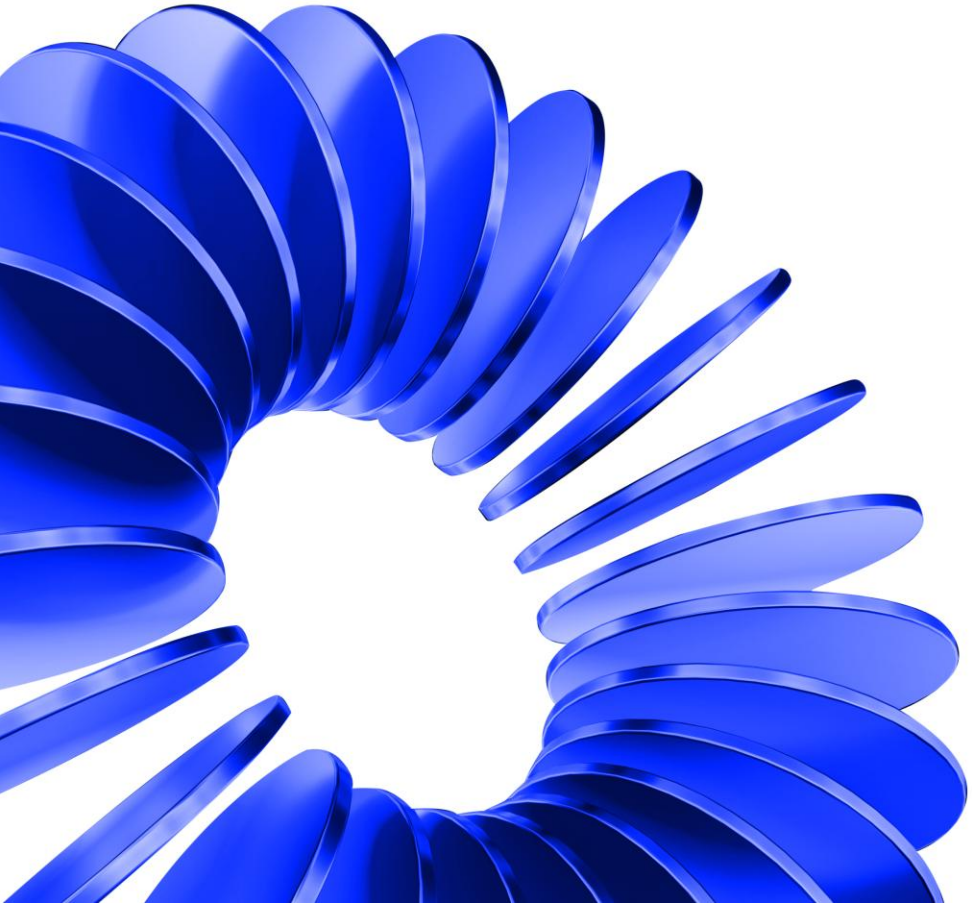
ОБЩЕСТВО С ОГРАНИЧЕННОЙ
ОТВЕТСТВЕННОСТЬЮ «РУСБИТЕХ-АСТРА»

СИСТЕМА ХРАНЕНИЯ ДАННЫХ TROK

РУКОВОДСТВО ПО НАСТРОЙКЕ



Москва, 2026г.

СОДЕРЖАНИЕ

1. ОБЩАЯ ИНФОРМАЦИЯ

Руководство по настройке иллюстрирует процесс создания кластерной системы хранения данных на примере конфигурации, состоящей из трёх узлов. Кластер представляет собой три физических сервера с установленной операционной системой Astra Linux. В кластере рекомендуется использовать не менее трёх серверов для обеспечения устойчивости к сбоям и достижения кворума при принятии решений. Подобная конфигурация минимизирует риск разделения сети и повышает общую отказоустойчивость системы.

Все серверы должны быть подключены к одной локальной сети (с единой подсетью), обеспечивая сетевое взаимодействие внутри единой инфраструктуры. Для корректного функционирования кластера необходимо использовать выделенную сеть для репликации данных. Такое решение обеспечивает изоляцию сетевых потоков приложений, функционирующих в организации, от сетевой нагрузки, создаваемой узлами кластера. Это позволяет минимизировать взаимное влияние между процессами и избежать деградации производительности из-за конкуренции за ресурсы сети.

Для обеспечения избыточности и повышенной отказоустойчивости системы репликация данных осуществляется между тремя узлами.

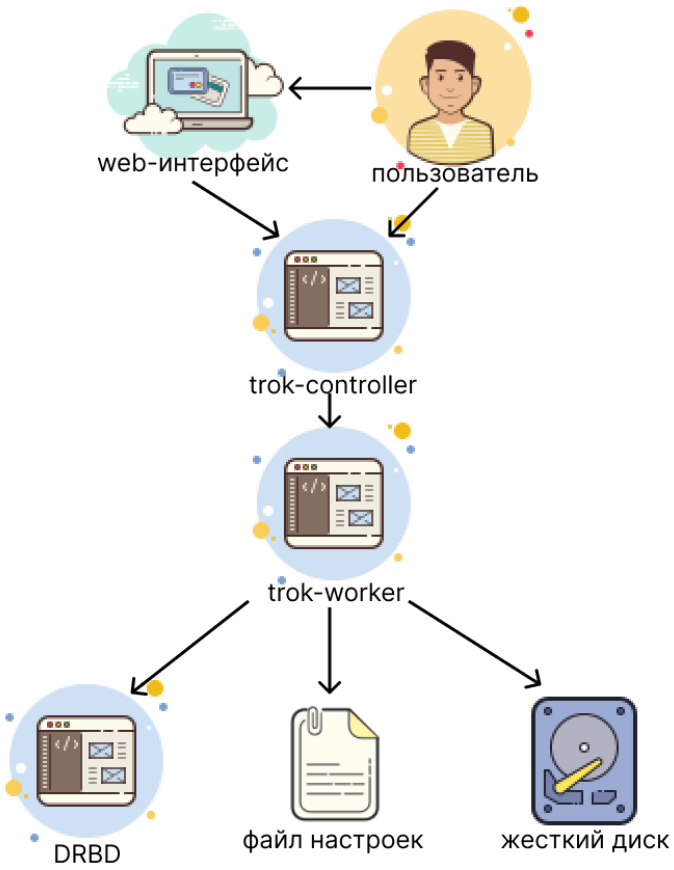


Рисунок 1 – Взаимодействие основных компонентов программного обеспечения

Техническая реализация программного обеспечения включает интеграцию четырех основных компонентов:

– trok-controller – центральный модуль, отвечающий за оркестрацию кластерных ресурсов и управление конфигурацией распределенного хранилища данных.

– trok-worker – служба, которая работает на каждом узле (сервере), где должны быть размещены реплицированные данные. Worker отвечает за управление локальными ресурсами хранения и взаимодействует с trok-controller.

– trok web GUI – веб-интерфейс для интуитивного администрирования функций системы хранения данных.

– trok-harbor – готовый программный модуль, который объединяет три ключевых элемента:

– drbd-reactor – событийно-ориентированная служба автоматизации для DRBD, которая в реальном времени отслеживает состояние узлов и ресурсов DRBD и автоматически выполняет заданные действия (сценарии агентов ресурсов) при изменении этих состояний, например, при переключении ролей Primary/Secondary в кластере, обеспечивая автоматическое управление ресурсами.

– resource-agents — это набор готовых сценариев (агентов OCF), которые используются в TROK для управления высокодоступными хранилищами на DRBD. Агенты позволяют автоматизировать контроль состояния ресурсов и обеспечивают отказоустойчивость сервисов доступа.

– сервисы доступа – поддержка протоколов: iSCSI, NFS, SMB.

trok-harbor **поставляется в формате отдельного мета-пакета** и должен быть установлен на всех узлах типа Worker.

1.1. Бездисковый режим

Diskless (бездисковый) режим – это режим работы DRBD, когда узел не имеет локального дискового хранилища, но участвует в кластере как узел доступа к DRBD-ресурсу.

Если выражаться более простым языком, Diskful-сервер (с дисками) – выступает в качестве аналога склада. Он хранит данные. Diskless-сервер (без дисков) может использоваться как некий пункт выдачи. Он не хранит, но даёт доступ к тому, что есть на складе.

Бездисковый узел может выполнять функции арбитра (решающего голоса) для поддержания кворума в кластере, предотвращая ситуации с разделением ресурсов, когда узлы не могут договориться о том, какие данные являются авторитетными. Он также может выступать в роли клиента, считывающего и записывающего данные с других узлов.

Преимущества режима Diskless:

– не требуется локальный диск на Diskless нодах;

– позволяет размещать Diskless узлы на любой платформе (например, на гипервизорах в конвергентных системах);

– позволяет быстро добавлять и удалять узлы;

– обеспечивает больше возможных точек доступа к данным;

– быстрый failover;

– не теряет связь с ресурсом, если вышел из строя диск на Diskful-узле.

Бездисковый узел имеет следующие особенности:

– нет локального хранения данных – только метаданные в оперативной памяти;

– зависимость от сети – при потере связи данные недоступны;

– требуется минимум 32MB RAM на ресурс для метаданных (32MB на 1 TB DRBD);

– метаданные хранятся в памяти – исчезают после перезагрузки;

– автоматическое восстановление при восстановлении связи.

1.2. Установка и настройка базовых компонентов ПО

Базовые компоненты программного обеспечения, необходимые, для работы системы указаны в таблице ниже:

Таблица 1 – Базовые компоненты системы

Компонент	Требования	Проверка
trok-controller	Версия, указанная в названии дистрибутива	systemctl status trok-controller
trok-worker	Версия, указанная в названии дистрибутива	systemctl status trok-worker
trok-cp-endpoint	Предназначен для подключения к API	systemctl status trok-cp-endpoint

	TROK. Версия, указанная в названии дистрибутива	
Ядро ОС	Версия 6.1	sudo uname -r
DRBD	Версия, совместимая с версией ядра. ≥ 9.0 Несовпадение версии ядра и версии DRBD может привести к неработоспособности хранения.	sudo apt list --installed | grep drbd Пример вывода (также см. примечание под таблицей): drbd-6.1.152-1-generic/stable,stable,now 9.2.12-4 amd64 [установлен] # Установлен связанный с ядром модуль DRBD версии 9.2.12. drbd-reactor/stable,now 1.4.1-1 amd64 [установлен] # Установлен компонент реактора для управления ресурсами. drbd-ueficert/now 9.2.11-1 all [установлен, локальный] # Установлен сертификат UEFI для DRBD. drbd-utils/stable,now 9.28.0-1 amd64 [установлен] # Установлены утилиты DRBD версии 9.28.0.
drbd-reactor	Версия > 1.4. Требует в качестве зависимости Python ≥ 3.6. Конфигурационные файлы в /etc/drbd-reactor/	Версия drbd-reactor будет указана в выводе предыдущей команды, но также существует отдельная команда для проверки: sudo drbd-reactor --version
Загруженные модули ядра DRBD	DRBD должен быть не просто установлен, а также загружен в ядро	sudo lsmod | grep -i drbd Пример вывода:

	ОС как модуль и готов к работе. Если в выводе строки drbd нет: DRBD не загружен – либо из-за ошибки, либо несовместимости. Нужно выполнить установочную команду trok-drbd-autoinstall от имени суперпользователя или разбираться с ошибкой ядра.	drbd 819200 0 # Основной модуль DRBD загружен (размер 819200 байт, используется 0 процессами). lru_cache 16384 1 drbd # Зависимый модуль LRU-кэша, используемый DRBD. libcrc32c 16384 6 nf_conntrack, nf_nat,dm_persistent_data,nf_tables,drbd, sctp # Модуль CRC32C, включая DRBD как одного из клиентов.
LVM	LVM version 2.x.x	sudo lvs --version
WEB-UI	Веб-сервер (Nginx) с HTTPS Порт 80 (или кастомный)	sudo systemctl status nginx
resource-agents	Набор OCF-скриптов для управления ресурсами кластера Поддержка агентов: DRBD, Filesystem, IPaddr2, portblock, nfsserver, exportfs, iSCSITarget,	sudo apt list --installed | grep resource-agents ls /usr/lib/ocf/resource.d/heartbeat/ # Проверка наличия основных OCF-агентов для DRBD и файловых систем ls /usr/lib/ocf/resource.d/heartbeat/ | grep -E "(DRBD|Filesystem|IPaddr2|portblock)" # Проверка наличия основных агентов для сервисов доступа

	iSCSILogicalUnit	ls /usr/lib/ocf/resource.d/heartbeat/ | grep -E "(nfsserver|exportfs|iSCSI|smb-share)"

Примечание. Вывод команды sudo apt list --installed | grep drbd зависит от того, на каком этапе установки был сделан запрос. Если вы установили trok-worker и trok-controller в результате выполнения команды вы получите только 2 строки:

```
   drbd-6.1.152-1-generic/stable,stable,now 9.2.12-4 amd64 [установлен] # Установлен связанный с ядром модуль DRBD версии 9.2.12.

   drbd-utils/stable,now 9.28.0-1 amd64 [установлен] # Установлены утилиты DRBD версии 9.28.0
```

Если помимо trok-worker и trok-controller вы установили trok-harbor в результате выполнения команды вы получите 3 строки:

```
   drbd-6.1.152-1-generic/stable,stable,now 9.2.12-4 amd64 [установлен] # Установлен связанный с ядром модуль DRBD версии 9.2.12.

   drbd-reactor/stable,now 1.4.1-1 amd64 [установлен] # Установлен компонент реактора для управления ресурсами.

   drbd-utils/stable,now 9.28.0-1 amd64 [установлен] # Установлены утилиты DRBD версии 9.28.0
```

Если вы выполнили все предыдущие условия и настроили сертификат, в результате выполнения команды вы получите вывод, который был указан в таблице.

1.3. Подготовка топологии сети и соблюдение аппаратных требований

Для комфортной работы желательно использовать минимум 3 серверные ноды, имеющие следующие характеристики:

- Процессор: 4+ ядер (x86_64), частота: от 2 ГГц;
- Размер оперативной памяти:
  - Минимальный объем: 8 ГБ;
  - Рекомендуемый объем: 64 ГБ и более;
- Тип оперативной памяти: DDR4, DDR5, ECC (защита от сбоев).
- Объем диска: 2 накопителя:

- Дисковое пространство для операционной системы и программного обеспечения емкостью 128 ГБ и более (SAS/SATA HDD/SSD);
- Дисковое пространство для хранения данных TROK (серверы хранения данных): 1 накопитель (и более) с емкостью не менее 1 ТБ (SAS/SATA HDD).;

– Для программного обеспечения TROK: 1 накопитель (и более) с емкостью не менее 128 ГБ (SAS/SATA HDD).

1.4. Сетевые интерфейсы

Рекомендации по скорости соединения:

– Минимум: 1 Гбит/с (1 GbE);

– Рекомендуется: 10 Гбит/с (10 GbE) и выше (особенно для репликации DRBD, чтобы избежать задержек при синхронизации данных между узлами)

Максимальный размер блока данных (в байтах), который может быть передан через сетевой интерфейс без фрагментации (размер MTU):

– Минимум: 1500 байт (стандартный Ethernet);

– Рекомендуется: Jumbo Frames (MTU = 9000) (критично для NVMe-oF, так как увеличивает пропускную способность и снижает накладные расходы при передаче данных).

Для обеспечения стабильной производительности и низкой задержки рекомендуется использовать раздельные сетевые пути для синхронизации данных DRBD и клиентского доступа. Это предотвращает конфликт трафика и повышает надежность системы.

1.5. Сетевые требования

Все ноды должны находиться в одной подсети без NAT (например, 192.168.100.1/24).

Каждое устройство должно иметь статический IP-адрес или использовать резервирование DHCP.

Рекомендуется настроить параметр MTU (Maximum Transmission Unit) на сетевых интерфейсах для оптимизации передачи данных и предотвращения фрагментации пакетов. Для высокоскоростных сетей (10 Гбит/с и выше) рекомендуется использовать значение MTU 9000 (Jumbo Frames) для повышения пропускной способности и снижения нагрузки на процессор. Для корректной настройки параметров обратитесь к официальной документации вашего дистрибутива, так как конфигурация может зависеть от конкретной реализации и версии ОС.

1.6. Резервное копирование данных

Перед выполнением любых операций по настройке или модификации системы хранения, если в системе хранения у вас уже размещены какие-то данные, настоятельно рекомендуется предварительно создать резервную копию всей информации, находящейся в системе. Это необходимо для предотвращения потери важных файлов и исключения риска их повреждения в случае ошибочных действий или сбоев в процессе изменения конфигурации. Такой подход позволяет обезопасить ваши ресурсы даже при возникновении непредвиденных ситуаций.

2. НАСТРОЙКА WEB-UI

Формирование структурных компонентов TROK осуществляется с использованием web-интерфейса, который предоставляет все необходимые функции для контроля и мониторинга ресурсов. Более подробную информацию вы можете найти в документе «TROK Руководство по работе с графическим интерфейсом».

Параметры для входа:

username: "admin"

password: "admin".

В следующих разделах будут представлены дополнительные инструкции по работе с командной строкой.

3. ИМЯ УЗЛА И ВОЗМОЖНЫЕ КОНФЛИКТЫ ПРИ СОЗДАНИИ

Перед тем как перейти к описанию правил именования, необходимо чётко разграничить используемые термины:

– имя хоста – сетевой идентификатор физического или виртуального сервера в операционной системе, обычно используемый для обращения к серверу в локальной сети или через DNS;

– имя узла TROK – уникальное имя, присваиваемое контролируемому TROK компоненту (узлу) в рамках программного обеспечения TROK, которое используется для управления ресурсами кластера.

