрой физической основе, поддерживающей протокол IP, например `Gigabit Ethernet` или `10G Ethern`
Использование стандартного протокола позволяет применять стандартные средства контроля и управления потоком, а также существенно уменьшает стоимость оборудования по сравнению с сетями `Fibre Channel`.

Терминология iSCSI во многом основывается на терминологии, использующейся в `SCSI`:

– **initiator** — тот, кто устанавливает соединение с целью(target). Чаще всего это узел (в общем случае) осуществляет ввод/вывод на блочные устройства.

– **target** — экспортируемый объект. В зависимости от контекста цель (target) называют или целиком экспортирующий узел, или только экспортируемый объект. Сам объект может делиться на lun-ы.

– **portal** — группа целей (targets), которые анонсируются вместе. Чаще всего один узел хранения — один портал.

– **IQN** — полное имя участника взаимодействия. На практике существует iqn у инициатора и у цели (target).

– **endpoint** — уточненное имя ресурса, чаще всего включает в себя iqn, номер LUN-а и указание на конкретный метод доступа к нему (например, номер соединения, LUN и IP-адрес, с которого следует получать доступ к устройству).

– **LUN** (Logical Unit Number) — номер объекта внутри цели (target). Ближайшим аналогом является раздел диска или отдельный том.

В ПК в качестве таргета используется программный комплекс **Linux-IO Target (LIO)**, в качестве инициатора — **open-iscsi**.

3.3.2.1. Установка таргета

Для поддержки iSCSI-таргета `LIO` необходимо установить следующий пакет:

```
apt-get install targetcli
```

Для управления и настройки таргета, имеется своя командная оболочка, доступ к которой можно получить, выполнив в терминале команду `targetcli`. Появится специаль-

ная командная строка с ограниченным набором инструкций для настройки ISCSI-таргетов. Targetcli на представляет таргет в виде иерархически построенной структуры и делит таргет на back-end и front-end части. На рисунке 2 ниже представлена структура таргета.

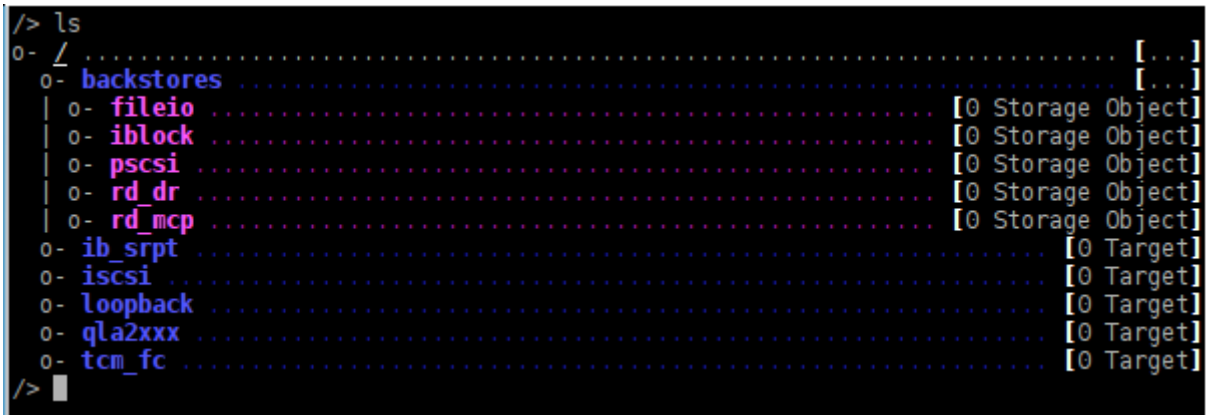

Рис. 2

Структура таргета включает:

– **Backstores** — это раздел, в котором отображаются объекты-хранилища. Backstores не видимая снаружи часть iSCSI таргета.

– **fileio** — файл.

– **iblock** —жесткий диск, символические ссылки на диск или LVM.

– **pscsi** (SCSI pass-through) — любое устройство для хранения информации, поддерживающее SCSI команды напрямую, то есть без эмуляции SCSI. Не рекомендуется к использованию, так может привести к повреждению оборудования.

– **iSCSI** — это видимая снаружи часть iSCSI таргета. Он содержит список таргетов, лунов, права доступа и т.п.

– **loopback** — модуль, предоставляющий доступ к таргету локально.

В таблице 2 представлен описание основных команд `targetcli`.

Таблица 2 – Список команд `targetcli`

Команда	Описание
help [topic]	Помощь
ls [path] [depth]	Вывод на экран структуры таргета `path` глубиной `depth`
pwd	Вывод на экран текущего пути дерева
get [group] [parameter...]	Получить значение параметра группы
set [group] [parameter=value...]	Установить значение параметра в группе
saveconfig	Сохранить конфигурацию
exit	Покинуть командную оболочку `targetcli`

3.3.2.2. Пример реализации таргета

Настройка будет рассмотрена на примере сервера с ОС СН (ip-адрес: `192.168.25.10`). В качестве раздаваемого блочного устройства выступает диск `/dev/vda`. В командной оболочке `targetcli` необходимо выполнить следующие шаги:

1) Настроить авторизацию для возможности поиска iSCSI-таргетов в сети:

```
/iscsi set discovery_auth enable=1 userid=youuser password=youpass
```

где yuuser и youpass — логин и пароль соответственно.

