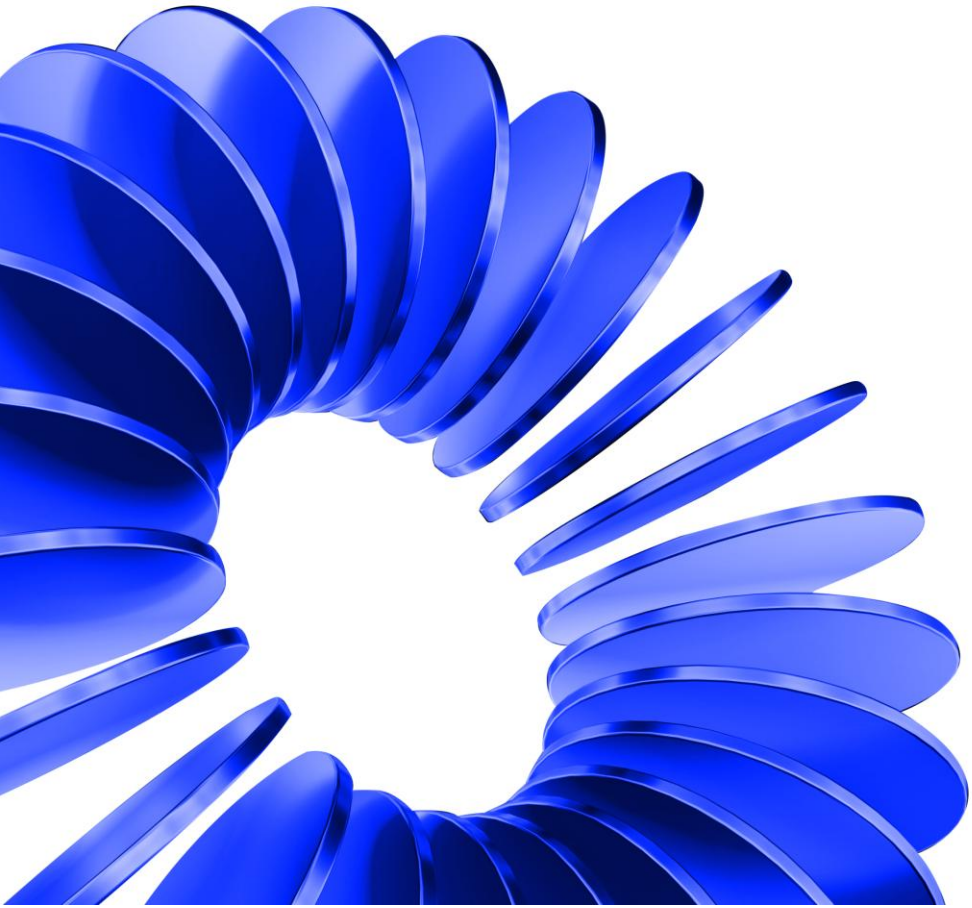
ОБЩЕСТВО С ОГРАНИЧЕННОЙ
ОТВЕТСТВЕННОСТЬЮ «РУСБИТЕХ-АСТРА»

СИСТЕМА ХРАНЕНИЯ ДАННЫХ TROK

# РУКОВОДСТВО ПО УСТАНОВКЕ



Москва, 2026г.

# СОДЕРЖАНИЕ

**1. ОПИСАНИЕ ПРОЕКТА ..... 4**
**2. ОПИСАНИЕ КОМПОНЕНТОВ ..... 5**
**3. ТРЕБОВАНИЯ ..... 7**
**4. ПОДГОТОВКА ОПЕРАЦИОННОЙ СИСТЕМЫ К УСТАНОВКЕ ..... 12**
4.1. Подготовка операционной системы специального назначения Astra Linux Special Edition к установке ПО TROK ..... 15
4.1.1. Настройка репозиториев для систем с доступом к сети Интернет (настройка для использования ветки репозиториев «frozen») ..... 16
4.1.2. Перенастройка репозиториев для автономных систем (локальные репозитории iso образа ОС) ..... 18
4.2. Монтирование образа TROK ..... 20
**5. КОНФИГУРАЦИЯ ..... 24**
**6. УСТАНОВКА С ИСПОЛЬЗОВАНИЕМ КОМАНДНОЙ СТРОКИ ..... 26**
**7. УСТАНОВКА В ИНТЕРАКТИВНОМ РЕЖИМЕ (GUI) ..... 28**
**8. УСТАНОВКА С ИСПОЛЬЗОВАНИЕМ ПАКЕТОВ ..... 31**
**9. ДЕПЛОЙ КЛАСТЕРА БЕЗ НА ..... 41**
9.1. Требования к управляющему хосту (узел управления Ansible) ..... 41
9.2. Установка коллекции TROK ..... 43
9.3. Подготовка структуры проекта ..... 44
9.4. Конфигурация файла инвентаря inventory.ini ..... 44
9.5. Настройка переменных (group_vars/all.yml) ..... 46
9.5.1. Базовая часть ..... 46
9.5.2. Опционально: глобальная настройка DRBD на worker-узлах ..... 47
9.5.3. Пул хранения ..... 50
9.5.4. iSCSI-экспорт ..... 51
9.5.5. NFS-экспорт ..... 53
9.5.6. Versity S3-шлюз ..... 55
9.6. Плейбук ..... 57
9.7. Запуск деплоя ..... 59
9.8. Проверка после деплоя ..... 61
9.8.1. Базовые сервисы ..... 61
9.8.2. Пул хранения ..... 62
9.8.3. iSCSI ..... 63

9.8.4. NFS ..... 65
9.8.5. Versity S3-шлюз ..... 67
**10. СЦЕНАРИЙ ДЛЯ ДЕПЛОЯ НА КЛАСТЕРА ..... 69**
10.1. Общая информация о ha_controllers ..... 69
10.2. Конфигурация файла инвентаря inventory.ini ..... 70
10.3. Настройка переменных (group_vars/all.yml) ..... 71
10.3.1. База для HA ..... 71
10.3.2. Пул хранения для HA ..... 71
10.3.3. HA-переменные ..... 72
10.3.4. Опциональная глобальная настройка DRBD на HA worker-узлах ..... 72
10.4. Плейбук site.yml ..... 72
10.5. Восстановление после оборванного первого первичного запуска HA ..... 74
10.5.1. Граница обычного повторного запуска ..... 74
10.5.2. Карта стадий для первого первичного запуска HA ..... 74
10.5.3. Диагностика ..... 75
10.5.4. Путь восстановления по стадиям ..... 78
10.6. Проверка корректности поднятия HA ..... 80
10.7. Возможные ошибки конфигурации ..... 81
10.8. Дальнейшие шаги после развёртывания HA ..... 82
**11. ЗАВЕРШЕНИЕ УСТАНОВКИ ..... 83**
**12. ОСОБЕННОСТИ ПЕРЕУСТАНОВКИ КОМПОНЕНТОВ ..... 87**
**13. ЛОГИРОВАНИЕ ..... 88**
**14. КРИТИЧЕСКИЕ ПРЕДУПРЕЖДЕНИЯ ..... 89**
**15. УДАЛЕНИЕ СТРУКТУРНЫХ КОМПОНЕНТОВ SDS ..... 90**
**16. ПОЛНОЕ УДАЛЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ..... 91**
16.1. Удаление пакетов формата deb ..... 91
16.2. Очистка LVM-томов и блочных устройств ..... 91

# 1. ОПИСАНИЕ ПРОЕКТА

ISO-образ включает программное обеспечение, предназначенное для установки и настройки компонентов TROK.

Программа, которая позволяет установить TROK, реализована на языке C++ с применением фреймворка Qt, стандартной библиотеки и библиотеки YAML-CPP для работы с конфигурационными файлами.

Основные компоненты программного обеспечения TROK разработаны на языке Go с использованием пакетов и основных библиотек protobuf, gRPC и yaml. Пользовательский интерфейс реализован с применением JavaScript и библиотеки React.

## 2. ОПИСАНИЕ КОМПОНЕНТОВ

Программа поддерживает установку следующих компонентов:

– Controller: центральный контроллер для управления кластером.

– Worker: системная утилита для управления хранилищем.

– WEBUI: веб-интерфейс для управления TROK.

– Harbor: компонент, необходимый для обеспечения возможности предоставления ресурсов по протоколам NFS, iSCSI или SMB.

– Controller – центральный контроллер для управления кластером. Отвечает за координацию работы всех узлов, обработку команд администратора, управление конфигурацией и состояние кластера.

– Worker – системная утилита для управления хранилищем. Выполняется на каждом узле хранения, обрабатывает команды контроллера, управляет локальными пулами хранения (LVM, ZFS) и обеспечивает репликацию данных через DRBD.

– WEBUI – веб-интерфейс для управления TROK. Предоставляет графический интерфейс для мониторинга и администрирования кластера, позволяя выполнять основные операции без использования командной строки.

– Harbor — компонент, необходимый для обеспечения возможности предоставления ресурсов по протоколам NFS, iSCSI или SMB. Реализует шлюзы для экспорта томов DRBD в виде сетевых ресурсов, доступных клиентам по стандартным протоколам файлового или блочного доступа. Включает в себя настройку и управление сервисами NFS, Samba и iSCSI Target.

– trok-s3-gw — компонент, обеспечивающий предоставление ресурсов по протоколу S3. Реализует S3-совместимый шлюз, позволяющий использовать хранилище TROK как объектное хранилище. Клиенты могут подключаться к нему через стандартные S3 API-интерфейсы, используя любые S3-совместимые приложения. В комплекте поставляется отдельный веб-интерфейс, позволяющий удобно управлять файлами и папками в объектном хранилище через браузер без использования командной строки и дополнительных клиентских приложений.