При выборе уникального имени узла TROK рекомендуется учитывать следующие правила:

– имя узла в системе TROK и имя хоста должны совпадать;

– разрешенные базовые символы – латинские буквы: a-z, A-Z; цифры: 0-9; спецсимволы: - (дефис); . (точка);

– начало имени не может начинаться со спецсимвола (например, имя .server недопустимо);

– длина уникального имени должна составлять не более 48 символов;

– регистр букв не учитывается (например, Node1 и node1 считаются идентичными);

– запрещенные символы и форматы – пробелы, кириллица, символы: _, @, #, $, %, ^, &, *, (), {}, [], /, \, :, ;, ", ', ~, !;

– запрещено использовать IP-адреса в качестве имён узла (например, 192.168.1.1 недопустимо).

Если ваши имя узла в системе TROK и имя хоста отличаются, можно воспользоваться следующими сценариями:

Приоритетный сценарий:

– Изменить имя хоста, используйте команду:

```
sudo hostnamectl set-hostname <имя_хоста>
```

Важно. Данная операция может повлечь за собой необходимость корректировки конфигурационных файлов иных программных компонентов, установленных в операционной системе. Поэтому перед выполнением изменения следует тщательно оценить потенциальное влияние на функционирование стороннего программного обеспечения и убедиться в безопасности данной процедуры для текущих процессов.

Сценарий, который используется в редких случаях и только в определенных условиях:

– Изменить имя узла в системе TROK.

Важно. Если узел участвовал в репликации, необходимо выполнить её перенастройку. Рекомендуется применять данную процедуру только в ситуациях, когда узел более не должен входить в состав кластера, перед переустановкой системы или выводом сервера из эксплуатации, а также если узел был добавлен в самом начале настройки ошибочно.

– Система TROK не имеет команды, изменяющей имя узла напрямую, поскольку имя узла тесно связано с её уникальным идентификатором в базе данных контроллера. Чтобы изменить имя узла, необходимо удалить существующий узел в системе TPOK и создать новый с нужным именем на том же хосте:

– Добавьте новый узел с правильным именем с использованием web-интерфейса системы. Используйте тот же IP-адрес, что у старого узла.

– Удалите старый узел.

4. ПОДГОТОВКА К РАЗМЕТКЕ ДИСКОВ

Для начала выведите детальный список устройств хранения используя команду:

```
sudo fdisk -l
```

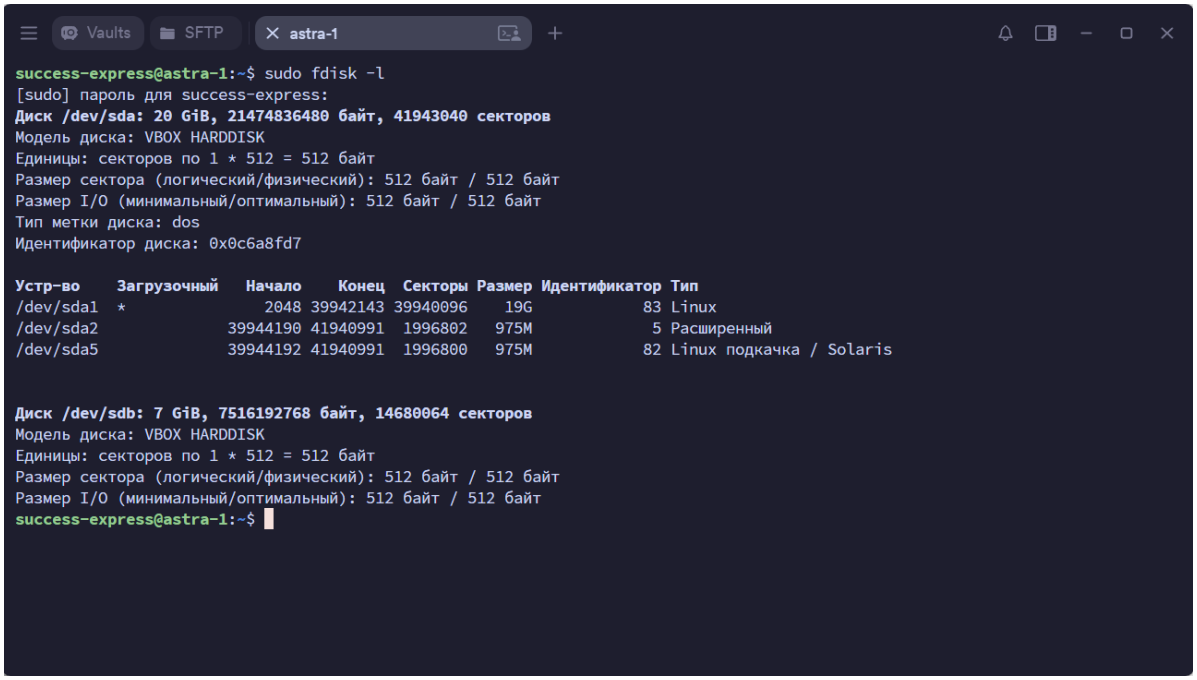


Рисунок 2 – Пример результата выполнения команды sudo fdisk -l

В текущем примере диск «sdb» свободен и готов к разметке. Именно он используется в дальнейшей работе.

Существуют альтернативные способы распределения дискового пространства. На изображении ниже мы видим **нежелательный вариант конфигурации системы**: один виртуальный диск на 20 ГБ (/dev/sda) и три его раздела:

– /dev/sda1 (19 ГБ, основной раздел, тип Linux) – системный раздел с ОС и всеми файлами;

– /dev/sda2 (975 МБ, расширенный);

– /dev/sda5 (975 МБ, подкачка – swap).

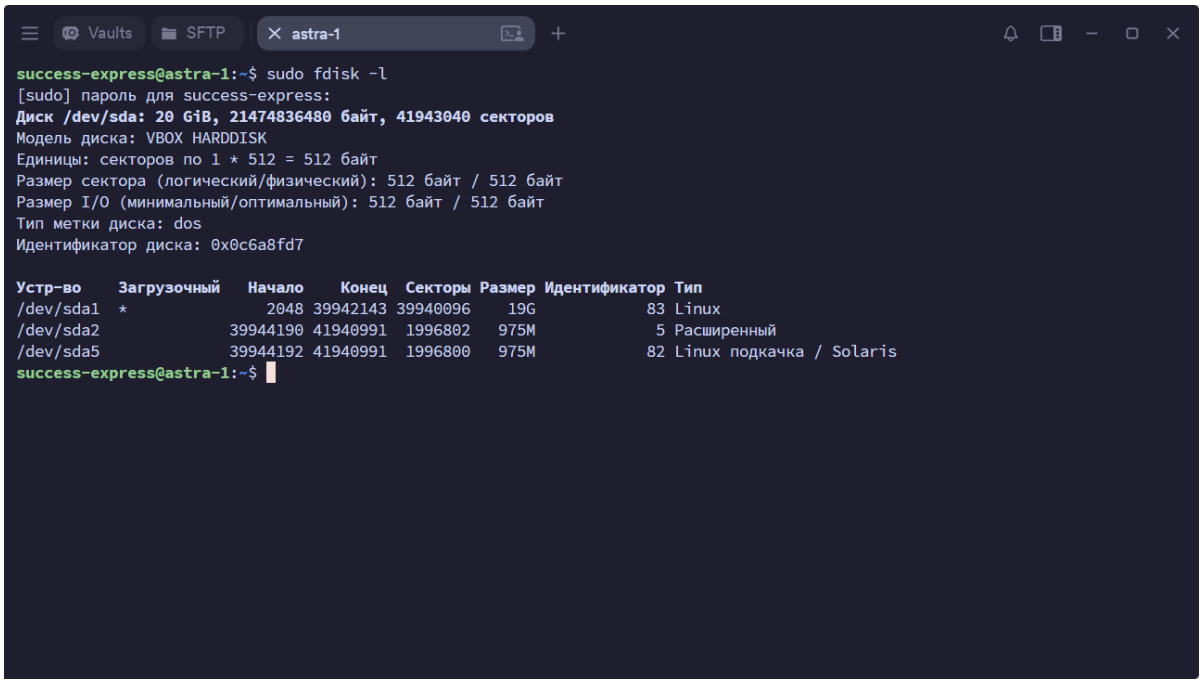


Рисунок 3 – Пример нежелательного набора устройств хранения

Единственный способ отделить необходимое количество дискового пространства в таком случае под TROK на этом диске – уменьшить текущий системный раздел /dev/sda1.

Это опасная процедура, требующая:

– Полного резервного копирования данных и ОС!

– Перезапуска системы через live-среду (например, GParted Live USB/CD), так как нельзя уменьшать смонтированное root-раздел (где работает ваша ОС).

Важно. Чтобы не допускать подобную ситуацию, при установке системы требуется обязательно производить предварительную разметку дисков.

5. РАЗМЕТКА ДИСКОВ LVM ДЛЯ ПУЛОВ ХРАНЕНИЯ

Разметка дисков осуществляется на узлах, на которых установлена служба trok-worker. Указанная операция необходима только в случае, если для хранения планируется использовать **не** весь диск, а его часть (например, отдельный раздел или логический том). Если же в состав кластера включается целый диск, и он будет целиком выделен под хранилище, этап предварительной разметки можно исключить.

LVM (Logical Volume Manager) – это система управления разделами диска в Linux, которая позволяет объединять несколько физических дисков или их разделов в одну большую группу хранения (volume group). Внутри этой группы можно создавать логические тома (logical volumes), которые, по сути, работают как обычные разделы, но могут быть динамически увеличены или уменьшены, перемещены между дисками, а также быстро снапшотированы (зафиксированы во времени). Использование LVM делает структуру хранения данных более гибкой, удобной для резервного копирования и масштабирования без необходимости переформатировать весь диск или останавливать систему.

С помощью классического LVM каждый логический том, выделяемый TROK для виртуальной машины или контейнера, полностью резервирует необходимое дисковое пространство заранее. Это гарантирует предсказуемое использование ресурсов и надёжность работы, так как том никогда не «закончится» неожиданно из-за нехватки места.

Процесс разметки дисков производится следующим образом:

Подготовьте место на диске для дальнейшей работы с помощью утилиты для работы с разделами fdisk:

```
sudo fdisk /dev/<имя_диска>
```

Важно. Все данные на диске будут удалены. Убедитесь, что выбрали правильный диск.

Внутри fdisk введите команды по порядку:

n → создать новый раздел.

p → выбрать тип раздела: основной (primary).

Enter → принять номер раздела по умолчанию.

Enter → принять первый сектор по умолчанию (2048).

Enter → принять последний сектор по умолчанию (весь диск).

w → сохранить изменения и выйти.

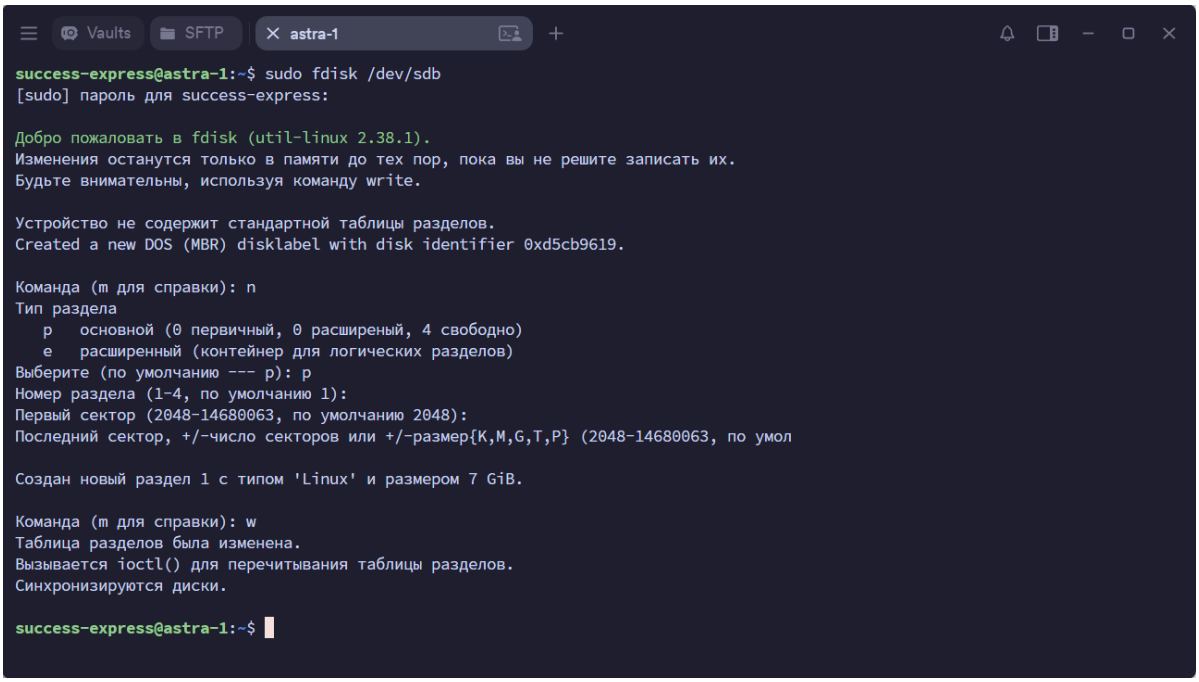


Рисунок 4 – Процесс подготовки места на диске для дальнейшей работы

Для инициализации раздела под LVM создайте физический том (PV) используя команду:

```
sudo pvcreate /dev/<имя_раздела>
```

pvcreate – команда для инициализации диска/раздела под LVM.

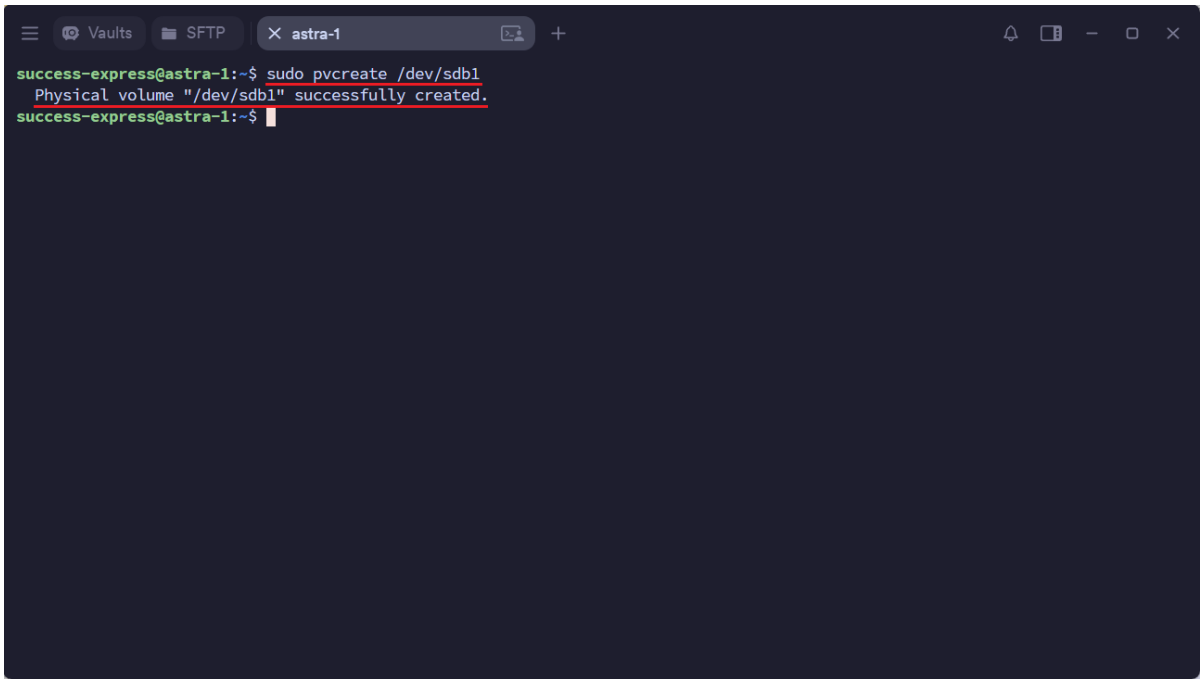


Рисунок 5 – Пример вывода при успешном выполнении команды sudo pvcreate /dev/sdb1

Для просмотра созданного физического тома используйте команду:

```
sudo pvdisplay
```