2) Создать блочное устройство под названием `my-disk`:

```
/backstores/iblock/ create name=my-disk dev=/dev/vda
```

3) Создать непосредственно сам target (экспортируемый объект):

```
/iscsi create
```

Результат вывода команды ’ls /’ после выполнения вышеуказанных действий представлен на рисунке 3

Рис. 3

Внутри iscsi появилась структура:

– `iqn.2003-01.org.linux-iscsi.astra15-one-repo.x8664:sn.091ab28a79cf` — имя таргета;

– `tpgt1` (Target Portal Group) — список IP-адресов и TCP портов, которые будет слушать этот таргет;

– `acl` — список адресов, с которых можно будет подключиться к таргету;

– `luns` — список лунов;

– `portals` — список IP-адресов и портов, которые будет слушать таргет.

4) Необходимо создать Lun на основе созданного хранилища `my-disk`:

```
/iscsi/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
```

```
x8664:sn.091ab28a79cf/tpgt1/luns/
create /backstores/iblock/my-disk
```

5) Создать portal:

```
/iscsi/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/tpgt1/portals/  create
```

6) Включить авторизацию:

```
/iscsi/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/tpgt1/
set attribute authentication=1 demo_mode_write_protect=0
```

7) Создать acl-доступ для инициатора `iqn.1993-08.org.astra-client:01:c1a92326f6b8`:

```
/iscsi/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/tpgt1/acls/
create iqn.1993-08.org.debian:01:c1a92326f6b8
```

8) Установить логин (`iqn.1993-08.org.debian:01:c1a92326f6b8`) и пароль (`mysecretpass`) для подключения блочного устройства:

```
/iscsi/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/tpgt1/acls/iqn.
1993-08.org.debian:01:c1a92326f6b8/
set auth userid=iqn.1993-08.org.debian:01:c1a92326f6b8

/iscsi/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/tpgt1/acls/iqn.
1993-08.org.debian:01:c1a92326f6b8/
set auth password=mysecretpass
```

9) Сохранить настройки:

```
/ saveconfig
```

Результат вывода команды ’ls /’ представлен на рисунке 4

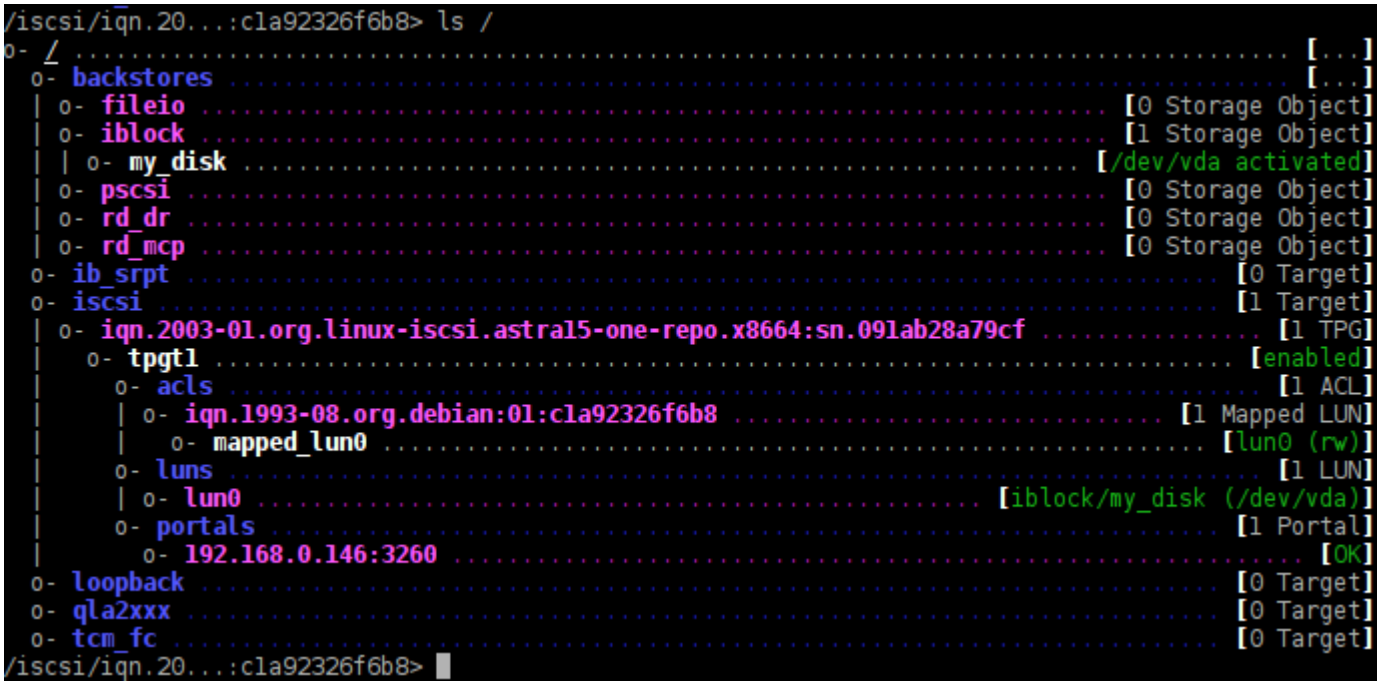

Рис. 4

3.3.2.3. Установка инициатора

Для подключения iSCSI-устройства необходимо установить следующий пакет:

```
apt-get install open-iscsi
```