Программа установки поддерживает три режима работы:

– Консольный режим: установка с использованием командной строки.

– Режим с ключами: запуск с указанием параметров через аргументы командной строки.

– Интерактивный режим: если программа установки запущена без ключей, пользователю будет предложено выбрать тип установки и дополнительные компоненты через графический интерфейс.

## 3. ТРЕБОВАНИЯ

Минимальная конфигурация аппаратных средств для корректной работы TROK представлена ниже:

– Операционная система специального назначения Astra Linux Special Edition. Конкретные поддерживаемые версии операционной системы указаны в релизных примечаниях к требуемой версии программного обеспечения. Допускается запуск программного обеспечения на операционной системе уровня защищённости Орел.

– Для установки ПО TROK пользователь должен обладать правами администратора (sudo).

Минимальные требования к аппаратному обеспечению (к серверам хранения данных, предназначенным для сохранения информации, и вычислительным узлам, которые используются для обработки этой информации):

Таблица 1 – Аппаратные требования по типам узлов

| Компонент | CPU | RAM | Дисковое пространство | Примечания |
|---|---|---|---|---|
| Контроллер | 1 чип с 4 ядрами и более<br>Частота: от 2 ГГц | Минимальный объем: 8 ГБ<br>Рекомендуемый объем: 64 ГБ и более<br>Тип: DDR4, DDR5, ECC (защита от сбоев) | Минимум 2 накопителя:<br>• Дисковое пространство для операционной системы и программного обеспечения емкостью 128 ГБ и более | 2 сетевых порта с пропускной способностью 10 Гбит/с |

| | | | | |
|---|---|---|---|---|
| | | | (SAS/SATA HDD/SSD);<br>• Дисковое пространство для хранения данных TROK: 1 накопитель (и более) с емкостью не менее 128 ГБ (SAS/SATA HDD). | |
| Worker (с хранилищем) | 1 чип с 4 ядрами и более<br>Частота: от 2 ГГц | Минимальный объем: 8 ГБ<br>Рекомендуемый объем: 64 ГБ и более<br>Тип: DDR4, DDR5, ECC (защита от сбоев)<br>Тип: DDR4, DDR5, ECC (защита от сбоев) | 2 накопителя:<br>• Дисковое пространство для операционной системы и программного обеспечения емкостью 128 ГБ и более (SAS/SATA HDD/SSD);<br>• Дисковое пространство для хранения данных TROK | 2 сетевых порта с пропускной способностью 10 Гбит/с |

| | | | | |
|---|---|---|---|---|
| | | | (серверы хранения данных): 1 накопитель (и более) с емкостью не менее 1 ТБ (SAS/SATA HDD). | |
| Diskless-worker | 1+ ядро | Минимум: 1 ГБ | Накопитель для операционной системы и программного обеспечения емкостью 128 ГБ и более (SAS/SATA HDD/SSD) | Используется для кворума или как initiator NVMe-oF |
| Клиент (CLI/GUI) | 1 ядро | 512 МБ | Накопитель для операционной системы и программного обеспечения емкостью 128 ГБ и более (SAS/SATA HDD/SSD) | Только для управления |

Таблица 2 – Список используемых портов

| Служба | Порт | Протокол | Назначение | Критичность |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Controller | 50000 | TCP | REST API и CLI-клиенты | Высокая |
| Worker | 50002 | TCP | Внутренняя коммуникация с контроллером (3367 является резервным портом) | Высокая |
| DRBD Replication | 7000 - 7999 | TCP | Синхронная репликация данных | Максимальная |
| NVMe-oF | 4420 | TCP | Экспорт томов через NVMe/TCP | Опционально |
| S3 API | 7070 | TCP | Предоставление доступа к объектному хранилищу по протоколу S3 (совместим с AWS S3 API) | Средняя |
| WebUI S3 | 7071 | TCP | Веб-интерфейс для управления файлами в объектном хранилище S3 (загрузка, скачивание, создание папок и корзин) | Средняя |
| Admin API | 7080 | TCP | Административный REST API для управления настройками S3-шлюза | Средняя |

| | | | | |
|---|---|---|---|---|
| | | | и мониторинга его состояния | |

Важно. Диапазон портов DRBD по умолчанию – 7000-7999.

При необходимости проверить, какие порты заняты, введите команду:

```
sudo ss -tulnp
```

- -t: TCP-порты;
- -u: UDP-порты;
- -l: Только слушающие порты (LISTEN);
- -n: Выводить номера портов (без резолвинга имён);
- -p: Показать процессы (требует sudo).

Функция будет доступна только после установки компонентов.

# 4. ПОДГОТОВКА ОПЕРАЦИОННОЙ СИСТЕМЫ К УСТАНОВКЕ

Выполните стандартную процедуру установки операционной системы специального назначения Astra Linux Special Edition с соблюдением требований к аппаратному обеспечению и файловым системам. Поддерживаемые версии операционной системы указаны в примечаниях к выпуску (release notes SDS TROK).

**Важно.** В процессе установки операционной системы, если виртуальный или физический сервер будет использоваться в качестве узла хранения данных, обязательно произведите разметку дисков, заранее выделив необходимое пространство для хранения (отдельный диск для хранения данных).

В ходе установки необходимо выбрать уровень защищённости операционной системы «Орел».

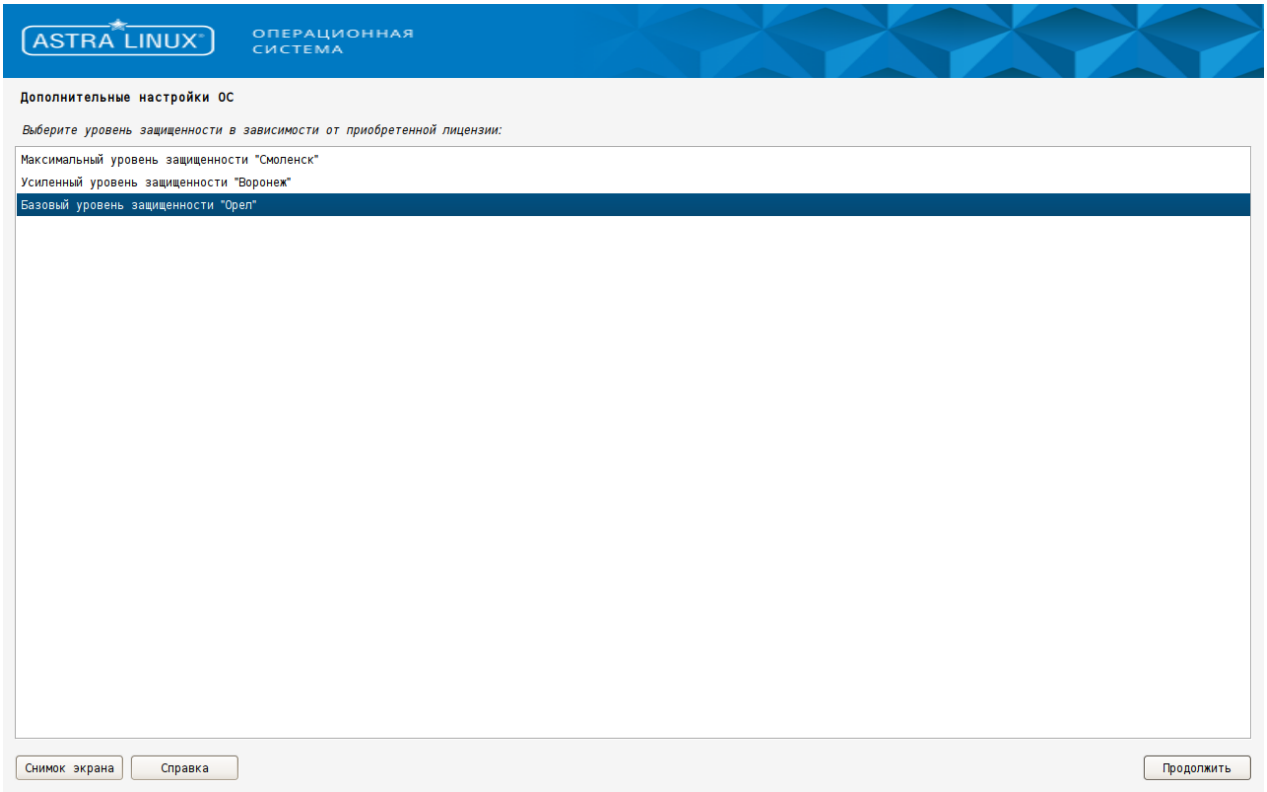



Рисунок 1 – Меню выбора уровня защищенности

В зависимости о типа iso образа, который вы используете, меню выбора выглядит немного иначе.