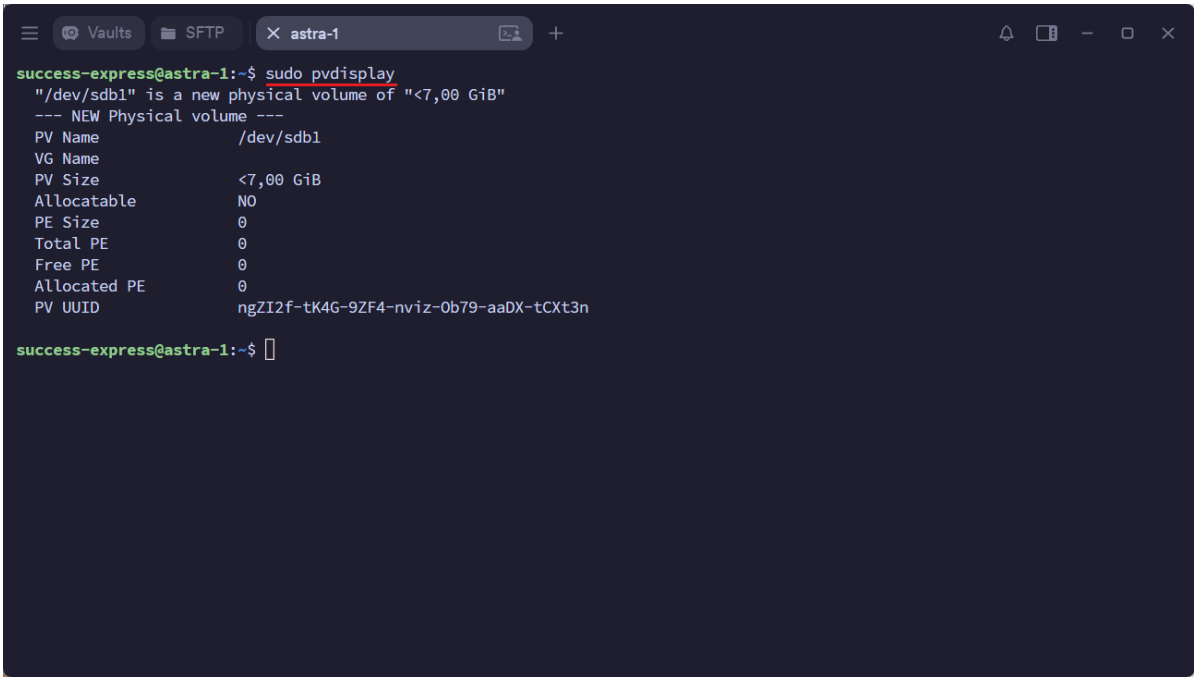


Рисунок 6 – Отображение созданного тома (PV)

Создайте группу томов (VG):

```
sudo vgcreate <имя_группы_ресурсов> /dev/<имя_раздела>
```

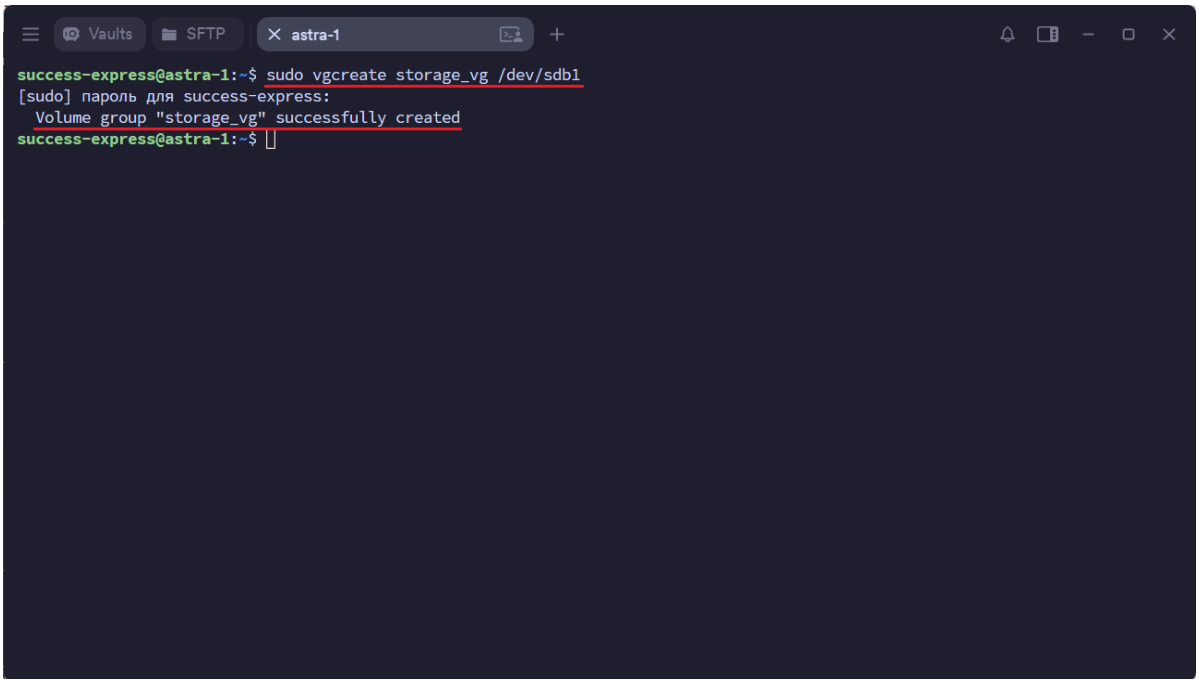


Рисунок 7 – Уведомление об успешном создании группы томов (VG)

Для проверки созданных объектов используйте команды:

```
   sudo pvdisplay | grep -B 1 <имя_раздела> # Фильтрует вывод, показывая строки с интересующим именем и дополнительную строку перед ними для контекстного понимания информации.

   sudo vgdisplay | grep -B 1 <имя_группы_ресурсов> # Аналогичный вывод с фильтрацией по имени тома
```

6. РАЗМЕТКА ДИСКОВ LVM_THIN ДЛЯ ПУЛОВ ХРАНЕНИЯ

Разметка дисков производится на нодах, на которых размещена служба trok-worker.

Как и в предыдущем случае, операция необходима только в случае, если для хранения планируется использовать **не** весь диск, а его часть.

LVM_THIN – это усовершенствованная технология внутри LVM, реализующая так называемое «тонкое выделение» (thin provisioning). В обычных томах LVM всё выделенное под логический том пространство резервируется сразу и полностью, даже если данные ещё не записаны. В LVM_THIN пространство для томов выделяется динамически, по мере необходимости: создаётся большой «thin pool» – специальный пул хранения, из которого каждый логический том получает столько реального места, сколько в действительности занимает его содержимое в данный момент. Это позволяет более эффективно использовать дисковое пространство и хранить, например, множество виртуальных машин или контейнеров, не резервируя под каждую физически максимально возможный объём, а выделяя ресурсы по факту использования.

LVM_THIN применяется в TROK для более гибкого и эффективного использования диска в крупных кластерах. При тонком выделении (LVM_THIN) TROK может создать множество томов, которые будут занимать ровно столько пространства, сколько им нужно в данный момент времени. Это особенно полезно для сценариев с динамическими виртуальными машинами и контейнерами – пользователей можно не ограничивать по размеру тома, а общий пул хранит только фактические данные. Таким образом, LVM_THIN экономит место и обеспечивает лёгкое масштабирование, а автоматизация TROK позволяет отслеживать использование.

Для начала создания LVM_THIN подготовьте место на диске для дальнейшей работы с помощью утилиты для работы с разделами fdisk:

```
sudo fdisk /dev/<имя_диска>
```

Важно. Все данные на диске будут удалены. Убедитесь, что выбрали правильный диск.

Внутри fdisk введите команды по порядку:

n → создать новый раздел.

p → выбрать тип раздела: основной (primary).

Enter → принять номер раздела по умолчанию.

Enter → принять первый сектор по умолчанию (2048).

Enter → принять последний сектор по умолчанию (весь диск).

w → сохранить изменения и выйти.

Для создания LVM Thin Pool необходимо выполнить следующую последовательность команд:

Создать физический том (PV):

```
sudo pvcreate /dev/<имя_раздела> # помечаем раздел как «физический том» для LVM
```

pvcreate – команда для инициализации диска/раздела под LVM.

/dev/<имя_раздела> – созданный раздел (после fdisk).

Создать группу томов (VG):

```
sudo vgcreate <имя_группы_ресурсов> /dev/<имя_раздела>
```

vgcreate – создает группу томов (логический контейнер для дисков).

Имя группы (можно выбрать любое, например, data_vg).

Создать «тонкий пул» в группе (Thin Pool):

```
sudo lvcreate -l 100%FREE -T <имя_группы_томов>/<имя_тома_lvm_thin>
```

lvcreate – создает логический том.

-l 100%FREE – использовать всё свободное место в группе.

-T – указание создать Thin Pool (пул с динамическим выделением места).

<имя_группы_ресурсов>/<имя_тома_lvm_thin> – том располагается внутри группы.

Для проверки созданных сущностей воспользуйтесь командой:

```
sudo lvdisplay | grep -E "<имя_логического_тома_в_LVM>|<имя_группы_томов_в_LVM>"
```

В случае возникновения необходимости в удалении созданных ресурсов, можно воспользоваться следующими командами:

– удаление группы томов:

```
sudo vgremove <имя_группы_ресурсов>
```

– удаление физического тома:

```
sudo pvremove /dev/<имя_раздела>
```

7. НАСТРОЙКА ДОСТУПА ПО ПРОТОКОЛАМ

Для работы с протоколами доступа должен быть установлен компонент trok-harbor.

Проверить его наличие можно с помощью команды:

```
sudo dpkg -s trok-harbor-meta
```

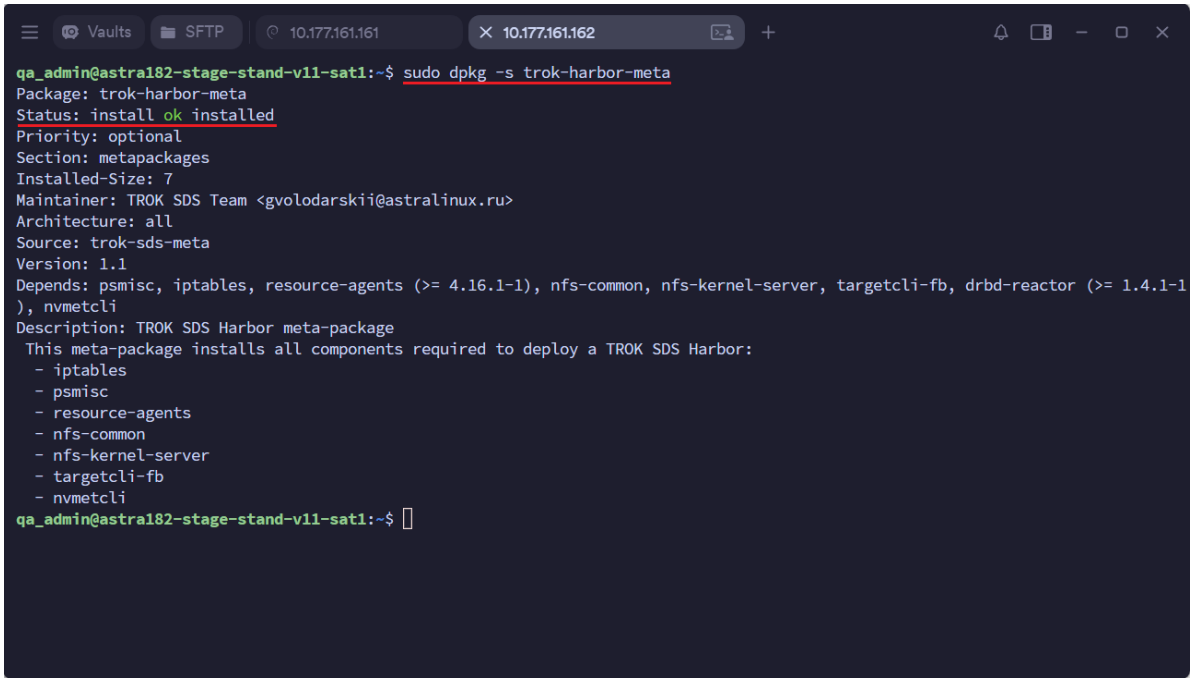


Рисунок 8 – Результат выполнения команды sudo dpkg -s trok-harbor-meta

Если пакет не установлен, используйте команду (подробная информация об установке пакетов указана в документе «TROK Инструкция по установке» в 8 пункте):

```
sudo apt install -y trok-harbor-meta
```

Настройка доступа к ресурсам осуществляется только после создания самих ресурсов. Подробная информация о процессе создания ресурса указана в документе «TROK Руководство по работе с графическим интерфейсом».

7.1. Настройка отказоустойчивого хранилища данных NFS на базе drbd-reactor

Служба NFS (Network File System) развёртывается в виде сервиса на каждом узле хранения, на котором расположена реплика ресурса, для обеспечения доступа к данным.

Для корректной работы NFS-кластера рекомендуется заранее создать отдельные DRBD-тома:

– На каждом узле хранения должен быть минимум один ресурс с томом для метаданных NFS, который предназначен для хранения служебной информации файловой системы. Обычно данный том имеет относительно небольшой объём (не менее 1% от общего объема системы хранения), так как метаданные занимают значительно меньше места по сравнению с пользовательскими данными.

– Ресурс (или ресурсы) с томами для данных NFS – используется для размещения непосредственно пользовательских данных, доступных через файловые шары (экспортируемые каталоги).

Все перечисленные DRBD-ресурсы должны быть предварительно созданы и полностью подготовлены к эксплуатации на всех узлах хранения, входящих в кластер.

Первый этап настройки включает в себя подготовку операционной системы.

Выполните команды для отключения и остановки стандартного NFS-сервиса:

```
sudo systemctl disable nfs-server
sudo systemctl stop nfs-server
```

Активируйте службу drbd-reactor:

```
sudo systemctl enable drbd-reactor
```

Вторым этапом следует подготовка файловой системы.

Переведите DRBD-ресурс в первичное (Primary) состояние. Если предполагается организация доступа по NFS к нескольким DRBD-ресурсам, необходимо перевести в состояние Primary все соответствующие ресурсы.

Для просмотра списка доступных DRBD-ресурсов и их текущего состояния используйте команду:

```
drbdadm status
```

Результат выполнения данной команды содержит информацию о каждом ресурсе, его роли и состоянии дисков на всех узлах. Например:

```
rd1 role:Secondary
  disk:UpToDate
  astra182-stage-stand-v11-sat2 role:Secondary
    peer-disk:UpToDate
  astra182-stage-stand-v11-sat3 role:Secondary
    peer-disk:UpToDate

rd2 role:Secondary
  disk:UpToDate
  astra182-stage-stand-v11-sat2 role:Secondary
    peer-disk:UpToDate
  astra182-stage-stand-v11-sat3 role:Secondary
    peer-disk:UpToDate
```

Для перевода выбранного DRBD-ресурса в состояние Primary используйте команду:

```
sudo drbdadm primary <имя_ресурса>
```

Примечание. Ресурс имеет то же имя, что и шаблон ресурса.

После изменения режима тома в web-интерфейсе изменят свой статус на «Используется».

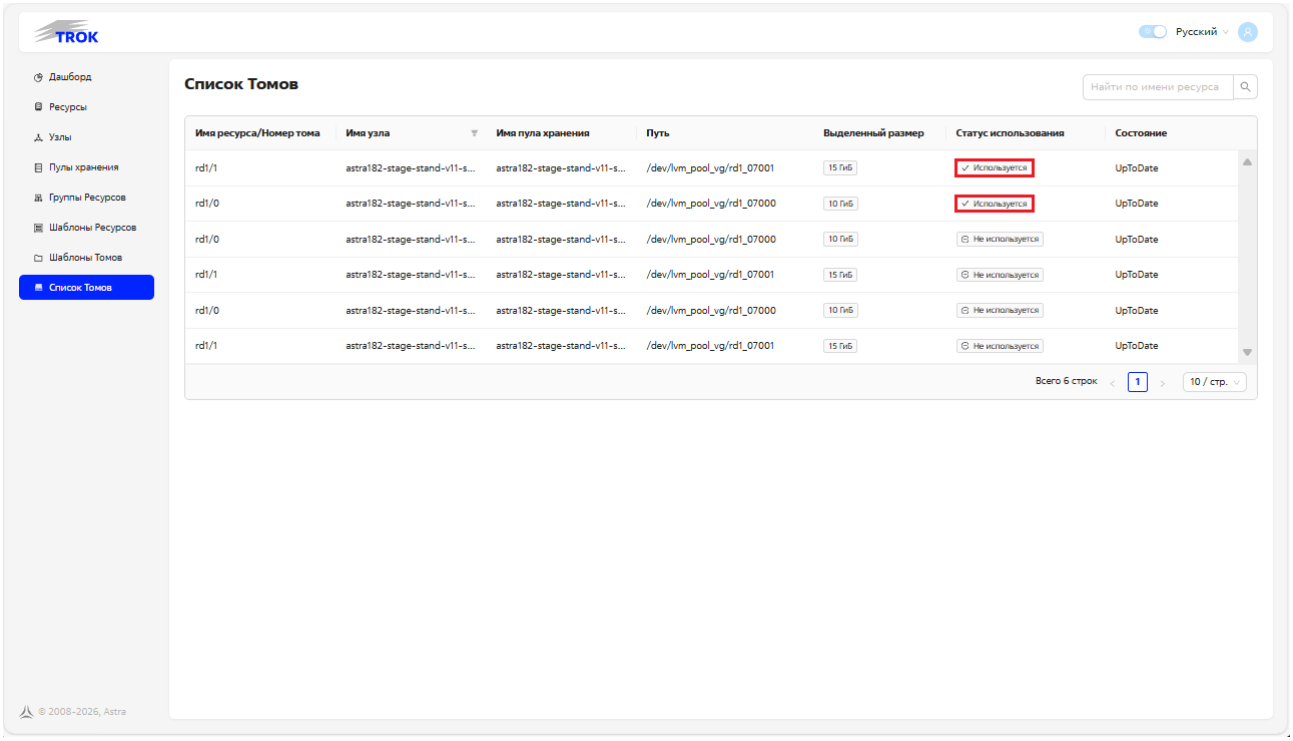


Рисунок 9 – Пример отображения статуса томов в web-интерфейсе

Создайте файловую систему на томах DRBD-ресурсов, к которым требуется доступ по NFS. Форматирование выполняется однократно на узле, где ресурс в состоянии Primary:

```
sudo mkfs -t ext4 /dev/drbd/by-res/<имя_ресурса>/<порядковый_номер_тома_ресурса>
```