3.3.2.4. Пример реализации инициатора

Порядок действий по установке и настройке инициатора будет рассмотрена на примере клиента с ip-адресом `192.168.25.11`:

1) После установки пакета, необходимо запустить одноименный сервис:

```
/etc/init.d/open-iscsi start
```

После первого запуска сгенерируется уникальный код инициатора, который можно увидеть в файле `/etc/iscsi/initiatorname.iscsi`:

```
InitiatorName=iqn.1993-08.org.debian:01:c1a92326f6b8
```

Именно это имя используется в ACL при настройке таргета (пункт 3.3.2.1 данного руководства).

2) В файл `/etc/iscsi/iscsid.conf` добавить/раскоментировать следующие сроки:

```
discovery.sendtargets.auth.authmethod = CHAP
discovery.sendtargets.auth.username = youuser
discovery.sendtargets.auth.password = youpass
```

где `youuser` и `youpass` — логин и пароль, указанные в пункте 3.3.2.1, в первом шаге настройки таргета.

3) Перезапустить сервис:

```
/etc/init.d/open-iscsi start
```

4) Выполнить сканирование для поиска ISCI-таргетов:

```
iscsiadm  -m discovery -t st -p  192.168.25.10
```

где 192.168.25.10 — ip-адрес таргета В результате появится папка

```
/etc/iscsi/nodes/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/192.168.25.11,3260,1
```

в которой в файле default

необходимо указать данные для авторизации на подключаемом ресурсе:

```
node.session.auth.authmethod = CHAP
node.session.auth.username = iqn.1993-08.org.debian:01:c1a92326f6b8
node.session.auth.password = mysecretpass
```

где iqn.1993-08.org.debian:01:c1a92326f6b8 и mysecretpass — логин и пароль для подключения iSCSI-устройства (пункт 3.3.2.1, шаг 6)

5) Снова перезапустить сервис:

```
/etc/init.d/open-iscsi start
```

6) На последнем шаге осталось подключить добавленное iSCSI-устройство. Подключить таргет можно с помощью команды терминала:

```
iscsiadm  -m node  --targetname \
"iqn.2003-01.org.linux-iscsi.astra15-one-repo.x8664:sn.091ab28a79cf" \
--portal "192.168.25.11:3260" --login
```

Для настройки автоматического подключения к ISCSI-устройству, необходимо в файле

```
/etc/iscsi/nodes/iqn.2003-01.org.linux-iscsi.astra15-one-repo.
x8664:sn.091ab28a79cf/192.168.25.11,3260,1/default
```

включить опцию:

```
node.startup = automatic
```

Если будет использоваться несколько таргетов, эту опцию можно указать по умолчанию в глобальном конфигурационном файле /etc/iscsi/iscsid.conf.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ЕПП — единое пространство пользователей

ОС СН — операционная система специального назначения

ПК — программный комплекс

ВМ — виртуальная машина

СЗИ — средства защиты информации

ФС — файловая система

DRBD — Distributed Replicated Block Device (распределённое реплицируемое блочное устройство)

RADOS — Reliable Autonomic Distributed Object Store (хранилище, отвечабщее за хранение объектов кластера Ceph, независимо от их типа данных)

OSD — Object Storage Device (основное устройстве хранения объектов Ceph, обычно связанное с одним физическим диском, в котором хранятся фактические данные пользователя)

MON — Monitor (демон, отслеживающий состояние кластера Ceph)

RBD — Rados block device (блочное хранилище кластера Ceph, которое может отображаться, форматироваться и монтироваться в точности как любой другой диск в сервере)

MDS — Metadata Server (сервер кластера Ceph, отслеживающий метаданные файловой иерархии для CephFS)

CephFS — Ceph File System (POSIX-совместивамя файловая система на базе кластера Ceph)

iSCSI — Internet Small Computer System Interface (протокол на базе TCP/IP для взаимодействия и управления системам хранения данных, серверов и клиентов)

LUN — Logical Unit Number (номер объекта внутри цели target)

ALD — Astra Linux Directory (единое пространство пользователей)

KVM — Kernel-based Virtual Machine (программное решение, обеспечивающее виртуализацию в среде Linux на платформе, которая поддерживает аппаратную виртуализацию на базе Intel VT (Virtualization Technology) либо AMD SVM (Secure Virtual Machine))

QEMU — Quick Emulator (средства эмуляции аппаратного обеспечения)

VDI — Virtual Desktop Infrastructure (инфраструктура виртуальных рабочих столов)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					