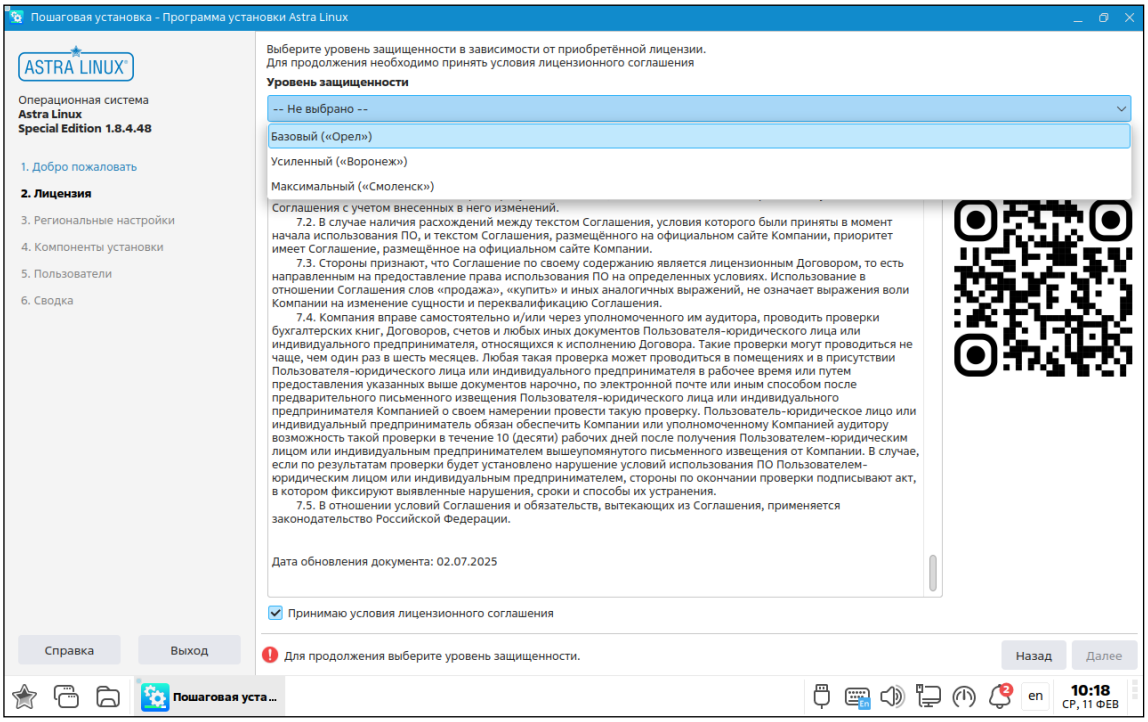



Рисунок 2 – Альтернативное меню выбора уровня защищенности

На последующих этапах установки, если вы устанавливаете Astra Linux SE версии 1.8.x система предложит выбрать одну из двух версий ядра Linux.

Рекомендуется использовать ядро linux-6.1-generic.

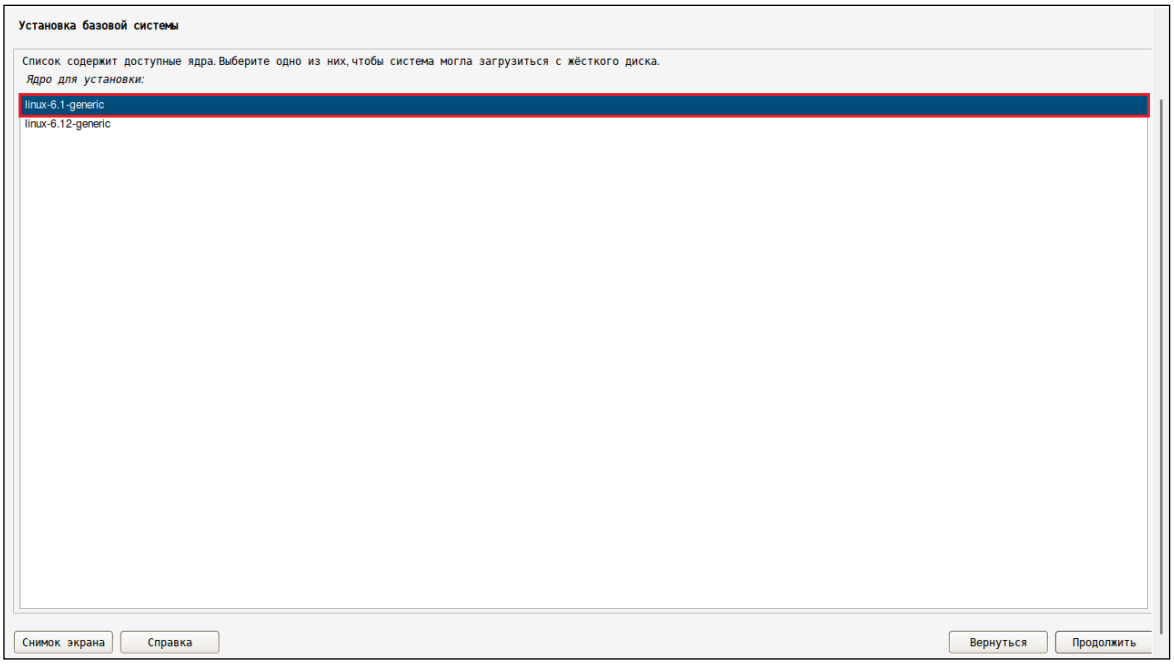



Рисунок 3 – Меню выбора ядра Linux в процессе установки Astra Linux SE 1.8.x

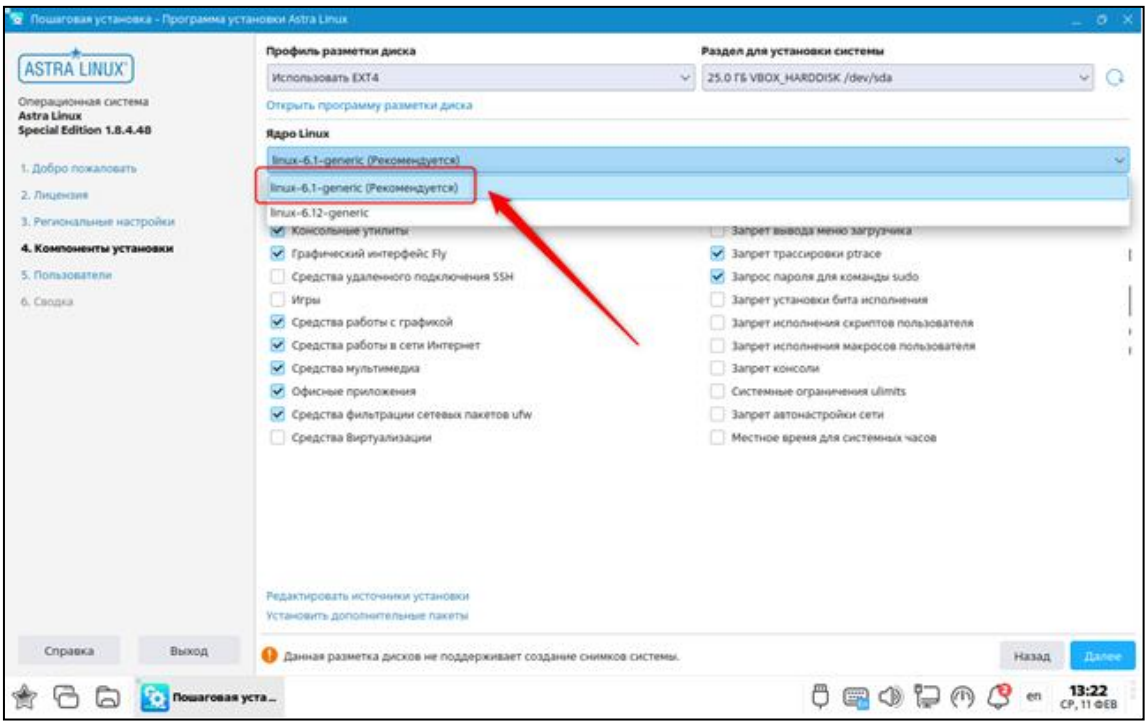



Рисунок 4 – Альтернативное меню выбора ядра Linux в процессе установки Astra Linux SE 1.8.x

Если вы устанавливаете Astra Linux SE версии 1.7.x система предложит выбрать альтернативные версии ядра Linux:

- linux-5.10-generic;
- linux-5.10-hardened;
- linux-5.15-generic;
- linux-5.15-hardened;
- linux-5.15-lowlatency;
- linux-5.4-generic;
- linux-5.4-hardened.

В этом случае допускается выбор любой версии ядра Linux, удовлетворяющей вашим задачам. Такой выбор не оказывает влияния на процедуру установки программного обеспечения.

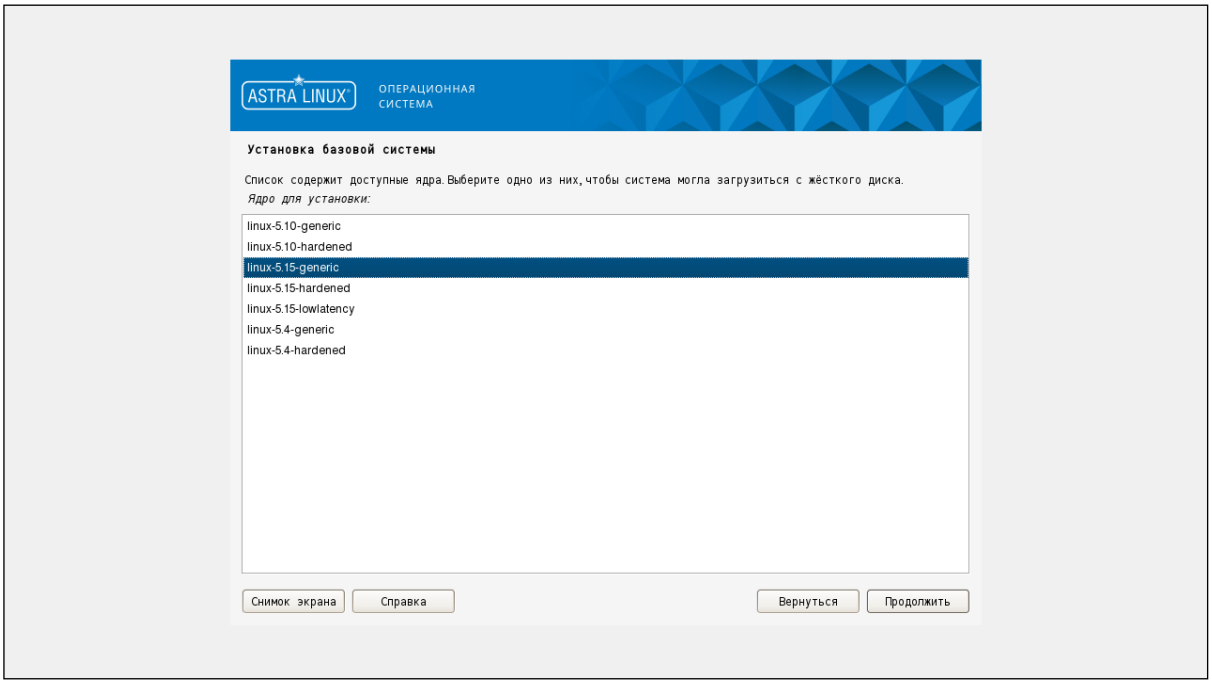



Рисунок 5 – Меню выбора ядра Linux в процессе установки Astra Linux SE 1.7.x

После завершения процесса установки операционной системы дождитесь её перезагрузки, затем выполните вход в систему, используя учётные данные администратора, указанные на этапе установки.