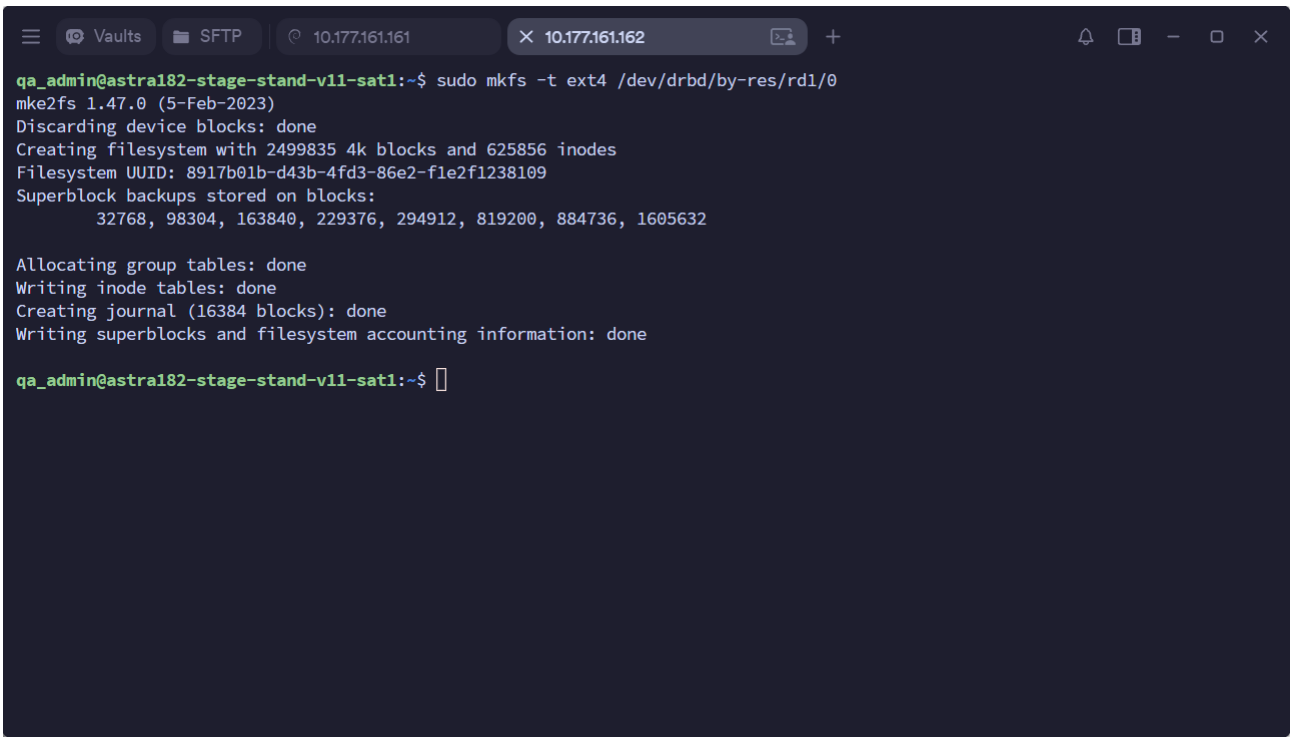


Рисунок 10 – Результат выполнения команды монтирования тома DRBD в файловую систему ext4

Создайте каталог для хранения метаданных NFS. Такой каталог должен быть создан на каждом узле хранения. Хранящиеся в данном каталоге метаданные используются для восстановления соединения с NFS-сервером в случае переключения роли primary-узла.

```
sudo mkdir -p /srv/ha/internal/nfs
```

Создайте структуру каталогов, которая будет использоваться для экспорта и служебных целей. Подготовьте каталоги для монтирования и служебные директории на всех узлах, где находятся drbd-ресурсы с томами для раздачи доступа по NFS. Сначала смонтируйте том для хранения метаданных NFS на первичном узле:

```
sudo mount /dev/drbd/by-res/<имя_ресурса_для_метаданных>/<номер_тома_для_метаданных> /srv/ha/internal/nfs
```

Создайте необходимые подкаталоги и установите права доступа на томе drbd-ресурса для хранения метаданных NFS используя команды:

```
sudo mkdir /srv/ha/internal/nfs/nfsinfo

sudo chmod -R 755 /srv/ha/internal/nfs
```

Создайте каталог на всех узлах:

```
sudo mkdir -p /srv/nfs-share/<имя_ресурса>
```

Смонтируйте тома для хранения пользовательских данных NFS на первичном узле:

```
sudo mount /dev/drbd/by-res/<имя_ресурса>/<номер_тома_для_данных> /srv/nfs-share/<имя_ресурса>

sudo chmod 755 /srv/nfs-share/<имя_ресурса>
```

Размонтируйте тома ресурсов для данных NFS:

```
sudo umount /srv/nfs-share/<имя_ресурса>
```

Размонтируйте тома и переведите ресурсы в состояние Secondary:

```
sudo umount /srv/ha/internal/nfs

sudo drbdadm secondary <имя_ресурса_1>

sudo drbdadm secondary <имя_ресурса_2> # Если у вас больше 1 ресурса на узле)
```

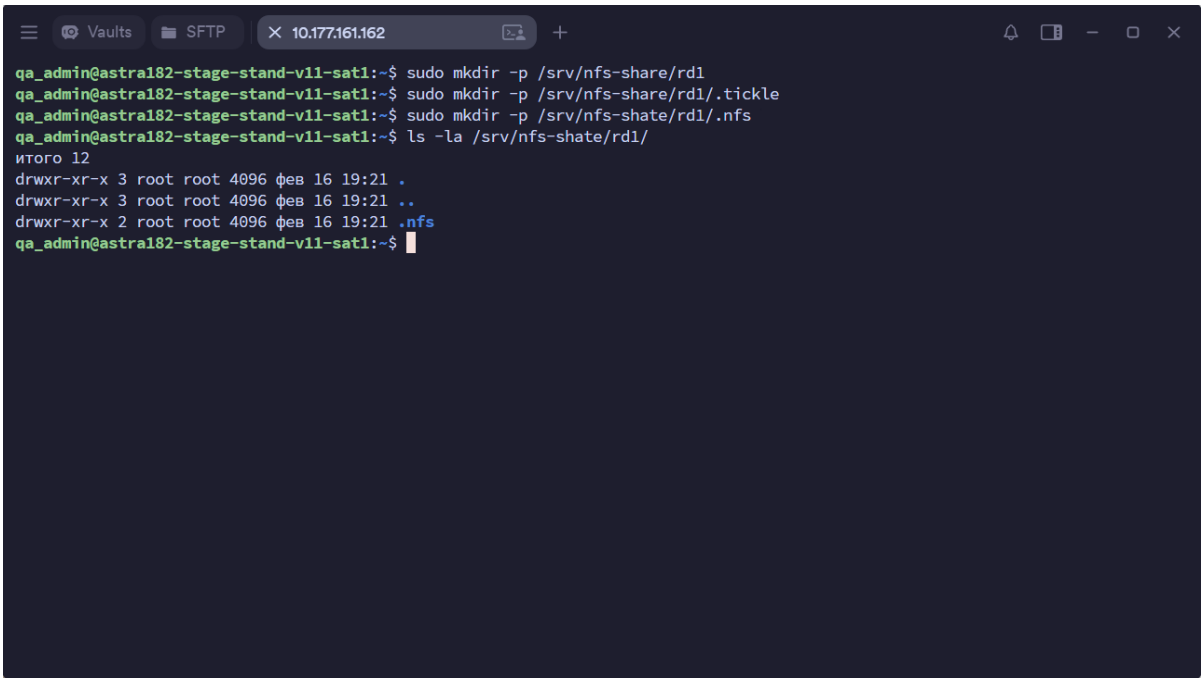


Рисунок 11 – Создание структуры каталогов

Далее следуют шаги для конфигурации службы drbd-reactor.

Создайте конфигурационный файл для promoter'a на всех узлах, где находятся drbd-ресурсы:

```
sudo touch /etc/drbd-reactor.d/nfs.toml
```

Откройте файл /etc/drbd-reactor.d/nfs.toml и добавьте следующую конфигурацию (универсальную для NFS-таргетов):

```
[[promoter]]

[promoter.resources]

 [promoter.resources.<имя_ресурса_для_метаданных>]
      on-drbd-demote-failure = "reboot-immediate"
      runner = "systemd"
      preferred-nodes = ["<имя_приоритетного_узла>"]

      start = [
            """ocf:heartbeat:Filesystem   fs_cluster_private   device=/dev/drbd/by-
```

```
res/<имя_ресурса_для_метаданных>/<номер_тома_для_метаданных>
directory=/srv/ha/internal/nfs fstype=ext4 run_fsck=no""",

               """ocf:heartbeat:Filesystem         fs_1         device=/dev/drbd/by-
res/<имя_ресурса>/<номер_тома_для_данных>  directory=/srv/nfs-share/rd2  fstype=ext4
run_fsck=no""",
               """ocf:heartbeat:nfsserver       nfsserver       nfs_ip=<плавающий_ip>
nfs_server_scope=<плавающий_ip>  nfs_shared_infodir=/srv/ha/internal/nfs/nfsinfo""",
               """ocf:heartbeat:exportfs    exp_nfs_a    directory=/srv/nfs-share/rd2
fsid=101                                clientspec=<подсеть_для_подключения_клиентов>
options='rw,no_subtree_check,no_root_squash'""",
               """ocf:heartbeat:IPaddr2    virtual_ip    cidr_netmask=<маска_подсети>
ip=<плавающий_ip> """,

   ]
```

Пример конфигурации:

```
   [[promoter]]

   [promoter.resources]

    [promoter.resources.nfs1]
         on-drbd-demote-failure = "reboot-immediate"
         runner = "systemd"
         preferred-nodes  =  ["astra182-stage-stand-v11-cmbnd","astra182-stage-stand-
v11-sat1"
   ]

         start = [
               """ocf:heartbeat:Filesystem  fs_cluster_private  device=/dev/drbd/by-
res/nfs1/0 directory=/srv/ha/internal/nfs fstype=ext4 run_fsck=no""",
               """ocf:heartbeat:Filesystem  fs_1  device=/dev/drbd/by-res/nfs-data1/0
directory=/srv/nfs-share/nfs-data1 fstype=ext4 run_fsck=no""",
               """ocf:heartbeat:nfsserver       nfsserver       nfs_ip=10.177.161.170
nfs_server_scope=10.177.161.170 nfs_shared_infodir=/srv/ha/internal/nfs/nfsinfo""",
               """ocf:heartbeat:exportfs      exp_nfs_data_1      directory=/srv/nfs-
share/nfs-data1                  fsid=101                  clientspec=10.177.161.0/24
options='rw,no_subtree_check,no_root_squash'""",
               """ocf:heartbeat:IPaddr2          virtual_ip          cidr_netmask=24
ip=10.177.161.170""",
   ]
```

Разбор конфигурации:

– [promoter]]

Открывает новый блок описания промоутера. Позволяет вам определить отдельный набор ресурсов/действий для работы с кластером.

– [promoter.resources]

Блок описания ресурсов, которыми управляет этот promoter.

– [promoter.resources.rd1]

Блок описания ресурсов с именем ресурса.

– on-drbd-demote-failure = "reboot-immediate"

Если при переключении ресурса DRBD с режима Primary в режим Secondary (demote) происходит ошибка, этот узел будет немедленно перезагружён.

– runner = "systemd"

Параметр задаёт способ управления всеми вспомогательными сервисами и ресурсами для DRBD-promoter. В данном случае использование systemd позволяет запускать, останавливать, отслеживать статус нужных служб через системный сервис-менеджер (systemd)

– preferred-nodes = ["<имя_узла_1>", "<имя_узла_2>"]

Определяет список узлов (серверов) – кандидатов для промоутирования ресурса в primary (активная роль). Кластер учитывает порядок и предпочтения, где в первую очередь держать ресурс в активном состоянии. Если что-то случилось с первым узлом в списке – промоут происходит на втором, и так далее. Например, если preferred-nodes = ["node1", "node2"] – DRBD-ресурс будет стараться быть primary на node1, и если недоступен – только тогда на node2.

– """ocf:heartbeat:Filesystem fs_metadata device=/dev/drbd/by-res/<имя_ресурса>/1 directory=/srv/ha/internal/nfs fstype=ext4 run_fsck=no""",

Монтирование приватной служебной директории, содержащей метаданные NFS, осуществляется с использованием выделенного тома DRBD-ресурса. При создании ресурса вы создаете в нем помимо основных томов, дополнительный том специально для NFS. Он служит исключительно для хранения служебной информации и не используется для размещения пользовательских данных, представляя собой метатом для функциональности NFS-кластера. В предварительно созданную директорию /srv/ha/internal/nfs монтируется том, например, <имя_ресурса>/1 (номер тома не имеет значение, вы выделяете его исходя из собственной архитектуры), в котором размещаются кластерные служебные данные, включая каталог nfs_shared_infodir.

– """ocf:heartbeat:Filesystem fs_1 device=/dev/drbd/by-res/<имя_ресурса>/<номер_тома> directory=/srv/nfs-share/<имя_ресурса> fstype=ext4 run_fsck=no""",

Правило подключает том с пользовательскими данными.

device – устройство DRBD-ресурс <имя_ресурса>/<номер_тома> реальные данные для первой шары. Его можно подключить, используя 2 вида ссылок: первая: /dev/drbd/by-res/<имя_ресурса>/<номер_тома> либо вторая: /dev/drbd<minor-номер_устройства>. Minor-номер устройства – уникальный идентификатор, используется DRBD для адресации своих блочных устройств. Например, drbd7000 – это уникальный номер 700 для защиты от конфликтов и том №0 (rd1/0), а drbd7001 – это аналогично уникальный номер 700 и том №1 (rd1/1). Посмотреть весь список блочных устройств можно использовав команду

```
ls /dev/ | grep drbd
```

Но в случае, если на сервере более одного ресурса, тома DRBD создаются со сквозным нумерованием. То есть, если у вас помимо ресурса rd1 также есть ресурс rd2 и тома в нем rd2/0 и rd2/1, то minor-номер устройств для них будет drbd7002 и drbd7003 соответственно. Чтобы не путаться в наименованиях, рекомендуется использовать более простой вариант ссылки – /dev/drbd/by-res/rd1/0.

Важно. В веб-интерфейсе отображаются тома LVM (путь тома LVM), однако их адреса не следует использовать при настройке правил. Для корректной работы необходимо указывать адреса томов DRBD, которые описаны выше.

Директория /srv/nfs-share/<имя_ресурса> – точка монтирования, где будут лежать файлы, которые NFS будет раздавать.

Если вы монтируете второй ресурс, то для него дополнительно нужно будет создать заранее каталог для монтирования и служебные директории. Пример каталога sudo mkdir -p /srv/nfs-share/<имя_второго_ресурса>, а также выдать соответствующие права, как описано выше.

Правило для него будет выглядеть аналогично:

"""ocf:heartbeat:Filesystem fs_2 device=/dev/drbd/by-res/<имя_второго_ресурса>/0 directory=/srv/nfs-share/<имя_второго_ресурса> fstype=ext4 run_fsck=no""",

– """ocf:heartbeat:nfsserver nfsserver nfs_ip=<плавающий_ip> nfs_server_scope=<плавающий_ip> nfs_shared_infodir=/srv/ha/internal/nfs/nfsinfo""",

Запускает фоновые процессы (демоны) NFS (nfsd, rpc.mountd, rpc.statd и так далее).

nfs_shared_infodir=/srv/ha/internal/nfs/nfsinfo – критически важная директория, которая должна находиться на общем хранилище (на выделенном томе DRBD). В ней NFS-сервер хранит служебную информацию, уникальные идентификаторы (например, для statd) и другие файлы состояния. Она необходима для корректного переключения сервиса между узлами, чтобы новый активный узел мог восстановить состояние NFS (например, уведомить клиентов о переезде сервиса).

NFS-сервер запущен, но порт 2049 всё ещё заблокирован (см. правило 1), поэтому клиенты пока не могут подключиться.

nfs_ip=<плавающий_ip> (ip можно выбрать произвольный, но он обязательно должен отличаться от ip адресов, использованных в других внутренних сетях организации для избежания конфликтов, например 192.168.10.199)

IP-адрес, на котором NFS-сервер будет принимать соединения. Это плавающий_virtual_ip, который также управляется кластером (отдельным ресурсом IPaddr2). NFS-сервер привязывается к этому адресу.

nfs_server_scope=<плавающий_ip>

Определяет, на каком интерфейсе или адресе NFS-сервер будет публиковать свои сервисы (через rpc.nfsd). Обычно совпадает с nfs_ip.

– """ocf:heartbeat:exportfs exp_nfs_1 directory=/srv/nfs-share/<имя_ресурса> fsid=1d0b09e1-1f61-5ae4-98c1-1661f84a5321 clientspec=<подсеть_клиентов>/24 options='rw,sync,no_subtree_check,no_root_squash"""",

Правило выполняет команду:

```
exportfs -o rw,sync,no_subtree_check,no_root_squash <подсеть_клиентов>/24:/srv/nfs-share/<имя_ресурса>
```

fsid: Уникальный идентификатор экспортируемой файловой системы. Критически важно, чтобы он совпадал на обоих узлах кластера и оставался уникальным в пределах одного NFS-сервера. Это обеспечивает корректное функционирование NFS-клиентов и предотвращает потерю доступа к экспортируемой файловой системе при переключении между узлами.

clientspec: Подсеть, из которой разрешен доступ клиентов. Например, 192.168.10.0/24

options: Специфический набор опции NFS.