### 4.1. Подготовка операционной системы специального назначения Astra Linux Special Edition к установке ПО TROK

Откройте файл конфигурации репозиториев операционной системы (например, с помощью команды sudo vim /etc/apt/sources.list) и проанализируйте его содержимое согласно дальнейшей инструкции. Данный файл содержит перечень репозиториев программного обеспечения, используемых системой для установки и обновления пакетов. Состав и структура этого списка зависят от используемой версии операционной системы.

**Для установки программного обеспечения TROK рекомендуется использовать операционные системы, настроенные на работу с репозиториями ветки «frozen» либо с репозиториями, размещёнными на установочном дистрибутиве Astra Linux Special Edition. Выбор и использование репозиториев должен осуществляться строго в соответствии**

**с версией установленной операционной системы, чтобы обеспечить корректную установку и функционирование программного обеспечения.**

Использование репозиториев ветки «stable» нежелательно. При работе с данным типом репозиториев существует риск автоматической установки неподходящей версии ядра, что может привести к конфликтам зависимостей и некорректной работе системы.

В пунктах 4.1.1 и 4.1.2 представлены два возможных варианта настройки, которые следует выбирать в зависимости от уровня защищенности операционной системы, которую вы используете.

### 4.1.1. Настройка репозиториев для систем с доступом к сети Интернет (настройка для использования ветки репозиториев «frozen»)

Процесс изменения репозиториев сводится к редактированию основного файла /etc/apt/sources.list.

Перед началом изменений узнайте номер нужного обновления (например, 1.8.4) из официального бюллетеня безопасности. Также можно проверить версию с помощью ввода команды:

```
cat /etc/astra_version
```

В файле измените строки, начинающиеся с deb https://dl.astralinux.ru/astra/stable/..., заменив stable на frozen и добавив номер обновления.

Для версии 1.8.4 список репозиториев выглядит следующим образом:

Было (ветка stable):

```
# Основной репозиторий, включающий актуальное оперативное или срочное обновление

deb            https://dl.astralinux.ru/astra/stable/1.8_x86-64/main-repository/ 1.8_x86-64 main contrib non-free non-free-firmware

# Расширенный репозиторий, соответствующий актуальному оперативному обновлению

deb        https://dl.astralinux.ru/astra/stable/1.8_x86-64/extended-repository/ 1.8_x86-64 main contrib non-free non-free-firmware
```

Стало (ветка frozen, обновление 1.8.4):

```
   deb      https://dl.astralinux.ru/astra/frozen/1.8_x86-64/1.8.4/main-repository/
1.8_x86-64 main contrib non-free non-free-firmware

   deb  https://dl.astralinux.ru/astra/frozen/1.8_x86-64/1.8.4/extended-repository/
1.8_x86-64 main contrib non-free non-free-firmware
```

Если строка закомментирована (в начале строки ссылки на репозиторий присутствует знак «#»), то ее необходимо раскомментировать (удалите знак «#» для активации доступа к репозиториям и обеспечения обновлений).

**Важно.** При переходе на новое оперативное или срочное обновление номер обновления в ссылке необходимо будет также изменить вручную.

Поддерево «frozen»: репозитории Astra Linux x.7 выглядят немного иначе:

```
   # Базовый репозиторий

   deb      https://dl.astralinux.ru/astra/frozen/1.7_x86-64/1.7.3/repository-base/
1.7_x86-64 main contrib non-free

   # Расширенный репозиторий

   deb  https://dl.astralinux.ru/astra/frozen/1.7_x86-64/1.7.3/repository-extended/
1.7_x86-64 main contrib non-free

   # Базовый репозиторий срочные обновления

   deb https://dl.astralinux.ru/astra/frozen/1.7_x86-64/1.7.3/uu/2/repository-base/
1.7_x86-64 main contrib non-free

   # Расширенный репозиторий срочные обновления

   deb      https://dl.astralinux.ru/astra/frozen/1.7_x86-64/1.7.3/uu/2/repository-
extended/ 1.7_x86-64 main contrib non-free
```

Выполните обновление кэша пакетов, используемого системой для отслеживания актуальных версий доступного программного обеспечения, с помощью команды:

```
   sudo apt update
```

Команда инициирует повторное считывание и актуализацию индексов пакетов из подключённых репозиториев.

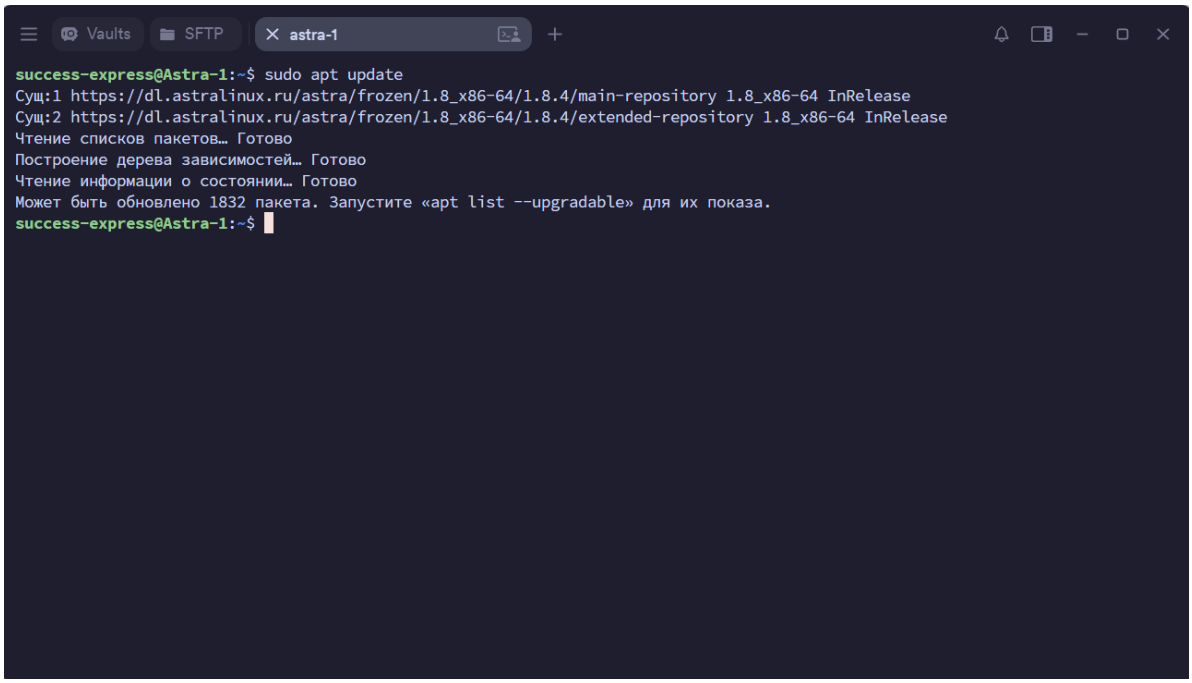



Рисунок 6 – Пример результата выполнения команды `sudo apt update`

После этого перейдите к процедуре монтирования ISO-образа программы TROK (п. 4.2).

### 4.1.2. Перенастройка репозиториев для автономных систем (локальные репозитории iso образа ОС)

Для получения данных из локальных репозиториев также требуется произвести редактирование файла /etc/apt/sources.list, однако в случае с локальными репозиториями необходимо внести другие изменения.

Закомментируйте строки (добавьте знак «#» перед началом строки), начинающиеся с deb https://dl.astralinux.ru/astra/stable/... или deb https://dl.astralinux.ru/astra/frozen/...