Для второго ресурса правило будет выглядеть аналогично:

"""ocf:heartbeat:exportfs exp_nfs_2 directory=/srv/nfs-share/<имя_второго_ресурса> fsid=2d0b09e2-1f62-5ae4-98c2-2661f84a5322 clientspec=<подсеть_клиентов>/24 options='rw,sync,no_subtree_check,no_root_squash""",

– """ocf:heartbeat:IPaddr2 virtual_ip cidr_netmask=24 ip=<плавающий_ip> """,

Плавающий IP-адрес активируется только после завершения всех подготовительных этапов, за исключением процедуры разблокировки сетевого порта.

Перезапустите службу drbd-reactor для применения конфигурации:

```
systemctl restart drbd-reactor
```

Проверьте статус drbd-reactor с помощью команды:

```
drbd-reactorctl status
```

Убедитесь, что promoter активен и все сервисы запущены.

Для проверки NFS экспорта выполните команду:

```
exportfs
```

Убедитесь, что экспорт настроен для вашей сети.

Пример вывода команды sudo drbd-reactorctl status для primary узла:

```
etc/drbd-reactor.d/nfs.toml:
Promoter: Currently active on this node
● drbd-services@nfs1.target
● ├─ drbd-promote@nfs1.service
● ├─ ocf.rs@fs_cluster_private_nfs1.service
● ├─ ocf.rs@fs_1_nfs1.service
● ├─ ocf.rs@nfsserver_nfs1.service
● ├─ ocf.rs@exp_nfs_data_1_nfs1.service
● └─ ocf.rs@virtual_ip_nfs1.service
```

Пример вывода команды sudo drbd-reactorctl status для secondary узла:

```
/etc/drbd-reactor.d/nfs.toml:
Promoter: Currently active on node 'astra182-stage-stand-v11-cmbnd'
○ drbd-services@nfs1.target
○ ├─ drbd-promote@nfs1.service
○ ├─ ocf.rs@fs_cluster_private_nfs1.service
○ ├─ ocf.rs@fs_1_nfs1.service
○ ├─ ocf.rs@nfsserver_nfs1.service
○ ├─ ocf.rs@exp_nfs_data_1_nfs1.service
○ └─ ocf.rs@virtual_ip_nfs1.service
```

Протестируйте подключение смонтировав NFS-шару для проверки:

```
mount -t nfs <плавающий_ip>:/srv/ha/internal/<имя_ресурса> /mnt
```

7.2. Настройка отказоустойчивого хранилища данных Samba на базе drbd-reactor

Первый этап настройки включает в себя установку необходимых пакетов.

Для обновления кэша пакетов выполните команду на всех узлах кластера:

```
sudo apt update
```

Установите необходимые программы и компоненты: Samba, DRBD Reactor и OCF-агенты на всех узлах:

```
sudo apt install samba samba-client drbd-reactor resource-agents
```

Поскольку службой Samba будет управлять drbd-reactor, отключите запуск сервиса Samba при включении узла:

```
sudo systemctl disable smbd
```

Остановите сервис Samba:

```
sudo systemctl stop smbd
```

Следующим шагом выполняется настройка поддержки современных версий SMB.

Отредактируйте файл конфигурации Samba на всех узлах:

```
sudo vim /etc/samba/smb.conf
```

Вставьте следующие параметры в секцию [global] (добавьте секцию, если она отсутствует):

```
[global]
   client min protocol = SMB2_02
   server min protocol = SMB2_02
   client max protocol = SMB3_11
   server max protocol = SMB3_11
```

Разбор конфигурации:

– [global]

Начало глобального раздела в файле настроек. Всё, что указано ниже, касается всех пользователей и всех ресурсов сервера Samba.

– client min protocol = SMB2_02

client min protocol: определение минимального протокола (самого старого), который Samba будет использовать при подключении к другим серверам как «клиент».

SMB2_02: это первая версия современного безопасного протокола SMB2. С данного узла нельзя подключаться (к другому узлу) по старой, небезопасной версии протокола SMB1.

– server min protocol = SMB2_02

server min protocol: определение минимального протокола, который Samba принимает от клиентов (других узлов).

SMB2_02: также разрешено подключение с использованием протокола не ниже SMB2.

– client max protocol = SMB3_11

client max protocol: максимально новая версия протокола, которую Samba использует для подключения (к другим узлам) как «клиент».

SMB3_11 – одна из последних (и самых защищённых) версий протокола SMB3.

– server max protocol = SMB3_11

server max protocol – максимально допустимая версия протокола, если к этому узлу подключаются другие узлы.

SMB3_11 – тоже самая новая поддерживаемая версия.

Добавьте тестовый ресурс общего доступа Samba в конец файла /etc/samba/smb.conf на сервере:

```
[public]
   comment = Public Share
   path = /var/lib/samba/share
   browseable = yes
   read only = no
   guest ok = yes
   create mask = 0755
   directory mask = 0755
   force user = nobody
   force group = nogroup
```

Разбор конфигурации:

– [public]

Блок конфигурации общей (публичной) директории общего доступа.

– comment = Public Share

Это описание, которое может отображаться в интерфейсах управления или сетевых списках.

– path = /var/lib/samba/share

Путь на диске сервера к каталогу, который будет доступен через этот сетевой общий ресурс. Физическая папка на сервере.

– browseable = yes

Параметр, определяющий видимость сетевого ресурса в списке доступных через сеть ресурсов.

– read only = no

Запрет на запись отсутствует – можно читать и записывать файлы в этот общий сетевой ресурс.

– guest ok = yes

Гостевой доступ разрешён – можно подключаться без входа в систему. При необходимости можно отключить гостевой доступ и ограничить вход только для авторизованных пользователей.

– create mask = 0755

Права по умолчанию для создаваемых файлов внутри общего сетевого ресурса: владелец может читать, писать и выполнять, остальные – читать и выполнять.

– directory mask = 0755

Права по умолчанию для создаваемых каталогов внутри общего сетевого ресурса: аналогично, владелец имеет полный доступ, остальные – чтение и выполнение.

– force user = nobody

Все новые файлы будут принадлежать пользователю nobody. Этот параметр влияет на доступ к файлам, и кто может их редактировать.

– force group = nogroup

Все новые файлы будут принадлежать группе nogroup. Этот параметр влияет на права группы и доступ к файлам.

Создайте каталог для общего доступа, где будут храниться файлы, доступные с помощью протокола Samba.

```
sudo mkdir -p /var/lib/samba/share
```

Установите владельца и группу каталога. Для общего доступа чаще используют «nobody» как анонимного пользователя и «nogroup» как группы:

```
sudo chown nobody:nogroup /var/lib/samba/share
```

Если вы хотите, чтобы внутри каталога и все файлы внутри унаследовали этот же владельца/группу, можно применить рекурсивно:

```
sudo chown -R nobody:nogroup /var/lib/samba/share
```

Установите права доступа на каталог:

```
sudo chmod 0777 /var/lib/samba/share
```

0777 означает: владелец, группа и другие имеют чтение (4), запись (2) и выполнение (1) – всего 7 каждый. «Выполнение» здесь действует как возможность заходить в каталог.

Важно. 0777 даёт полный доступ всем. Если вам важно использовать более строгие настройки безопасности, можно задать более строгие параметры (например, 0775 или 0770) и настроить соответствующие параметры в smb.conf (guest ok = yes, writable = yes и т. п.).

Перезапустите службы Samba, чтобы применить внесённые изменения в конфигурацию:

```
sudo systemctl restart smbd
```

Примечание. На некоторых системах также может потребоваться перезапуск nmbd (сетевой сервис NetBIOS):

```
sudo systemctl restart nmbd
```

Проверьте настройки вашего файла конфигурации smb.conf с помощью команды:

```
sudo testparm
```

Если в smb.conf есть ошибки, testparm вернёт сообщение об ошибке, например:

```
ERROR: invalid parameter 'guest ok' in section [share]
```

Подготовьте DRBD ресурсы и миграцию данных Samba.

На одном из узлов переведите DRBD ресурс в первичный (active) режим:

```
sudo drbdadm primary <имя_ресурса>
```

Отформатируйте устройство DRBD в файловую систему XFS (создайте файловую систему).

Введите команду для форматирования:

```
sudo mkfs.xfs /dev/drbd/by-res/<имя_ресурса>/<номер_тома>
```

Внимание. После выполнения этой команды все старые данные на этом устройстве сотрутся.

Подготовьте директорию Samba. Создайте резервную копию данных и новую директорию. Сначала переместите старую папку с данными в «резервную»:

```
sudo mv /var/lib/samba /var/lib/samba.bak
```

Затем создайте новую пустую директорию:

```
sudo mkdir /var/lib/samba
```

Смонтируйте DRBD устройство и восстановите данные из резервной копии:

```
sudo mount /dev/drbd/by-res/<имя_ресурса>/<номер_тома> /var/lib/samba

sudo cp -av /var/lib/samba.bak/* /var/lib/samba/ # Команда копирует в том drbd-ресурса (вместе со скрытыми файлами и правами доступа).
```

Настройте drbd-reactor, выполнив следующую последовательность действий.

На всех узлах создайте файл конфигурации:

```
sudo vim /etc/drbd-reactor.d/<имя_ресурса>.toml
```

Отформатируйте файл, вставив следующий конфигурационный блок:

```
[[promoter]]
id = "<имя_ресурса>"
[promoter.resources.<имя_ресурса>]
on-drbd-demote-failure = "reboot-immediate"
runner = "systemd"
preferred-nodes = []
start = [
   "ocf:heartbeat:Filesystem     fs_1     device=/dev/drbd/by-res/<имя_ресурса>/0 directory=/var/lib/samba fstype=xfs run_fsck=no",
   "ocf:heartbeat:IPaddr2 service_ip cidr_netmask=24 ip=<плавающий_ip-адрес>",
   "smbd.service",
]
stop-services-on-exit = true
target-as = "BindsTo"
```

Разбор конфигурации:

– [promoter]]

Открывает новый блок описания промоутера. Позволяет определить отдельный набор правил для управления ресурсами в кластере. Каждый такой блок организует автоматизацию включения/выключения сервисов и управления ресурсом DRBD на выбранном сервере (узле).

– id = "<имя_ресурса>"

Указывает уникальное имя данного блока промоутера, обычно совпадает с именем DRBD-ресурса. Используется для идентификации и связи между конфигурационными секциями и сервисами.

– [promoter.resources.<имя_ресурса>]

Этот раздел описывает параметры для управления конкретным ресурсом с заданным именем. Все команды и действия ниже внутри этого блока будут относиться именно к названному ресурсу.

– on-drbd-demote-failure = "reboot-immediate"

Если возникла ошибка при попытке «понизить» (перевести в неактивное состояние) DRBD-ресурс, компьютер немедленно перезагрузится. Это помогает избежать проблем, когда одновременно два сервера считают себя «главными».

– runner = "systemd"

systemd – программа, которая управляет другими программами/службами в Linux. Запуск и остановка сервисов (Samba, ФС и т.д.) будет через systemd.

– preferred-nodes = []

Список «предпочтительных» серверов, на которых лучше всего держать этот ресурс активным. Пусто – значит, может быть активен на любом сервере.

– """ocf:heartbeat:Filesystem fs_1 device=/dev/drbd/by-res/<имя_ресурса>/<номер_тома> directory=/srv/lib/samba fstype=xfs run_fsck=no",,

Монтирование основной директории данных Samba происходит с использованием DRBD-устройства, которое представляет собой защищённый и реплицируемый диск между всеми узлами кластера.

Параметр device=/dev/drbd/by-res/<имя_ресурса>/<номер_тома> определяет путь к DRBD-устройству, связанному с данным ресурсом.

Монтирование выполняется в папку /var/lib/samba – стандартное расположение для сервисных и пользовательских данных Samba.

Указывается тип файловой системы fstype=xfs, что оптимально для больших файловых хранилищ с поддержкой современных возможностей.

Флаг run_fsck=no означает, что проверки целостности файловой системы при монтировании проводиться не будут, что ускоряет процедуру старта в целом.

– "ocf:heartbeat:IPaddr2 service_ip cidr_netmask=24 ip=<плавающий_ip-адрес>"

Плавающий (виртуальный) IP-адрес активируется на данном узле только после завершения всех обязательных подготовительных процедур: успешного монтирования DRBD-устройства и создания нужных сервисных папок.

Плавающий IP (ip=10.177.161.250) позволяет пользователям и клиентским компьютерам всегда подключаться к одному и тому же адресному объекту, даже если работающий в данный момент сервер меняется.

Параметр cidr_netmask=24 соответствует стандартной маске для подсети 255.255.255.0.

Этот адрес автоматически переносится между серверами кластера в зависимости от того, какой из них в текущий момент активен («primary»), обеспечивая бесперебойный доступ к файловому ресурсу Samba.

– "smbd.service"

Запуск сервиса Samba (smbd), который делает возможным обмен файлами по сети между компьютерами и этим сервером.

Сервис инициируется только после успешного завершения всех предыдущих этапов: монтирования хранилища и назначения плавающего IP-адреса, что гарантирует правильную доступность всех данных для пользователей.

Служба smbd обеспечивает работу протокола SMB/CIFS и взаимодействие с сетевыми папками.

Последним шагом будет запуск проверка конфигурации и тестирование подключения.

Перезапустите службу drbd-reactor. Выполните команду на всех узлах:

```
sudo drbd-reactorctl restart
```

Проверьте состояние конфигурации и запущенных сервисов (также выполняется на каждом узле):

```
sudo drbd-reactorctl status
```

Ожидаемый вывод:

```
/etc/drbd-reactor.d/<имя_ресурса>.toml:
Promoter: Currently active on this node
● drbd-services@<имя_ресурса>.target
● ├─ drbd-promote@<имя_ресурса>.service
● ├─ ocf.rs@fs_1_<имя_ресурса>.service
● ├─ ocf.rs@service_ip_<имя_ресурса>.service
● └─ smbd.service
```

Проверьте доступность Samba-ресурса, выполнив команду с любого клиента:

```
smbclient -L //<плавающий_ip-адрес> -m SMB3_11 -U%
```

Ожидаемый результат:

```
        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        public          Disk      Public Share
        IPC$             IPC       IPC  Service  (Samba  4.21.4-Debian-4.21.4+dfsg-
1ubuntu3.5astra1)
        nobody          Disk      Home Directories
SMB1 disabled -- no workgroup availableSharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        IPC$            IPC       IPC Service (Samba 4.21.4-Debian-4.21.4+dfsg-1as
tra4+b2)
        nobody          Disk      Home Directories
```

```
SMB1 disabled -- no workgroup available
```

Проверка возможности подключения к samba-ресурсу на удаленном хосте:

Установите пакет утилит cifs используя команду:

```
sudo apt install cifs-utils
```

Создайте точку монтирования:

```
sudo mkdir -p /mnt/samba_share
```

Выполните монтирование и проверьте результат:

```
sudo mount -t cifs //<плавающий_ip-адрес>/nobody /mnt/samba_share -o vers=3.11,guest,uid=$(id -u),gid=$(id -g),noperm
```

Набор параметров монтирования:

– vers=3.11 – версия протокола SMB, которая используется при монтировании.

– guest – монтирование выполняется как гость без ввода учётных данных. Если сервер позволяет гостевой доступ (guest ok = yes на сервере), не нужно вводить имя пользователя и пароль.

– uid=$(id -u) – (подстановка оболочки) устанавливает идентификатор владельца файлов в точке монтирования равным текущему пользователю Linux.

– Этот параметр необходим, чтобы файлы и каталоги, созданные в шаре, отображались как принадлежащие вам, а не «nobody».

– gid=$(id -g) – устанавливает идентификатор группы для файлов в точке монтирования равным основной группе текущего пользователя, чтобы групповые права соответствовали вашему пользователю и было проще работать совместно с другими пользователями.

– noperm – отключает проверку прав доступа на стороне клиента.

Для проверки монтирования используйте команду:

```
findmnt /mnt/samba_share
```

Она выведет информацию о точке монтирования, если такая существует. Если ничего не найдено, значит монтирование не получилось.

Проверка состояния DRBD ресурсов осуществляется с помощью команды для мониторинга репликации и показывает состояние синхронизации и целостности хранилища:

```
sudo drbdadm status
```

Состояние сервисов под управлением drbd-reactor можно проверить, выполнив команду:

```
sudo drbd-reactorctl status
```

7.3. Настройка отказоустойчивого хранилища данных iSCSI на базе drbd-reactor

Первый этап настройки включает в себя установку необходимых пакетов.

Для тестирования подключения установите iSCSI инициатор на клиентских узлах:

```
sudo apt install open-iscsi -y
```

Вторым этапом настройте конфигурации DRBD Reactor для iSCSI, а также конфигурацию iSCSI Initiator (клиентской части).

Создайте или отредактируйте файл конфигурации DRBD Reactor:

```
sudo vim /etc/drbd-reactor.d/iscsi-<имя_ресурса>.toml
```

Вставьте следующий конфигурационный блок iSCSI в файл:

```
[[promoter]]

[promoter.resources]

[promoter.resources.<имя_ресурса>]
on-drbd-demote-failure = "reboot-immediate"
runner = "systemd"
start = [
  "ocf:heartbeat:IPaddr2 service_ip0 cidr_netmask=24 ip=<плавающий_ip-адрес>",
  "ocf:heartbeat:iSCSITarget             target             allowed_initiators='' incoming_password=<пароль_iSCSI>    incoming_username=<логин_iSCSI>    iqn=iqn.2025-12.ru.astra:<имя_ресурса> portals=<плавающий_ip-адрес>:3260",
```

```
     "ocf:heartbeat:iSCSILogicalUnit       lu1       lun=1       path=/dev/drbd/by-
res/<имя_ресурса>/<номер_тома>     product_id='TROK     iSCSI'     scsi_sn=7509fab1
target_iqn=iqn.2025-12.ru.astra:<имя_ресурса>",
   ]
   stop-services-on-exit = true
   target-as = "Requires"
```

Разбор конфигурации:

– [promoter]]

Открывает новый блок описания промоутера. Позволяет определить отдельный набор правил для управления ресурсами в кластере. Каждый такой блок организует автоматизацию включения/выключения сервисов и управления ресурсом DRBD на выбранном сервере (узле).