Для установки в закрытом контуре укажите источник cdrom:

Для версии 1.8.1 Astra Linux Special Edition строка со ссылкой на локальные репозитории выглядит следующим образом:

```
   deb cdrom:[OS Astra Linux 1.8.1.6  DVD]/ 1.8_x86-64 contrib main non-free non-free-firmware
```

Для версии Astra Linux Special Edition 1.7.5 следующий пример:

```
deb cdrom:[OS Astra Linux 1.7.5 1.7_x86-64 DVD]/ 1.7_x86-64 contrib main non-free
```

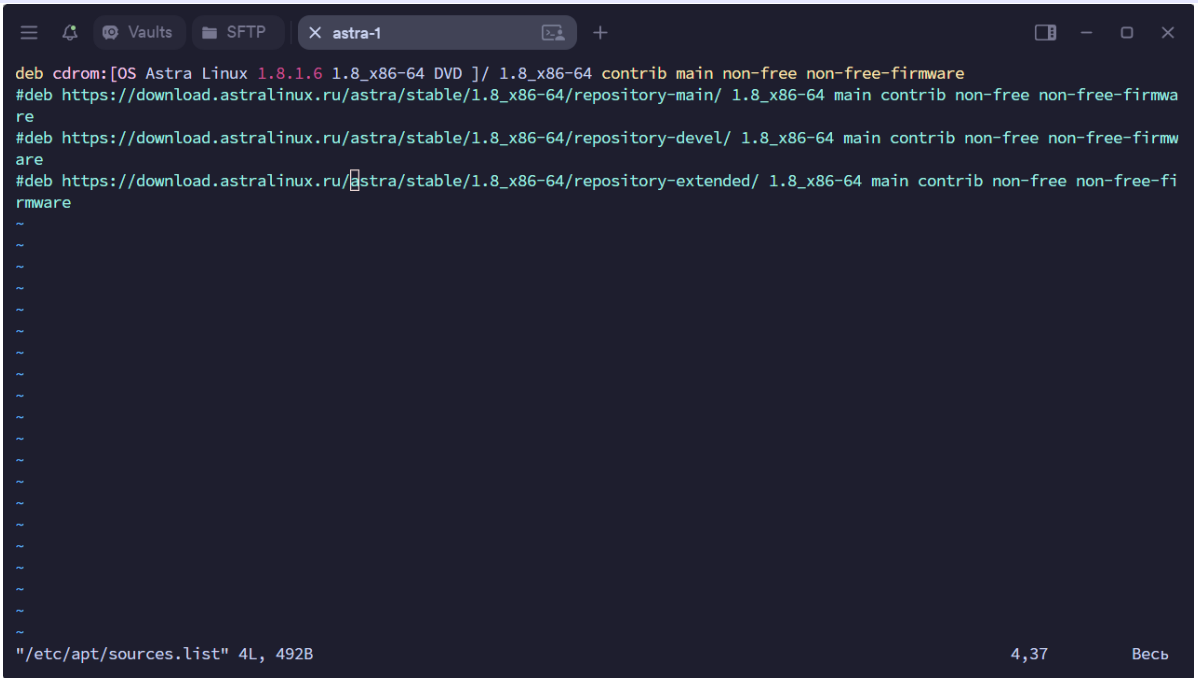



Рисунок 7 – Файл sources.list с внесенным адресом локальных репозиториев

**В случае ручного монтирования** установочного образа, строку в файле /etc/apt/sources.list необходимо заменить на соответствующую альтернативу. В качестве примера рассмотрим конфигурацию для версии 1.7.1 Astra Linux Special Edition:

```
deb file:///media/cdrom/ 1.7_x86-64 contrib main non-free
```

, где:

file:///media/cdrom/ – путь к каталогу, в который был смонтирован установочный образ,

1.7_x86-64 – соответствует версии и архитектуре операционной системы,

contrib, main, non-free — обозначают секции репозитория, включающие основной, дополнительный и несвободный программный продукт соответственно.

После сохранения файла /etc/apt/sources.list следует также выполнить обновление кэша пакетов, используемого системой для отслеживания актуальных версий доступного программного обеспечения, с помощью команды:

```
sudo apt update
```

## 4.2. Монтирование образа TROK

Перед установкой ПО TROK необходимо убедиться в наличии необходимой версии библиотеки libyaml-cpp для чтения и создания YAML-файлов. Требуемая версия этой библиотеки зависит от используемой архитектуры и версии операционной системы.

Версию библиотеки для вашей конкретной платформы можно узнать с помощью команды:

```
apt-cache search libyaml-cpp
```

Команда покажет все доступные в репозиториях пакеты, содержащие libyaml-cpp.

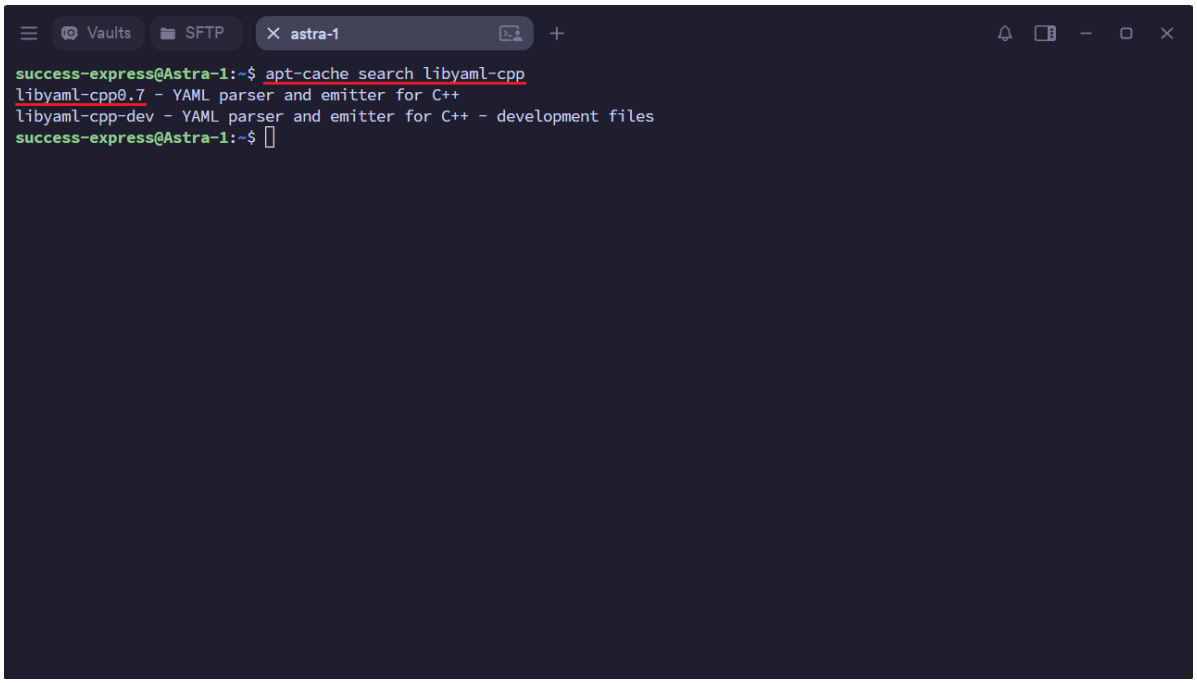



Рисунок 8 – Пример вывода команды apt-cache search libyaml-cpp

В качестве примера ниже приведена попытка установки пакета устаревшей версии. Система сообщает, что указанный пакет не найден в доступных репозиториях.

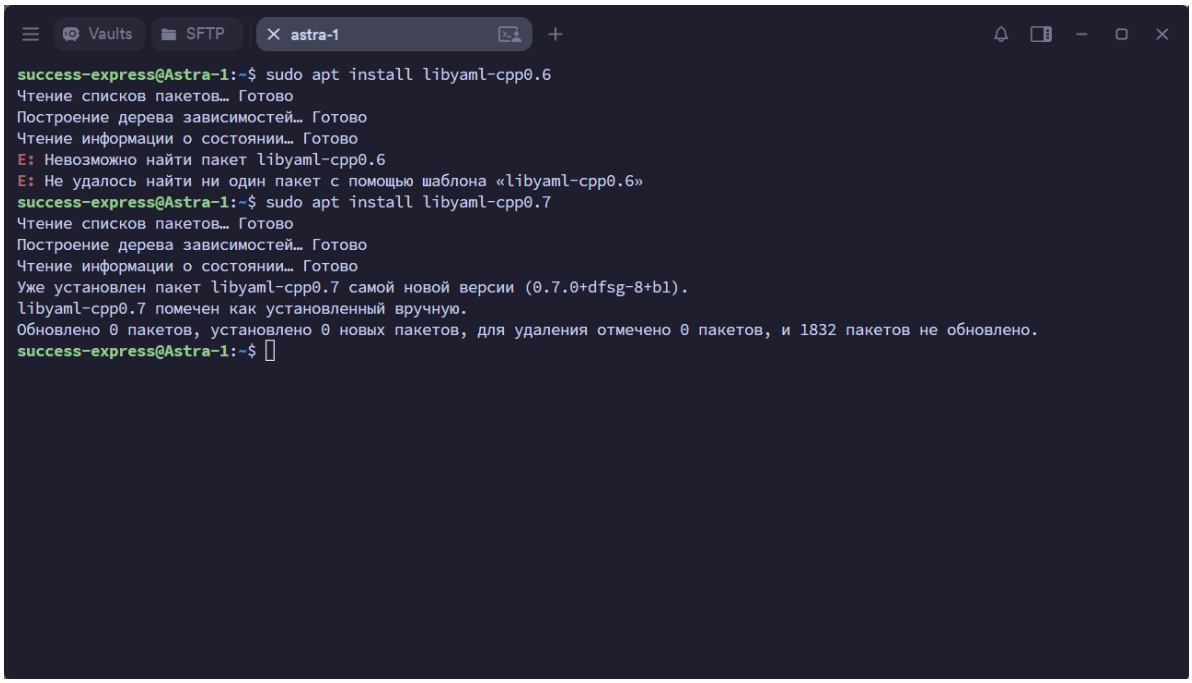



Рисунок 9 – Пример реакции системы на попытку установки неподходящего пакета libyaml-cpp

Для специальной версии TROK, предназначенной для архитектуры ARM, чаще требуется предварительная установка пакета libyaml-cpp0.6. Установку можно выполнить командой:

```
sudo apt install libyaml-cpp0.6
```

В ряде версий Astra Linux Special Edition (например, CH ALSE 1.8.3uu1 для архитектуры amd64 (x86_64)), установка или работа программного обеспечения может требовать наличие более новой версии библиотеки – libyaml-cpp0.7.

Для установки воспользуйтесь командой:

```
sudo apt install libyaml-cpp0.7
```

Для установки программного обеспечения TROK используется ISO-образ. Перед началом установки необходимо выполнить монтирование образа, чтобы получить доступ к его содержимому.

Скопируйте образ диска в удобное для вас место в операционной системе.

Создайте каталог /mnt/trok_installer используя команду:

```
sudo mkdir /mnt/trok_installer
```

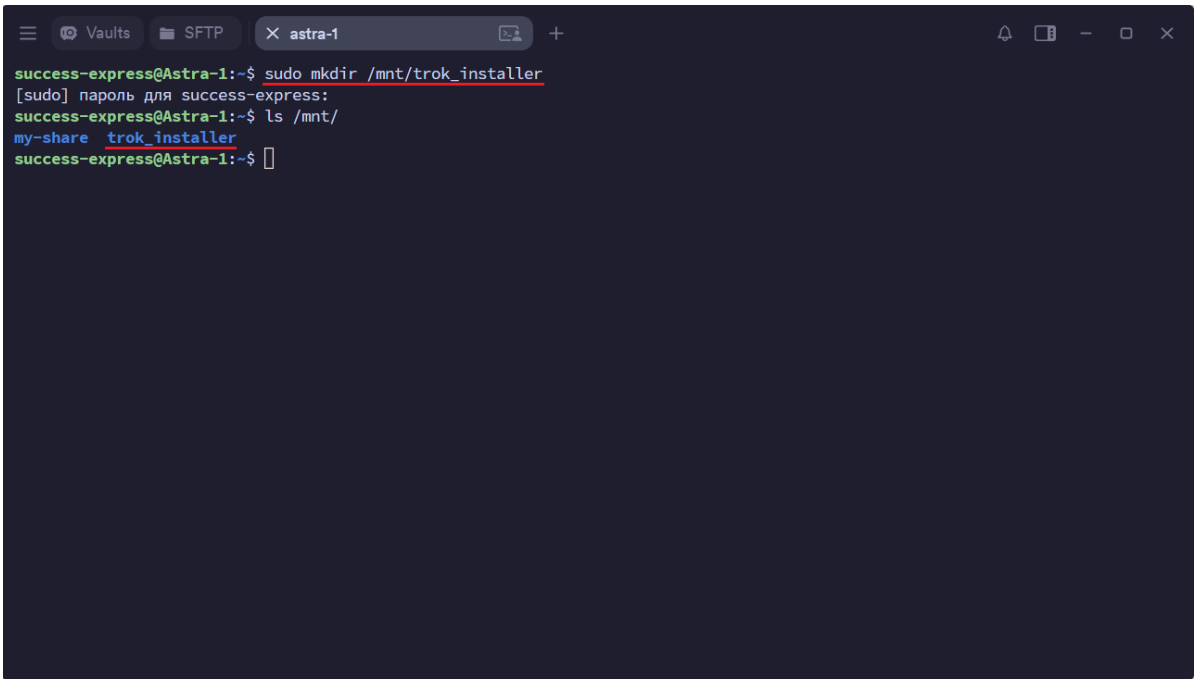



Рисунок 10 – Результат выполнения команды sudo mkdir /mnt/trok_installer

С помощью команды:

```
sudo mount -o loop <место_расположения_ISO-образа> /mnt/trok_installer
```

производится монтирование ISO-файла в созданный каталог с использованием опции loop, позволяющей обращаться с образом как с виртуальным блочным устройством.

(например, `sudo mount -o loop /home/admin/TPOK-2.0.iso /mnt/trok_installer`)

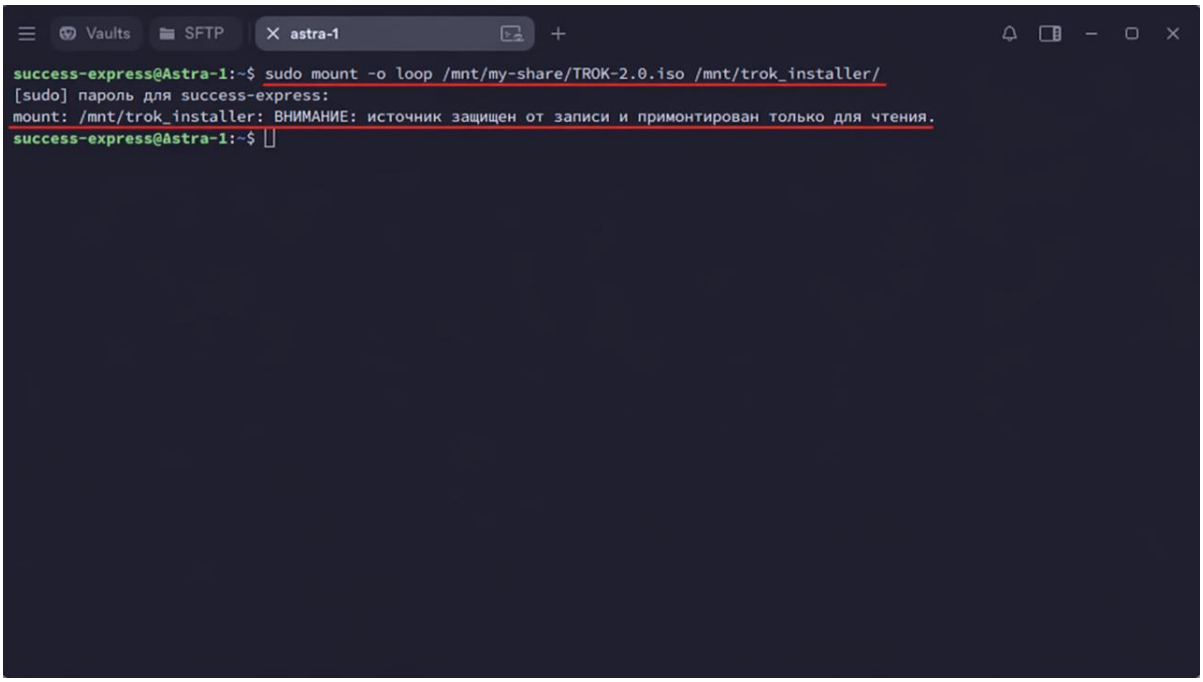


Рисунок 11 – Уведомление системы об успешном завершении процедуры монтирования образа

Для проверки успешного монтирования и просмотра содержимого образа используйте команду:

```
ls /mnt/trok_installer
```

**Важно!** Перед началом процедуры установки перейдите в каталог, в который был смонтирован образ:

```
cd /mnt/trok_installer/installer
```

**Далее все команды установки предполагают, что текущая рабочая директория – /mnt/trok_installer/installer**.

Если по каким–то причинам вы выполняете команды из другого каталога, замените префикс «./» в командах вида sudo ./trok_installer на соответствующий абсолютный или относительный путь к исполняемому файлу, например: sudo /mnt/trok_installer/installer/trok_installer.

# 5. КОНФИГУРАЦИЯ

Программа для установки использует YAML-конфигурацию настройки репозиториев и пакетов. Файл расположен в iso образе TROK и доступен после выполнения процедуры монтирования образа.

Пример файла config.yaml:

```
   options:
     archive: ../repository/TROK-2.0.tar.gz
     repository_entry: deb [arch=amd64 signed-by=/usr/share/keyrings/trok-archive-
keyring.gpg] file:///opt/TROK/SDS/apt-repo stable main
     repository_config: /etc/apt/sources.list.d/trok_repo.list
     keyring_file: ../repository/trok-archive-keyring.gpg
     keyring_dest: /usr/share/keyrings/trok-archive-keyring.gpg
     eula_file: ../EULA_SDS_Trok.txt
     eula_dest: /usr/share/doc/trok/EULA_SDS_Trok.txt
     log: /var/log/trok/trok-installer
     autoinstall_path: /usr/sbin/trok-drbd-autoinstall
     ansible_collection_archive: ../ansible-collection/astra-trok-2.0.0.tar.gz
     eula_path: /usr/share/doc/trok
   packages:
     worker:
     - name: trok-worker-meta
       version: ''
     controller:
     - name: trok-controller-meta
       version: ''
     harbor:
     - name: trok-harbor-meta
       version: ''
     webui:
     - name: nginx
       version: ''
     - name: trok-webui
       version: ''
     s3:
     - name: trok-s3-gw
       version: ''
     satellite: []
     gateway: []
     brest: []
     base: []
   repository:
     archive: ../repository/TROK-2.0-rc.1.tar.gz
     entry: deb [arch=amd64 signed-by=/usr/share/keyrings/trok-archive-keyring.gpg]
file:///opt/TROK/SDS/apt-repo stable main
     log: /var/log/trok/trok-installer
```

При необходимости Вы можете изменить путь к директории для сохранения лог-файлов заранее, используя указанный файл конфигурации, указав в параметре log нужные значения.

В описании устанавливаемых пакетов в случае, если значение version остается пустым, программа установки автоматически осуществляет загрузку самой последней версии пакета. В случае явного указания конкретного значения параметра version будет загружена и установлена именно эта версия.

# 6. УСТАНОВКА С ИСПОЛЬЗОВАНИЕМ КОМАНДНОЙ СТРОКИ

Для установки программного обеспечения TROK в папке installer предоставляется 2 программы установки на выбор:

– установка с использованием командной строки;

– установка с применением графического интерфейса пользователя.