– [promoter.resources]

Секция-«контейнер» для всех ресурсов промоутера. Здесь перечисляются конкретные блоки ресурсов. В данном примере этот раздел пуст, но он предусматривает существование секции [promoter.resources.<имя_ресурса>].

– [promoter.resources.<имя_ресурса>]

Описание параметров для управления конкретным ресурсом с заданным именем. Все команды и действия внутри этого блока относятся именно к названному ресурсу. Обратите внимание, что в этом примере явного поля id нет; имя ресурса задаётся в имени секции вместо обозначения <имя_ресурса>.

– on-drbd-demote-failure = “reboot-immediate”

Если возникает ошибка при попытке понизить DRBD-ресурс (перевести его в неактивное состояние), система немедленно перезагружается. Это предотвращает ситуацию, когда два узла считают себя «primary» одновременно.

– runner = “systemd”

Указывает систему-инициатор управления сервисами. systemd будет запускать/останавливать сервисы и блоки, указанные далее (например, IP-адрес, iSCSI-ресурсы и т.д.) через systemd Unit-файлы и цели.

– stop-services-on-exit = true

После завершения работы промоутера (или при выходе из блока) следует остановить запущенные сервисы. Это помогает корректно освободить ресурсы

и не держать сервисы в запущенном состоянии при уходе узла из активной роли.

– start = []

Перечисление команд в стартовом блоке означает следующие шаги:

– ocf:heartbeat:IPaddr2 service_ip0 cidr_netmask=24 ip=<плавающий_ip-адрес>

Плавающий (виртуальный) IP-адрес, который будет активироваться на текущем активном узле. service_ip0 — имя или метка IP-адреса, который будет управляться навыком Failover-кластера. cidr_netmask=24 задаёт сетевую маску 255.255.255.0. Такой IP создаёт единый доступ к ресурсу независимо от того, на каком узле он запущен.

– ocf:heartbeat:iSCSITarget target allowed_initiators=‘’ incoming_password=12345678 incoming_username=astra iqn=iqn.2025-12.ru.astra:ssd1 portals=<плавающий_ip-адрес>:3260

iSCSI Target – компонент, который делает доступным iSCSI-ресурс для инициаторов.

target – имя цели (target) внутри iSCSI Target-объекта.

allowed_initiators=‘’ – список разрешённых инициаторов; пустое значение означает любые инициаторы разрешены, или настройка по умолчанию.

incoming_username/incoming_password – учётные данные для аутентификации.

iqn=iqn.2025-12.ru.astra:<имя_ресурса> – iSCSI Qualified Name цели.

portals=<плавающий_ip-адрес>:3260 – список порталов (IP-адрес/порт) для подключения инициаторов к цели.

– ocf:heartbeat:iSCSILogicalUnit lu1 lun=1 path=/dev/drbd/by-res/<имя_ресурса>/<номер_тома> product_id=‘TROK iSCSI’ scsi_sn=7509fab1 target_iqn=iqn.2025-12.ru.astra:<имя_ресурса>

iSCSI Logical Unit (LU) – логический том, который будет использоваться как SCSI-устройство для инициатора.

lu1 – имя LU внутри целевой конфигурации.

lun=1 – номер LUN внутри этой LU.

path=/dev/drbd/by-res/<имя_ресурса>/<номер_тома> – путь к DRBD-устройству, которое реплицируется между узлами и будет экспортировано как LU.

product_id=‘TROK iSCSI’ – идентификатор продукта для LU (обычно используется для описания на стороне инициатора).

scsi_sn=7509fab1 – SCSI-serial number LU.

target_iqn=iqn.2025-12.ru.astra:<имя_ресурса> – целевой IQN, к которому привязан LU.

– stop-services-on-exit = true

Завершение работы промоутера сопровождается остановкой запущенных связанных сервисов и ресурсов, чтобы корректно освободить узел и снизить риск конфликтов или «зацикливания» ресурсов.

– target-as = “Requires”

Указывает тип зависимости цели (target) от других сервисов. Значение “Requires” означает, что целевой блок будет активен только если зависимости-сервисы запущены; иначе запуск происходить не будет. Это формирует корректную последовательность запуска: сначала необходимые сервисы, затем целевые ресурсы.

Пример использования конфигурационного блока:

```
   [[promoter]]
   [promoter.resources]

   [promoter.resources.disk1]
   on-drbd-demote-failure = "reboot-immediate"
   runner = "systemd"
   start = [
     "ocf:heartbeat:IPaddr2 service_ip0 cidr_netmask=24 ip=10.177.161.168",
     "ocf:heartbeat:iSCSITarget             target             allowed_initiators=''
incoming_password=12345678   incoming_username=astra   iqn=iqn.2026-02.ru.astra:disk1
portals=10.177.161.168:3260",
     "ocf:heartbeat:iSCSILogicalUnit    lu1    lun=1    path=/dev/drbd/by-res/disk1/0
product_id='TROK iSCSI' scsi_sn=7509fab1 target_iqn=iqn.2026-02.ru.astra:disk1",
```

```
]
stop-services-on-exit = true
target-as = "Requires"
```

После внесения изменений перезапустите drbd-reactor командой:

```
sudo systemctl restart drbd-reactor
```

Выполните проверку состояния ресурса:

```
drbd-reactorctl status
```

Пример вывода:

```
/etc/drbd-reactor.d/iscsi-disk1.toml:
Promoter: Currently active on this node
● drbd-services@disk1.target
● ├─ drbd-promote@disk1.service
● ├─ ocf.rs@service_ip0_disk1.service
● ├─ ocf.rs@target_disk1.service
● └─ ocf.rs@lu1_disk1.service
```

Разбор вывода:

– Promoter: Currently active on this node

Узел в настоящий момент выступает промотором (то есть активным узлом, который несёт роль ведущего для данного DRBD-ресурса).

Далее перечислены связанные элементы:

drbd-services@disk1.target – системная единица типа target, предназначенная для управления DRBD-сервисами ресурса disk1.

drbd-promote@disk1.service – сервис, который выполняет продвижение ресурса disk1 на текущем узле (приведение к состоянию primary).

ocf.rs@service_ip0_disk1.service – агент ресурса OCF (Open Cluster Framework) для IP-адреса, связанного с ресурсом disk1.

ocf.rs@target_disk1.service – управления ресурсом DRBD-таргета disk1.

ocf.rs@lu1_disk1.service – агент ресурса для связанного логического устройства (LU) disk1.

Далее следует блок настроек ISCSI на клиентских узлах.

Задайте уникальное имя для идентификации iSCSI инициатора:

```
echo "InitiatorName=iqn.2026-02.ru.astra:<имя_ресурса>" > /etc/iscsi/initiatorname.iscsi
```

Отредактируйте файл конфигурации iSCSI и укажите учетные данные для настройки аутентификации CHAP:

```
sudo vim /etc/iscsi/iscsid.conf
```

Добавьте или измените следующие строки:

```
node.session.auth.username = <логин_ iSCSI>

node.session.auth.password = <пароль_iSCSI>
```

Кандидаты логин/пароль должны соответствовать настройкам iSCSI Target, которые указаны в вашей конфигурации start блока файла iscsi-<имя_ресурса>.toml.

Выполните команду для обнаружения iSCSI Target:

```
sudo iscsiadm -m discovery -t sendtargets -p <плавающий_ip-адрес>
```

Пример успешного вывода:

```
10.177.161.168:3260,1 iqn.2026-02.ru.astra:disk1
```

Выполните команду для подключения к iSCSI Target:

```
sudo iscsiadm -m node --login
```

Пример успешного вывода:

```
Logging in to [iface: default, target: iqn.2026-02.ru.astra:disk1, portal: 10.177.161.168,3260]

Login to [iface: default, target: iqn.2026-02.ru.astra:disk1, portal: 10.177.161.168,3260] successful.
```

Определите метку диска TROK iSCSI. Введите команду:

```
sudo fdisk -l
```

Пример вывода:

```
Disk /dev/sda: 979,66 MiB, 1027252224 bytes, 2006352 sectors
Disk model: TROK iSCSI
```

```
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 1048576 bytes
```

Выполните команду для получения уникального SCSI ID:

```
sudo /usr/lib/udev/scsi_id -g -u /dev/sda # Для примера имя диска из вывода выше
```

Запишите полученный идентификатор (например: 360014057509fab100000000000000000).

Создайте файл правила для автоматического создания символической ссылки:

```
sudo cat > /etc/udev/rules.d/30-change-iscsi-name.rules << EOF
KERNEL=="sd[a-z]", SUBSYSTEM=="block", PROGRAM="/usr/lib/udev/scsi_id -g -u /dev/\$name", RESULT=="360014057509fab100000000000000000", SYMLINK+="iscsi/disk_1"
EOF
```

Перезагрузите правила udev для немедленного применения:

```
sudo udevadm control --reload-rules
```

Отключитесь от iSCSI Target (при необходимости). Для удаления подключения выполните следующие команды:

```
iscsiadm -m node -p <плавающий_ip-адрес>:3260,0 -T iqn.2025-12.ru.astra:<имя_ресурса> --logout

iscsiadm -m node -p <плавающий_ip-адрес>:3260,0 -T iqn.2025-12.ru.astra:<имя_ресурса> -o delete
```

Пример команды с указанными данными:

```
sudo iscsiadm -m node -p 10.177.161.168:3260,0 -T iqn.2026-02.ru.astra:disk1 --logout

sudo iscsiadm -m node -p 10.177.161.168:3260,0 -T iqn.2026-02.ru.astra:disk1 -o delete
```

Пример успешного вывода команд:

```
Logging out of session [sid: 1, target: iqn.2026-02.ru.astra:disk1, portal: 10.177.161.168,3260]

Logout of [sid: 1, target: iqn.2026-02.ru.astra:disk1, portal: 10.177.161.168,3260] successful.
```

На финальном этапе следует настроить проверку и мониторинг.

Проверьте статус drbd-reactor. Убедитесь, что конфигурация загружена и ресурс активен:

```
sudo drbd-reactorctl status
```

Пример успешного вывода:

```
/etc/drbd-reactor.d/iscsi-disk1.toml:
Promoter: Currently active on this node
● drbd-services@disk1.target
● ├─ drbd-promote@disk1.service
● ├─ ocf.rs@service_ip0_disk1.service
● ├─ ocf.rs@target_disk1.service
● └─ ocf.rs@lu1_disk1.service
```

Восстановите подключение к таргету:

```
sudo iscsiadm -m discovery -t sendtargets -p <плавающий_ip-адрес>

sudo iscsiadm -m node --login
```

Проверьте доступность iSCSI Target. С клиентского узла убедитесь, что диск доступен по символической ссылке:

```
sudo ls -la /dev/iscsi/<имя_ресурса>
```

Пример вывода команды:

```
lrwxrwxrwx 1 root root 6 мар  2 11:40 /dev/iscsi/disk_1 -> ../sda
```

8. МОНИТОРИНГ PROMETHEUS ДЛЯ КОНТРОЛЛЕРОВ

Программное обеспечение предоставляет возможности для интеграции контроллеров с системой мониторинга Prometheus. В случае, если контроллер запущен, он автоматически экспортирует метрики в формате Prometheus, и данные могут быть извлечены посредством запроса к конечной точке http:// controller’s-ip/metrics.

8.1. Метрика trok_info

Метрика trok_info предоставляет базовую информацию о версии и сборке контроллера и содержит метки:

– getid – уникальный идентификатор сборки (хеш коммита Git);

– buildtime – время сборки программного обеспечения;

– version – версия программного обеспечения (семантическое версионирование).

Пример значения:

```
trok_info{getid="372c916b7d97fa10e8ea480b66ea3da665ab5849",buildtime="2025-10-13T14:22:02+03:00",version="1.29.2"} 1
```

Интерпретация:

Значение всегда равно 1 – подтверждает доступность метрик. Используется для проверки работоспособности экспортера и позволяет отслеживать версии в гетерогенных средах.

8.2. Метрика trok_node_state

Метрика trok_node_state отслеживает состояние узлов в кластере и имеет следующие коды состояний:

– 0 = OFFLINE – узел недоступен;

– 1 = ONLINE – узел работает нормально;

– 2 = CONNECTING – выполняется подключение к узлу;

– 3 = EVICTED – узел исключен из кластера.

Метки:

- node – имя узла в кластере;
- nodetype – тип узла (satellite или controller);
- address – IP-адрес узла;
- port – порт для подключения;
- encryption – тип шифрования (PLAIN = без шифрования).

Пример значения:

```
trok_node_state{node="nodeA",nodetype="SATELLITE",address="10.177.161.43",port="50002",encryption="PLAIN"} 1
```

8.3. Метрика trok_storage_pool_capacity_free_bytes

Метрика trok_storage_pool_capacity_free_bytes отображает свободное место в пулах хранения и содержат следующие метки:

- storage_pool – имя пула хранения;
- node – UUID узла, на котором расположен пул;
- driver – тип драйвера хранилища:
  - LVM – классический LVM;
  - LVM_THIN – тонкий LVM;
  - ZFS – файловая система ZFS;
  - DISKLESS – бездисковый пул.

Пример значения:

```
trok_storage_pool_capacity_free_bytes{storage_pool="PoolA",node="0acfd509-93a6-4504-9df9-5f078095f42b",driver="LVM"} 1234000

trok_storage_pool_capacity_free_bytes{storage_pool="PoolB",node="0acfd509-93a6-4504-9df9-5f078095f42b",driver="LVM_THIN"} 1234000
```

8.4. Метрика trok_storage_pool_capacity_total_bytes

Метрика trok_storage_pool_capacity_total_bytes отображает общую емкость пулов хранения и использует набор меток аналогичный пункту 1.3.

Пример значения:

```
   trok_storage_pool_capacity_total_bytes{storage_pool="PoolA",node="0acfd509-
93a6-4504-9df9-5f078095f42b",driver="LVM"} 1236000

   trok_storage_pool_capacity_total_bytes{storage_pool="PoolB",node="0acfd509-
93a6-4504-9df9-5f078095f42b",driver="LVM_THIN"} 1236000
```

8.5. Метрика trok_storage_pool_error_count

Метрика trok_storage_pool_error_count является счетчиком ошибок в пулах хранения и использует метки аналогичные предыдущим метрикам пулов.

Пример значения:

```
   trok_storage_pool_error_count{storage_pool="PoolA",node="0acfd509-93a6-4504-
9df9-5f078095f42b",driver="LVM"} 0

   trok_storage_pool_error_count{storage_pool="PoolB",node="0acfd509-93a6-4504-
9df9-5f078095f42b",driver="LVM_THIN"} 0
```

Интерпретация:

- 0 – ошибок нет;
- >0 – имеются проблемы с пулом.

Используется для уведомлений о проблемах хранилища.

9. МОНИТОРИНГ PROMETHEUS ДЛЯ DRBD REACTOR

Настройка DRBD Reactor для экспорта метрик Prometheus так же проста. Следует разместить следующий файл конфигурации в директории конфигураций DRBD Reactor и перезапустить DRBD Reactor:

```
# cat << EOF > /etc/drbd-reactor.d/prometheus.toml
[[prometheus]]
enums = true
address = "0.0.0.0:9942"
EOF

# systemctl restart drbd-reactor.service
```

После перезапуска вы сможете отправить curl-запрос на конечную точку метрик и просмотреть метрики DRBD, предоставляемые DRBD Reactor.

```
# curl 127.0.0.1:9942
# TYPE drbd_device_alwrites_total counter
# HELP Number of updates of the activity log area of the meta data
drbd_device_alwrites_total{name="trok_db",volume="0",minor="1000"} 1
# TYPE drbd_resource_resources gauge
# HELP Number of resources
drbd_resource_resources 1
…snip…
```

10. ПРОЦЕДУРА ПРОВЕРКИ И УСТАНОВКИ ПАКЕТА TROK-WEBUI

Прежде чем приступить к настройке системы, важно убедиться в том, что пакет trok-webui корректно установлен на целевом сервере. Пакет является компонентом для развертывания веб-интерфейса и включает в себя необходимые зависимости. В случае отсутствия пакета его установка обязательна для обеспечения функциональности системы.

После установки рекомендуется вручную проверить и, при необходимости, скорректировать конфигурацию nginx для соответствия специфическим требованиям среды (например, настройка SSL, прокси-параметров или ограничений доступа).

Для проверки статуса установки пакета trok-webui воспользуйтесь командой управления пакетами:

```
dpkg -l | grep trok-webui
```

Команда фильтрует вывод списка всех установленных пакетов (dpkg -l) по ключевому слову trok-webui, отображая только релевантные записи. Если пакет установлен, вы увидите вывод, подобный следующему примеру:

```
trok-webui   1.0.2~rc6   amd64   TROK Web UI frontend
```

Здесь:

– trok-webui – имя пакета;

– 1.0.2~rc6 – версия пакета (может варьироваться в зависимости от доступной версии в репозиториях);

– amd64 – архитектура (для 64-битных систем);

– Web UI frontend – краткое описание пакета.

Если команда возвращает такой вывод пакет trok-webui успешно установлен, и конфигурация nginx должна быть автоматически применена.

Если вывод команды dpkg -l | grep trok-webui пустой (то есть пакет не найден), необходимо выполнить установку. Для этого используйте менеджер

пакетов apt, который автоматически разрешит зависимости и загрузит пакет из официальных репозиториев.

```
sudo apt install trok-webui
```

После выполнения этой команды система:

– Скачает пакет trok-webui и его зависимости (включая nginx, если они не установлены ранее).