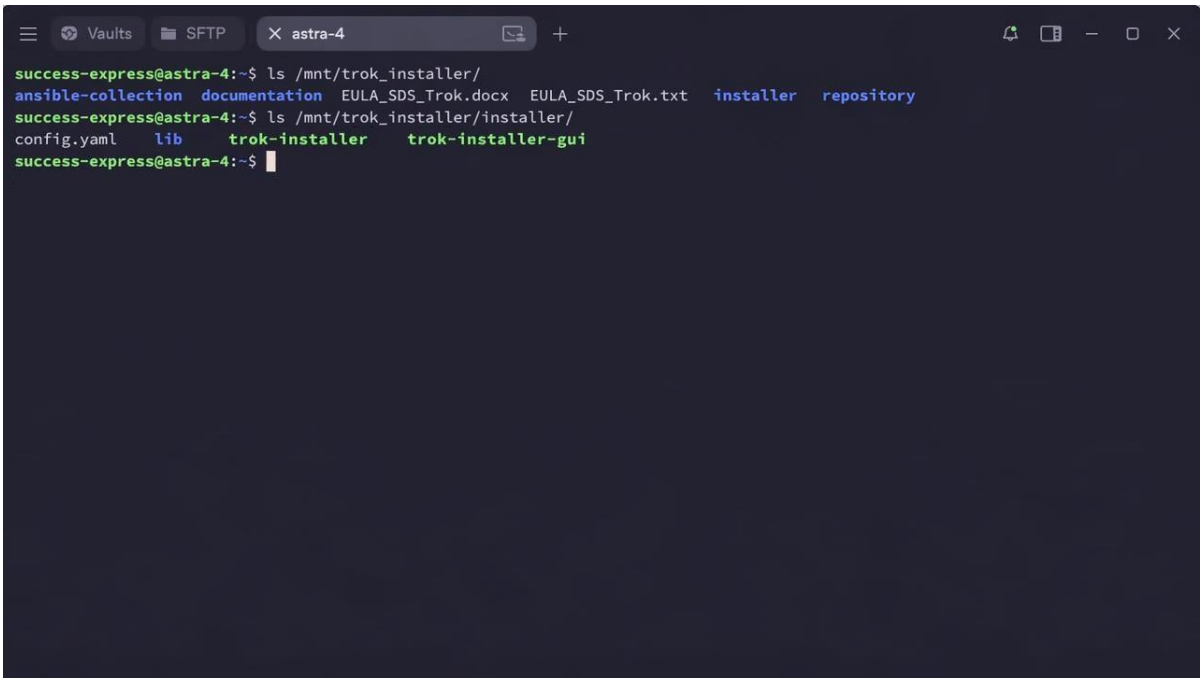



Рисунок 12 – Пример размещения установочных файлов в структуре iso образа

Для установки с использованием консоли введите команду (предварительно нужно зайти в папку, в которой располагается trok-installer):

```
sudo ./trok-installer
```

Далее в руководстве все команды в примерах будут указаны на примере версии Astra Linux Special Edition 1.8.

Программа поддерживает следующие аргументы командной строки:

Таблица 3 – Аргументы, используемые при установке в режиме с ключами

| Аргумент | Описание |
|---|---|
| --worker | Установка узла типа Worker |
| --controller | Установка узла типа Controller |

| | |
|---|---|
| --combined | Установка узла типа Combined (Worker + Controller) |
| --webui | Установка WEBUI и nginx |
| --log <директория> | Указание пути к директории для сохранения лог-файлов (по умолчанию: /var/log/trok-installer) |
| --console | Запуск с консольным интерфейсом |
| --harbor | Установка Harbor |

Пример использования:

– Установка с указанием пути к директории для сохранения лог-файлов:

```
./trok-installer --log <директория>
```

**Важно.** В процессе ввода пути к директории для сохранения логов вы можете использовать как абсолютный путь (например, /home/user/my_logs/trok), так и относительный (например, ../trok-installer/logs).

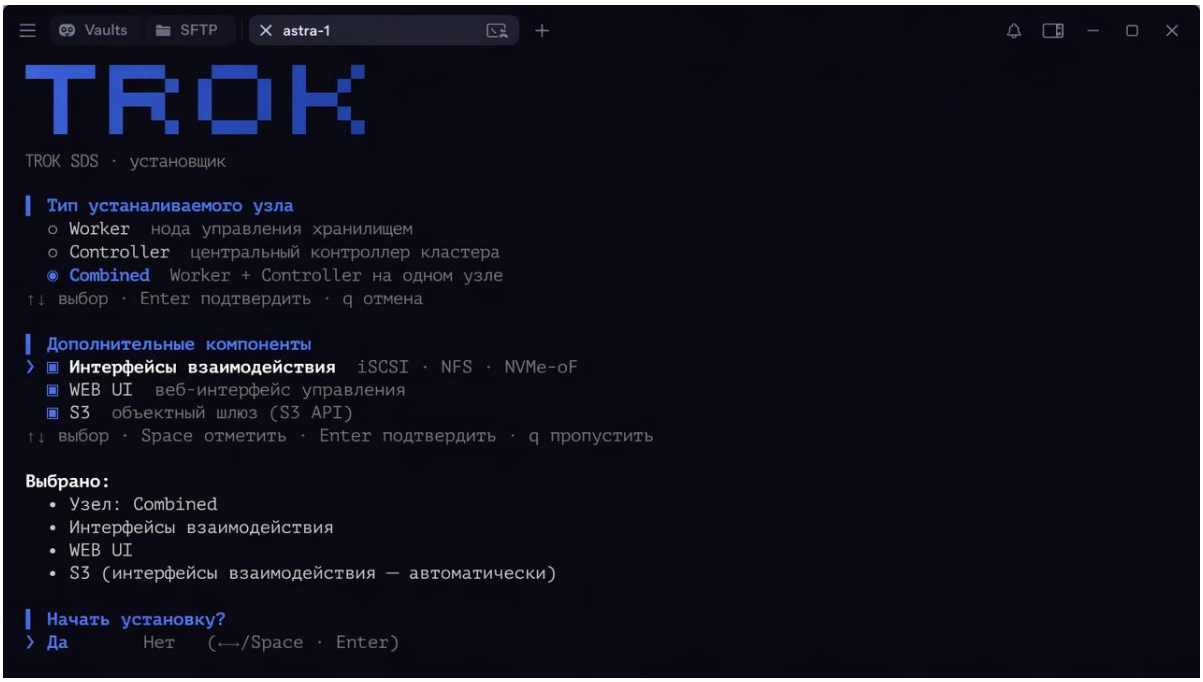



Рисунок 13 – Начало установки программного обеспечения в режиме командной строки

После выполнения установки рекомендуется перезагрузить систему.

В случае возникновения проблем с установкой DRBD смотрите пункт 8.

# 7. УСТАНОВКА В ИНТЕРАКТИВНОМ РЕЖИМЕ (GUI)

В случае использования графического интерфейса в операционной системе и запуска установочного файла trok-installer-gui посредством данного интерфейса автоматически инициируется графический режим конфигурирования.

**Важно.** Для запуска необходимо находиться в директории, в которой располагается установочный файл.

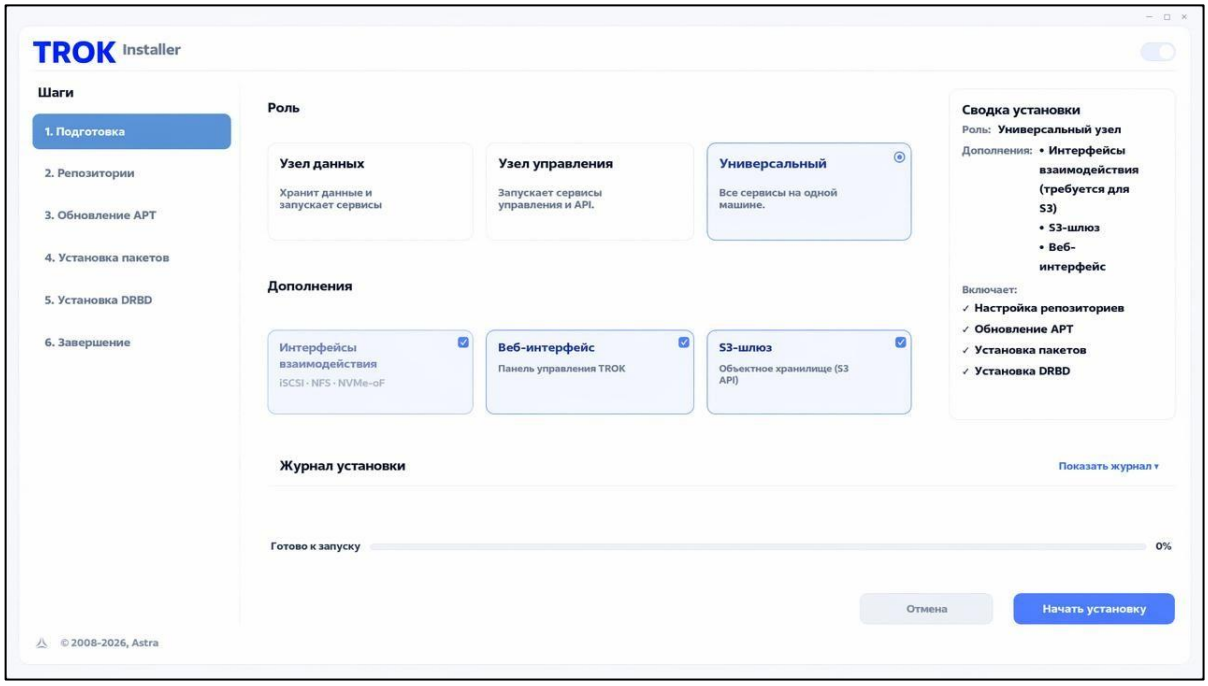



Рисунок 14 – Установка в интерактивном режиме

После запуска программы установки пользователю предоставляется выбор между следующими архитектурными конфигурациями:

– Определение топологии узла:

– Узел данных: Установка узла категории worker (исполнительный компонент).