– Установит их в соответствующие директории.

– Автоматически настроит базовую конфигурацию nginx для работы с trok-webui.

В процессе установки следите за сообщениями в терминале: они могут содержать важную информацию о потенциальных конфликтах или дополнительных шагах (например, перезапуск служб). После завершения установки рекомендуется перезапустить службу nginx для применения изменений:

```
sudo systemctl restart nginx
```

11. НАСТРОЙКА SSL-СЕРТИФИКАТОВ ДЛЯ NGINX

Для обеспечения безопасного соединения с веб-интерфейсом TROK рекомендуется использовать SSL/TLS-сертификаты. Это позволит шифровать трафик между клиентом (браузером) и сервером, предотвращая перехват данных. Без SSL соединение будет происходить по HTTP, что уязвимо для современных угроз.

Для файлов сертификатов и приватных ключей рекомендуется использовать стандартную директорию в структуре Nginx, такую как /etc/nginx/ssl.

Для создания директории выполните команду:

```
sudo mkdir -p /etc/nginx/ssl
```

Флаг -p (parents) обеспечивает создание родительских директорий, если они отсутствуют (хотя в большинстве случаев /etc/nginx уже существует).

Если в вашей системе используется альтернативная структура (например, /etc/ssl/private), адаптируйте путь соответственно.

Установка прав доступа выполняется командой:

```
sudo chmod 700 /etc/nginx/ssl
```

Команда устанавливает права drwx------: только владелец (обычно root) может читать, писать и выполнять. Группа и другие пользователи полностью исключены.

Приватные ключи – это чувствительные данные. Несанкционированный доступ может привести к компрометации сервера. После создания файлов внутри директории вы можете дополнительно настроить владение (chown) для пользователя nginx, если требуется (например, sudo chown -R nginx:nginx /etc/nginx/ssl).

После выполнения этих шагов директория готова. Проверьте её наличие и права командой:

```
ls -ld /etc/nginx/ssl
```

В результате выполнения команды должно появиться сообщение похожее на drwx------ 2 root root 4096 [дата] /etc/nginx/ssl. Теперь перейдем к генерации или установке сертификата.

При генерации или установке SSL-сертификата выбор метода зависит от вашей среды: для разработки подойдет самоподписанный сертификат, для корпоративной сети – внутренний CA, а для продакшена с внешним провайдером – fullchain от Let's Encrypt или аналогичного. Сертификат состоит из публичного файла (.crt или .pem) и приватного ключа (.key). После установки следует перезапустить Nginx (sudo systemctl restart nginx), чтобы изменения применились, а также протестировать соединение.

11.1. Создание самоподписанного (self-signed) сертификата

Самоподписанный сертификат генерируется с помощью OpenSSL и не требует внешнего CA, но браузеры и клиенты будут считать его недоверенным, показывая предупреждение «Небезопасное соединение».

Выполните единую команду для генерации пары ключ + сертификат (замените параметры на свои):

```
sudo openssl req -x509 -nodes -days 365 \
  -newkey rsa:4096 \
  -keyout /etc/nginx/ssl/trok-webui.key \
  -out /etc/nginx/ssl/trok-webui.crt \
  -subj "/C=RU/ST=Region/L=City/O=TROK/OU=WebUI/CN=192.168.1.10"
```

Разбор параметров:

– req -x509: генерирует запрос на сертификат и сразу самоподписывает его (X.509 – стандарт формата).

– -nodes: не шифрует приватный ключ паролем.

– -days 365: срок действия – 1 год (можно увеличить до 825 дней, но не более, чтобы соответствовать рекомендациям).

– -newkey rsa:4096: создает новый RSA-ключ длиной 4096 бит (высокий уровень безопасности; альтернатива – ECDSA для лучшей производительности).

– -keyout и -out: пути к файлам ключа и сертификата в директории.

– -subj: строка с данными субъекта (Subject). Здесь:

– C=RU: страна (Russia).

– ST=Region: регион (замените на свой, например, "Moscow").

– L=City: город (например, "Moscow").

– O= TROK: организация.

– OU=WebUI: подразделение.

– CN=192.168.1.10: Common Name – IP-адрес или домен сервера. Важно: CN должен точно совпадать с адресом, по которому клиенты подключаются (IP или hostname). Если используете доменное имя, замените на CN=trok-webui.local (или реальный домен, например, trok.example.com).

После выполнения проверьте файлы: ls /etc/nginx/ssl/ – должны появиться trok-webui.key и trok-webui.crt.

Установите права на ключ (только чтение для владельца):

```
sudo chmod 600 /etc/nginx/ssl/trok-webui.key
```

11.2. Установка сертификата, выданного внутренним CA

Собственный центр сертификации (внутренний CA) обеспечивает доверие внутри домена без обращения к публичным CA. Процесс включает копирование файлов сервера и настройку цепочки доверия (root CA + intermediate CA), чтобы клиенты автоматически доверяли сертификату.

Для начала создания у вас должны быть: server.crt (сертификат сервера), server.key (приватный ключ), rootCA.crt (корневой CA) и intermediateCA.crt (промежуточный CA, если есть).

Для копирования сертификата и ключа сервера выполните команды:

```
sudo cp server.crt /etc/nginx/ssl/trok-webui.crt

sudo cp server.key /etc/nginx/ssl/trok-webui.key
```

Установите права:

```
sudo chmod 600 /etc/nginx/ssl/trok-webui.*.
```

Данные операции изолируют файлы в директории Nginx и предотвращает случайное удаление оригиналов.

Перед установкой убедитесь, что сертификат сервера правильно подписан CA. Выполните:

```
openssl verify -CAfile rootCA.crt trok-webui.crt
```

Если цепочка полная (intermediate включен), используйте -CAfile intermediateCA.crt -untrusted rootCA.crt для многоуровневой проверки.

Ожидаемый вывод:

```
trok-webui.crt: OK.
```

Если возникает ошибка (например, «unable to get local issuer certificate»), проверьте файлы на валидность (openssl x509 -in trok-webui.crt -text -noout).

Установка CA-сертификатов в системное хранилище доверия:

Разные дистрибутивы имеют свои пути установки CA-сертификатов в системное хранилище доверия.

11.2.1. Установка CA-сертификатов для Debian/Ubuntu/Astra Linux

Выполните следующие команды:

```
sudo cp rootCA.crt /usr/local/share/ca-certificates/rootCA.crt
sudo cp intermediateCA.crt /usr/local/share/ca-certificates/intermediateCA.crt
sudo update-ca-certificates
```

update-ca-certificates обновит хранилище /etc/ssl/certs/ и перезапустит службы.

11.2.2. Установка CA-сертификатов для RHEL/AlmaLinux/Rocky Linux

Выполните:

```
sudo cp rootCA.crt /etc/pki/ca-trust/source/anchors/
```

```
sudo cp intermediateCA.crt /etc/pki/ca-trust/source/anchors/

sudo update-ca-trust
```

Это обновит глобальное хранилище NSS (Network Security Services), используемое Firefox и другими.

11.2.3. Установка CA-сертификатов для Windows-клиентов (если сервер обслуживает Windows-машины)

Выполните следующие шаги:

– Откройте инструмент certmgr.msc (Run → certmgr.msc);

– Импортируйте rootCA.crt в «Trusted Root Certification Authorities» (Доверенные корневые центры сертификации);

– Импортируйте intermediateCA.crt в "Intermediate Certification Authorities" (Промежуточные центры сертификации);

– Для групповой политики: Разверните через GPO или скрипты (certutil.exe).

11.3. Использование сертификата с полной цепочкой (fullchain)

Если ваш CA (внутренний или внешний, например, Let's Encrypt) выдает сертификат в формате fullchain (сертификат сервера + встроенная цепочка CA в одном файле), процесс упрощается – нет нужды вручную устанавливать промежуточные CA на сервере. Это удобно для автоматизированных систем.

У вас есть: trok-webui.fullchain.crt (fullchain-сертификат) и trok-webui.key (ключ).

Скопируйте их, используя команды:

```
sudo cp trok-webui.fullchain.crt /etc/nginx/ssl/trok-webui.fullchain.crt

sudo cp trok-webui.key /etc/nginx/ssl/trok-webui.key

sudo chmod 600 /etc/nginx/ssl/trok-webui.*
```

В файле конфигурации сайта (обычно /etc/nginx/sites-available/default или /etc/nginx/conf.d/trok.conf) в блоке server для HTTPS (порт 443) укажите:

```
ssl_certificate /etc/nginx/ssl/trok-webui.fullchain.crt;

ssl_certificate_key /etc/nginx/ssl/trok-webui.key;
```

Полный пример блока:

```
server {
    listen 443 ssl http2;
    server_name trok-webui.local;
    ssl_certificate /etc/nginx/ssl/trok-webui.fullchain.crt;
    ssl_certificate_key /etc/nginx/ssl/trok-webui.key;
    # Другие настройки: ssl_protocols TLSv1.2 TLSv1.3; etc.
}
```

Проверьте конфигурацию:

```
sudo nginx -t
```

Если ответ OK, перезапустите nginx:

```
sudo systemctl restart nginx
```

12. ВКЛЮЧЕНИЕ HTTPS В ФАЙЛЕ КОНФИГУРАЦИИ

Для получения доступа к файлу конфигурации введите команду:

```
sudo nano /etc/nginx/sites-available/trok-webui-nginx.conf
```

Путь /etc/nginx/sites-available/ – стандартная директория в Nginx, где хранятся конфигурации виртуальных хостов (сайтов). Файлы из этой директории обычно подключаются к активной конфигурации через симлинк в директорию /etc/nginx/sites-enabled/.

Внутри файла найдите закомментированный раздел, который скорее всего помечен как SSL CONFIG (OPTIONAL) или подобным образом. Раскомментируйте следующие критически важные директивы:

```
# Находим и убираем символ '#' в начале каждой нужной строки
listen 443 ssl;
http2 on;
ssl_certificate     /etc/nginx/ssl/trok-webui.crt;
ssl_certificate_key /etc/nginx/ssl/trok-webui.key;
ssl_session_cache   shared:SSL:10m;
ssl_session_timeout 1h;
ssl_protocols       TLSv1.2 TLSv1.3;
ssl_ciphers         HIGH:!aNULL:!MD5;
```

Значение каждой директивы:

– listen 443 ssl – указывает Nginx прослушивать входящие соединения на стандартном HTTPS-порту 443 и обрабатывать их с использованием SSL/TLS. Без этой директивы Nginx будет работать только по HTTP на порту 80, и шифрование применяться не будет.

– http2 on – включает поддержку современного протокола HTTP/2. HTTP/2 значительно увеличивает производительность веб-сайта за счёт мультиплексирования запросов, сжатия заголовков и других оптимизаций. Для его работы обязательно требуется HTTPS.

– ssl_certificate и ssl_certificate_key – указывают пути к файлу вашего SSL-сертификата (публичная часть) и соответствующему ему закрытому ключу. Nginx должен знать, где находятся эти файлы, чтобы установить безопасное

соединение. Убедитесь, что пути указаны абсолютно верно и файлы существуют.

– ssl_session_cache shared:SSL:10m; – настраивает кеш для параметров SSL-сессий размером 10 мегабайт, совместно используемый между всеми рабочими процессами Nginx (shared). При повторном соединении клиент может использовать данные из кеша сессии (resumption), что позволяет избежать полного и ресурсоемкого TLS-рукопожатия. Это значительно снижает нагрузку на процессор и ускоряет установление соединения.

– ssl_session_timeout 1h; – устанавливает время жизни кешированных параметров сессии – в данном случае на 1 час. Определяет, как долго клиент может возобновлять сессию без повторного рукопожатия. Баланс между безопасностью (меньшее время) и производительностью (большее время).

– ssl_protocols TLSv1.2 TLSv1.3; – явно разрешает для использования только современные и безопасные версии протоколов TLS версий 1.2 и 1.3. Старые протоколы (SSLv2, SSLv3, TLSv1.0, TLSv1.1) имеют уязвимости и считаются небезопасными. Эта директива запрещает их использование, повышая безопасность.

– ssl_ciphers HIGH:!aNULL:!MD5; – задает набор шифров, которые можно использовать при соединении. Позволяет использовать только самые сильные и безопасные алгоритмы шифрования, исключая уязвимые варианты.

– HIGH – разрешает только шифры с высокой стойкостью.

– !aNULL – запрещает шифры с анонимной аутентификацией (уязвимые для атак).

– !MD5 – запрещает шифры, использующие устаревший и небезопасный алгоритм хеширования MD5.

После внесения изменений и сохранения файла обязательно выполните следующее:

Проверьте синтаксис конфигурации командой:

```
sudo nginx -t
```

Успешный результат должен выглядеть следующим образом:

```
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Если проверка прошла успешно, примените конфигурацию – перезагрузите или перезапустите службу Nginx, чтобы изменения вступили в силу.

```
sudo systemctl reload nginx

# Или:

sudo systemctl restart nginx
```

reload предпочтительнее, так как он плавно загружает новую конфигурацию без полной остановки работы сервера.

Теперь веб-интерфейс TROK должен быть доступен по безопасному HTTPS-соединению по адресу https://your-server-ip-or-domain. Вы можете проверить это в браузере или с помощью утилит командной строки, таких как curl или openssl s_client.

13. АКТИВАЦИЯ БЛОКА РЕДИРЕКТА (ОПЦИОНАЛЬНО)

Чтобы обеспечить безопасность и соответствие веб-стандартам, рекомендуется настроить автоматическое перенаправление всех HTTP-запросов на защищенное HTTPS-соединение.

В вашем конфигурационном файле Nginx (/etc/nginx/sites-available/trok-webui-nginx.conf) найдите закомментированный блок сервера, предназначенный для редиректа. Обычно он выглядит следующим образом:

```
# server {
#   listen 80;
#   return 301 https://$host$request_uri;
# }
```

Раскомментируйте этот блок, удалив символы # в начале каждой строки.

14. РАСПОЛОЖЕНИЕ И СТРУКТУРА ЛОГ-ФАЙЛОВ

Логи системы разделены по функциональным компонентам и хранятся в директории /var/log/trok/. Представленная структура облегчает поиск, анализ и ротацию записей, относящихся к конкретному модулю.

Структура каталогов по умолчанию:

```
/var/log/trok/
├── trok-auth/                      # Логи модуля аутентификации и авторизации
│   ├── trok-auth.log               # Текущий активный лог-файл
│   └── trok-auth.log-2025-12-24    # Архивный лог-файл за указанную дату
├── trok-controller/                # Логи основного управляющего контроллера
│   ├── trok-controller.log
│   └── trok-controller.log-2025-12-24
├── trok-cp-endpoint/               # Логи Control Plane API
│   ├── trok-cp-endpoint.log
│   └── trok-cp-endpoint.log-2025-12-24
├── trok-installer/                 # Логи установки и обновления системы
│   └── install_20251124_131523.log  # Уникальный файл на каждую сессию установки
└── trok-worker/                    # Логи службы worker
    ├── trok-worker.log
    ├── trok-worker.log-2025-12-24.gz # Сжатый архивный лог
    └── trok-worker.log-2025-12-25
```

Настройка уровня логирования производится в конфигурационном файле соответствующего сервиса.

Важно. Для применения внесённых изменений требуется выполнить перезапуск данного сервиса.

Конфигурационные файлы располагаются в каталоге /etc/<имя-сервиса>/. Например, для открытия конфигурационного файла trok-controller можно воспользоваться следующей командой:

```
sudo vim /etc/trok-controller/config.yaml
```

В программном обеспечении используется несколько уровней подробности записываемых сообщений:

– Первый уровень – ERROR – фиксируются ошибки критического характера, которые приводят к нарушению работы системы или отдельных её компонентов. Сообщения этого уровня указывают на события, требующие немедленного вмешательства для восстановления штатной работы.

– Второй уровень – WARN – регистрируются предупреждения о потенциальных проблемах или аномалиях, которые пока не мешают функционированию системы, но могут впоследствии привести к ошибкам. Уровень позволяет своевременно выявлять и устранять возможные риски.

– Третий уровень – INFO – отображаются информационные сообщения о штатных событиях и процессах в системе. Этот уровень используется для отслеживания нормальной работы сервисов, фиксации ключевых этапов выполнения операций и общего контроля.

– Четвертый уровень – DEBUG – предоставляется наиболее подробная диагностическая информация, предназначенная главным образом для разработчиков и администраторов при отладке и глубоком анализе работы приложения. Сообщения этого уровня обычно включают подробные данные о выполнении программного кода.

Каждый последующий уровень логирования включает в себя сообщения всех предыдущих, менее подробных уровней. По умолчанию уровень логирования, заданный в конфигурационных файлах, устанавливается в значение INFO.

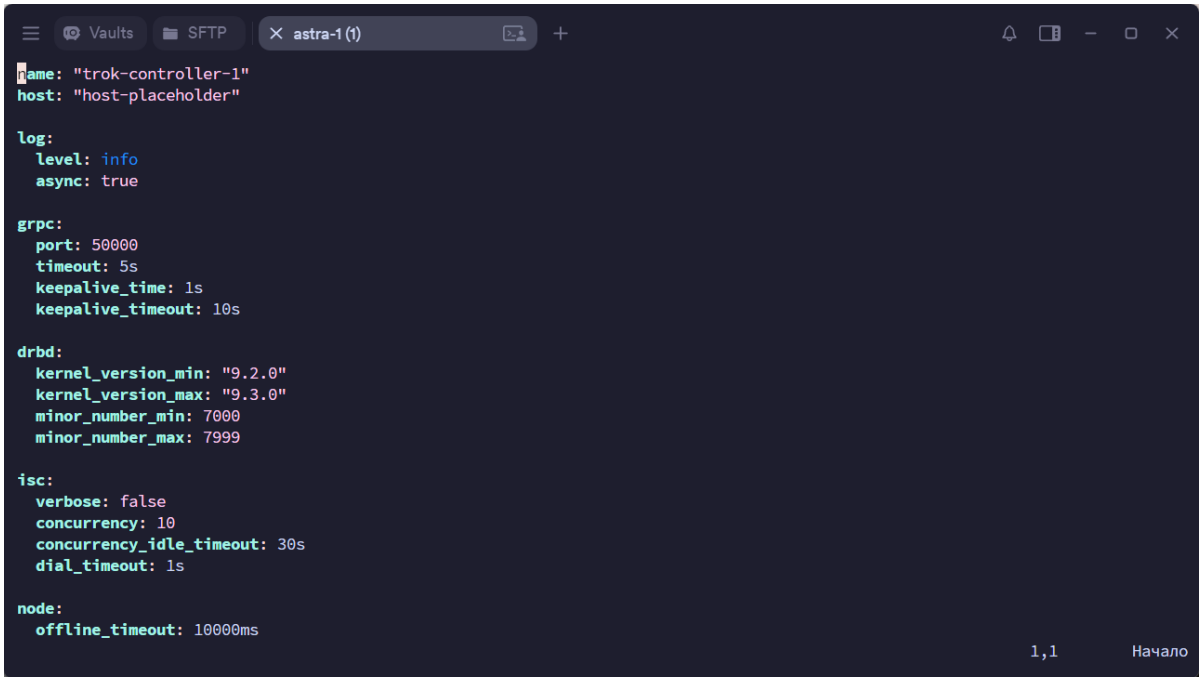


Рисунок 12 – Настройка уровня логирования в конфигурационном файле

К числу основных событий, фиксируемых в журналах (логах), относятся следующие:

– Уведомление о запуске сервиса контроллера с идентификацией версии ПО:

```
INFO  Starting trok-controller service  {"version": "20260217.1403", "hash": "7b127e8"}
```

Это описание обозначает запуск службы контроллера, указывая её версию и хэш-коммит исходного кода.

При обращении в службу поддержки укажите полученные в данные. Это позволит специалистам оперативно идентифицировать используемую версию системы и ускорит процесс диагностики и устранения проблемы.

– Уведомление о подключении к основной базе данных сервиса:

```
opening storage...  {"storage path": "/var/lib/trok-controller/data.db"}
```

В случае утраты базы данных вместо данного сообщения будет отображаться уведомление об ошибке подключения.

– Уведомление о запуске менеджера узлов, отвечающего за управление кластерными нодами:

```
starting nodes manager...  {"nodes manager": "nodes manager"}
```

Начинается обработка состояния и взаимодействия между узлами.

– Уведомление об успешной инициализации:

```
initialization finished, loaded 5 nodes id 958 us  {"nodes manager": "nodes manager"}
```

Сообщение на примере говорит об успешном завершении инициализации, в ходе которой было загружено 5 узлов за 958 микросекунд. Кластерные узлы успешно обнаружены и готовы к использованию.

– Уведомление об ошибке инициализации менеджера узлов:

```
error while initializing node manager
```

Система не смогла корректно загрузить информацию об узлах, необходима диагностика.

– Уведомление о запуске gRPC сервера:

```
Starting grpc server...
```

gRPC сервер отвечает за обработку удалённых вызовов и взаимодействие с клиентами/узлами.

– Уведомление о готовности к работе gRPC сервера:

```
grpc server started
```

– Уведомление об успешном корректном добавлении узла и регистрации его в кластере:

```
handshake response for hostname: astra17-dev1-sat2, drbd kernel version: 9.2.10 {"uuid": "36d0b11b-cd9b-4cc0-81df-5c45596799e1", "name": "node2"}
```

На примере кластерный контроллер подтверждает успешное установление соединения с узлом astra17-dev1-sat2, сообщая его DRBD-версию и уникальный идентификатор.

– Уведомление об установке канала связи для передачи событий между сервером и клиентом:

```
opening event stream for client dns:///192.168.161.45:50002  {"Events": ""}
```

Открытие потоковой передачи событий (event stream) для подключённого клиента по указанному адресу.

– Событие о состоянии ресурса:

```
got ResourceStateEvent, resource uuid: 2a116e74-a097-49af-be7c-bc251aecf45a, role: Secondary  {"Events": ""}
```

Указывается уникальный идентификатор ресурса и его роль (здесь – Secondary, то есть пассивная копия).

– Событие изменения состояния тома:

```
got DiskStateEvent, resource uuid: 2a116e74-a097-49af-be7c-bc251aecf45a, node uuid: 8e14bed2-5ce0-4386-965f-3cf0e383ddf8, volume: 0, state: UpToDate  {"Events": ""}
```

В примере указывается, что на определённом узле для конкретного ресурса и соответствующего тома изменено состояние на «UpToDate» (реплика данных синхронизирована).

– Событие изменения состояния соединения между двумя узлами кластера для конкретного ресурса:

```
   got ConnectionStateEvent, resource uuid: 2a116e74-a097-49af-be7c-bc251aecf45a, node from uuid: 8e14bed2-5ce0-4386-965f-3cf0e383ddf8, node to uuid: astra17-dev1-sat2, state: Connected  {"Events": ""}
```

В примере сообщение об установке соединения между двумя узлами (указаны уникальные идентификаторы и статус соединения).

– Уведомление о получении и регистрации вызова API-метода ListNode (список узлов) с заданными параметрами:

```
   grpc method ListNode called   {"reqID": "82377301-8553-41b5-8e34-cde6c2db6565", "offset": 0, "limit": -1}
```

Внешний клиент или процесс запрашивает информацию о списке узлов.

– Уведомление о регистрации вызова API-метода ViewRscs, предназначенного для просмотра ресурсов кластера с определёнными фильтрами:

```
   grpc method ViewRscs called   {"reqID": "8f62a44a-9ac8-4e0e-b36d-dd0f79744911", "filter by nodes": [], "filter by resources": [], "filter by storage pools": [], "filter by props": [], "offset": 0, "limit": 10}
```

15. НАСТРОЙКА ВЫСОКОДОСТУПНОГО КЛАСТЕРА

В разделе представлен пример развертывания высокодоступного кластера на узлах с именами astra182-stage-stand-v11-cmbnd, astra182-stage-stand-v11-sat1 и astra182-stage-stand-v11-sat2

На первом узле установите необходимые мета-пакеты TROK:

```
sudo apt install trok-combined-meta trok-harbor-meta
```

Для обеспечения отказоустойчивой работы сервиса контроллера необходимо развернуть его в кластерной конфигурации типа combined (комбинированные узлы, совмещающие в себе функции controller и worker), состоящей из минимум трех узлов, что позволяет поддерживать работоспособность системы даже при выходе из строя одного из серверов.

Переведите DRBD-ресурс с именем r1 (или другое имя, которое вы использовали) в активное (primary) состояние на этом узле:

```
sudo drbdadm primary r1
```

В результате выполнения команды данные на узле становятся доступными для записи и использования.

Отформатируйте DRBD-устройство под файловую систему ext4 (создайте файловую систему ext4 на DRBD-устройстве по пути /dev/drbd/by-res/r1/0), чтобы подготовить устройство к хранению файлов в привычном формате:

```
sudo mkfs -t ext4 /dev/drbd/by-res/r1/0
```

Вывод команды выглядит следующим образом:

```
mke2fs 1.47.0 (5-Feb-2023)
Discarding device blocks: done
Creating filesystem with 250794 4k blocks and 62720 inodes
Filesystem UUID: 4806735a-df87-4aba-9c91-79ce637e7d51
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
```

```
Writing superblocks and filesystem accounting information: done
```

– mke2fs 1.47.0 (5-Feb-2023)

Это версия утилиты mke2fs (часть пакета e2fsprogs), которая создаёт файловую систему на устройстве. Указана версия и дата выпуска, откуда следует, что именно эта программа выполняла форматирование.

– Discarding device blocks: done

Процесс очистки завершён успешно.

– Creating filesystem with 250794 4k blocks and 62720 inodes

250794 4k blocks – на устройстве/разделе будет создано 250,794 блока размером 4 килобайта (KB) каждый. 62720 inodes – будет создано 62,720 i-узлов (инодов), которые представляют файлы и каталоги в Linux. Иноды нужны для того, чтобы файловая система могла хранить информацию о файлах (права доступа, размер, дата и т. д.).

Из предоставленной информации можно примерно понять размер создаваемой файловой системы: 250 794 блоков × 4 KB ≈ 979 МБ. Это приблизительная ёмкость; фактическое доступное место может быть чуть меньше из-за метаданных и журналирования.

– Filesystem UUID: 4806735a-df87-4aba-9c91-79ce637e7d51

UUID – уникальный идентификатор файловой системы. Он нужен для точной идентификации диска в системе, особенно при монтировании через конфигурационные файлы или скрипты.

– Superblock backups stored on blocks:

Это перечисление блоков, на которых сохранены резервные копии суперблока. Суперблок содержит основную информацию о файловой системе.

В выводе на примере указаны блоки: 32768, 98304, 163840, 229376. Если основной суперблок повреждён, система может восстановиться, используя одну из этих резервных копий.

– Allocating group tables: done

Создаются таблицы групп блоков (group descriptors). Эти таблицы помогают файловой системе организовать данные на диске и ускоряют доступ к файлам.

– Writing inode tables: done

Записываются таблицы инодов.

– Creating journal (4096 blocks): done

Создается журнал (journal) размером 4096 блоков. Журнал нужен для устойчивости к сбоям: если система внезапно выключится, журнал помогает восстановить консистентность файловой системы.

– Writing superblocks and filesystem accounting information: done

Запись основного суперблока и всей сопутствующей информации файловой системы. Это финальный этап подготовки файловой системы к использованию.

Временно остановите службу trok-controller, чтобы можно было безопасно работать с данными/устройствами и работающие фоновые процессы не вызывали конфликтов:

```
sudo systemctl stop trok-controller
```

Отключите автоматический запуск trok-controller при старте системы до тех пор, пока вы не настроите все нужные шаги:

```
sudo systemctl disable trok-controller
```

Скопируйте данные контроллера в заранее подготовленное место, чтобы сохранить текущую базу данных отдельно на случай дальнейших изменений:

```
sudo cp -av /var/lib/trok-controller/data.db /opt
```

Команда копирует файл data.db из текущей папки контроллера в папку /opt, сохранив все атрибуты и символьные ссылки.

Примонтируйте DRBD-устройство в директорию контроллера, делая содержимое доступным как часть файловой системы, чтобы приложение (trok-controller) могло работать с данными уже на новом носителе:

```
sudo mount /dev/drbd/by-res/r1/0 /var/lib/trok-controller
```

Установите права доступа на директорию и все вложенные файлы/папки: владельцу – чтение/запись/исполнение, остальным – чтение и исполнение, чтобы пользователь и сервисы могли читать файлы и работать с папкой:

```
sudo chmod -R 755 /var/lib/trok-controller
```

Измените владельца директории на пользователя trok для того, чтобы сервисы имели правильные права на файлы и могли управлять ими:

```
sudo chown trok:trok -R /var/lib/trok-controller
```

Скопировать обратно данные data.db в новую директорию сохраняя атрибуты и структуру:

```
cp -av /opt/data.db /var/lib/trok-controller
```

В результате выполнения команды приложение снова получит доступ к своей базе в новой конфигурации.

Отмонтируйте каталог с помощью команды:

```
sudo umount /var/lib/trok-controller
```

Команда отключает активный доступ к данным по указанному пути, чтобы можно было безопасно изменить статус ресурса и подготовить смену роли DRBD.

Переведите DRBD-ресурс в вторичное состояние:

```
sudo drbdadm secondary r1
```

После выполнения команды узел начинает синхронизироваться как запасной.

Создайте конфигурацию drbd-reactor для ресурса r1. В ней описывается, какие действия нужно выполнить, когда этот узел станет активным (promoter):

```
sudo cat << EOF > /etc/drbd-reactor.d/trok-controller.toml
[[promoter]]
id = "r1"
[promoter.resources.r1]
start = [
"ocf:heartbeat:Filesystem ha_1 device=/dev/drbd/by-res/r1/0
directory=/var/lib/trok-controller fstype=ext4 run_fsck=no",
"ocf:heartbeat:IPaddr2 virtual_ip cidr_netmask=24 ip=10.177.161.169",
```

```
"trok-controller.service"
]
EOF
```

Конфигурация позволяет монтировать файловую систему, назначать плавающий IP и запускать trok-controller.

Перезагрузите процесс drbd-reactor, чтобы он применил новую конфигурацию:

```
sudo systemctl restart drbd-reactor
```

Проверьте состояние drbd-reactor

```
sudo drbd-reactorctl status
```

В результате выполнения команды отображается текущее состояние drbd-reactor и список активных ресурсов. Убедитесь, что всё запущено корректно и ваш promotor r1 активен на этом узле.

Пример вывода:

```
/etc/drbd-reactor.d/trok-controller.toml:
Promoter: Currently active on this node
● drbd-services@r1.target
● ├─ drbd-promote@r1.service
● ├─ ocf.rs@ha_1_r1.service
● ├─ ocf.rs@virtual_ip_r1.service
● └─ trok-controller.service
```

Пример вывода показывает, что файл trok-controller.toml загружен и активен на данном узле, а также запущенные сервисы.

Порядок действий для второго и третьего узла выглядит аналогичным образом (пример astra182-stage-stand-v11-sat1, astra182-stage-stand-v11-sat2).

Установите необходимые мета-пакеты для работы кластера:

```
sudo apt install trok-combined-meta trok-harbor-meta
```

Остановите trok-controller:

```
sudo systemctl stop trok-controller
```

Отключите автозапуск trok-controller:

```
sudo systemctl disable trok-controller
```

Удалите старую базу данных контроллера (data.db), чтобы можно было заново синхронизировать хранение с помощью команды:

```
sudo rm -f /var/lib/trok-controller/data.db
```

Создайте конфигурацию drbd-reactor для других узлов (r1):

```
sudo cat << EOF > /etc/drbd-reactor.d/trok-controller.toml
[[promoter]]
id = "r1"
[promoter.resources.r1]
start = [
"ocf:heartbeat:Filesystem            ha_1            device=/dev/drbd/by-res/r1/0
directory=/var/lib/trok-controller fstype=ext4 run_fsck=no",
"ocf:heartbeat:IPaddr2 virtual_ip cidr_netmask=24 ip=10.177.161.169",
"trok-controller.service"
]
EOF
```

Конфигурация promotor для ресурса r1 на этом узле необходима для его активации в случае необходимости и запуска сервисов.

Перезапустите drbd-reactor:

```
sudo systemctl restart drbd-reactor
```

Проверьте статус drbd-reactor (на узле, который настраивали):

```
sudo drbd-reactorctl status
```

Пример вывода для узла «astra182-stage-stand-v11-cmbnd»:

```
/etc/drbd-reactor.d/trok-controller.toml:
Promoter: Currently active on node 'astra182-stage-stand-v11-cmbnd'
○ drbd-services@r1.target
○ ├─ drbd-promote@r1.service
○ ├─ ocf.rs@ha_1_r1.service
○ ├─ ocf.rs@virtual_ip_r1.service
○ └─ trok-controller.service
```

Завершающий этап включает последовательное выполнение следующих действий:

Удалите конфигурацию на первом узле (на astra182-stage-stand-v11-cmbnd):

```
sudo mv /etc/drbd-reactor.d/trok-controller.toml /opt
```

Остановите соответствующие сервисы:

```
sudo systemctl stop trok-controller ocf.rs@virtual_ip_r1.service
ocf.rs@ha_1_r1.service drbd-promote@r1.service
```

Проверьте, что DRBD-ресурс r1 переведен в состояние primary на другом узле:

```
sudo drbdadm status
```

Пример вывода:

```
sudo drbdadm status
r1 role:Secondary
  disk:UpToDate
  astra182-stage-stand-v11-sat1 role:Primary
    peer-disk:UpToDate
  astra182-stage-stand-v11-sat2 role:Secondary
    peer-disk:UpToDate
```

Верните конфигурацию реактора в исходное состояние:

```
sudo cp -av /opt/trok-controller.toml /etc/drbd-reactor.d
```

Проверьте статус drbd-reactor на узле «astra182-stage-stand-v11-sat2»:

```
sudo drbd-reactorctl status
```

Результат выполнения команды:

```
sudo drbd-reactorctl status
/etc/drbd-reactor.d/trok-controller.toml:
Promoter: Currently active on node 'astra182-stage-stand-v11-sat1'
○ drbd-services@r1.target
× ├─ drbd-promote@r1.service
○ ├─ ocf.rs@ha_1_r1.service
○ ├─ ocf.rs@virtual_ip_r1.service
○ └─ trok-controller.service
```

Проверьте статус drbd-reactor на узле «astra182-stage-stand-v11-sat1»:

```
sudo drbd-reactorctl status
```

Результат выполнения команды:

```
/etc/drbd-reactor.d/trok-controller.toml:
Promoter: Currently active on this node
● drbd-services@r1.target
● ├─ drbd-promote@r1.service
● ├─ ocf.rs@ha_1_r1.service
● ├─ ocf.rs@virtual_ip_r1.service
● └─ trok-controller.service
```

Проверьте доступ по адресу http://10.177.161.169. Это можно сделать обычной командой проверки доступности сервиса, например:

```
curl -I http://10.177.161.169
```

Если команда выполняется успешно, сервис доступен через плавающий IP.