– Универсальный (combined): Комбинированная установка (worker + controller).

– Узел управления (controller): Развертывание управляющего узла (control plane).

– Компоненты, автоматически устанавливающиеся при установке trok-controller:

– trok-auth: обеспечивает централизованную аутентификацию и авторизацию для всех компонентов TROK через gRPC API, управляя OAuth2 клиентами (приложениями) и выдачей JSON Web Tokens (JWT) в соответствии со спецификацией OpenID.

– trok-cp-endpoint: обеспечивает REST API TROK для управления и мониторинга DRBD - ресурсов. Сервис выступает единой точкой входа для всех операций управления кластером, обеспечивая взаимодействие между пользователями/клиентами и бэкендом контроллера.

– Технические особенности реализации:

– harbor: готовый программный модуль, объединяющий службу автоматизации DRBD (drbd-reactor), набор сценариев управления ресурсами (resource-agents) и поддержку сервисов доступа (iSCSI, NFS, SMB), который поставляется в виде отдельного мета-пакета для установки на узлах хранения, а также входит в состав контроллера.

– WEBUI: Веб-интерфейс управления (включая сервис nginx).

Перед началом установки ознакомьтесь с лицензионным соглашением и нажмите «Я принимаю условия соглашения», если готовы продолжать установку программы.

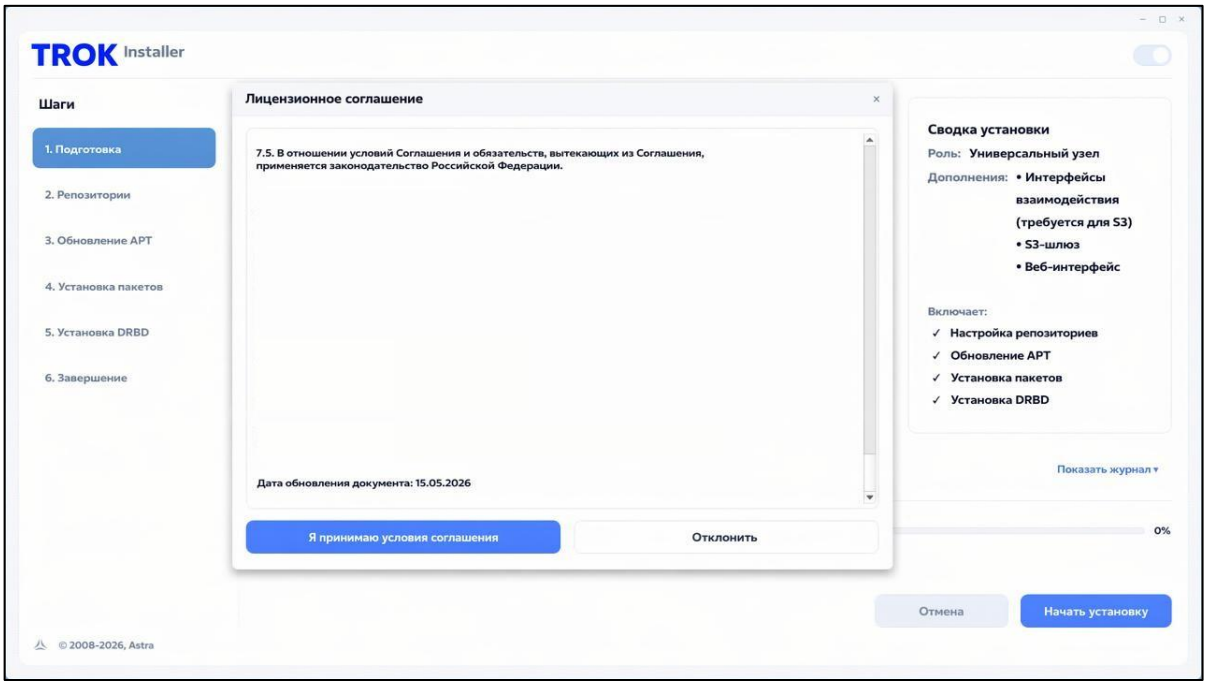


Рисунок 15 – Лицензионное соглашение

Все дальнейшие шаги установки производятся автоматически. По завершении установки отображается сообщение об успешном завершении процесса.

После выполнения установки рекомендуется перезагрузить систему.

В случае возникновения проблем с установкой DRBD смотрите пункт 8.

# 8. УСТАНОВКА С ИСПОЛЬЗОВАНИЕМ ПАКЕТОВ

В зависимости от типа узла может быть установлены различные категории пакетов программ. Ниже приведен вписок доступных пакетов.

Таблица 4 – Пакеты и их назначение

| Пакет | Назначение | На каких узлах устанавливать |
|---|---|---|
| trok-controller-meta | Установка модуля, предназначенного для реализации функций управления данными и координации узлов в составе кластера хранения. | На узлах, выполняющих роль controller. |
| trok-worker-meta | Установка модуля обеспечивающего непосредственное хранение данных, локальное управление и обработку команд, поступающих от управляющего узла (controller). | На узлах, выполняющих роль worker. |
| trok-combined-meta | Установка двух модулей единовременно: управляющего (controller) и модуля хранения данных (worker). | На совмещённых (комбинированных) узлах с функциями controller и worker. |
| trok-webui (+ nginx) | Установка компонента, предоставляющего веб-интерфейс управления с | На комбинированных узлах или узлах controller. |

| | | |
|---|---|---|
| | использованием веб-сервера nginx. | |
| trok-harbor-meta | Установка модуля, реализующего API и функцию шлюза для интеграции с внешними системами. | На комбинированных узлах или узлах controller. |
| trok-s3-gw | Установка модуля, реализующего S3-совместимый шлюз для предоставления объектного хранилища по протоколу S3 API. Включает в себя веб-интерфейс для управления файлами через браузер (порт 7071) и административный API (порт 7080). | На узлах, где требуется предоставление объектного доступа к данным. |

Перед началом установки программного обеспечения с использованием пакетов уточните, какую роль будет играть узел (controller, worker, combined).

Найдите архив с пакетами для установки. В смонтированной директории /mnt/trok_installer в папке repository находится файл TROK-<версия>.tar.gz.

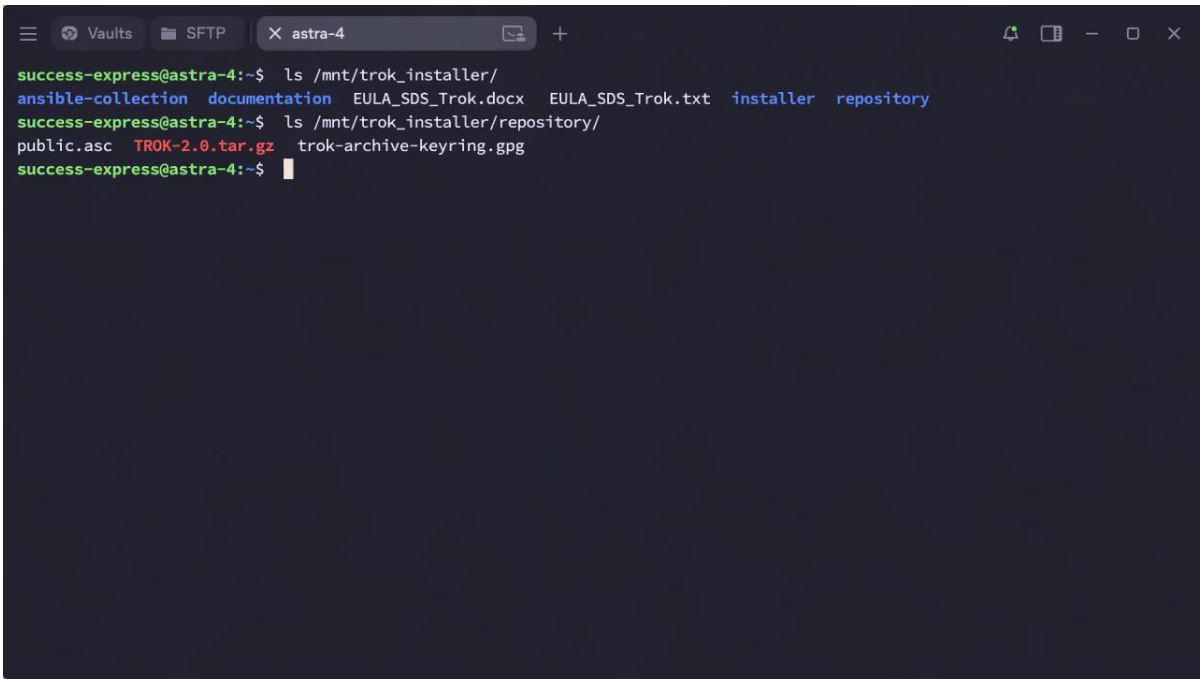



Рисунок 16 – Расположение архива, содержащего пакеты для установки TROK

Выберите место для распаковки архива и создайте в нем папку, например trok-pack. Из-за того, что в смонтированную директорию нельзя вносить изменения, выберите другую папку, например /opt (используется для сторонних программ).

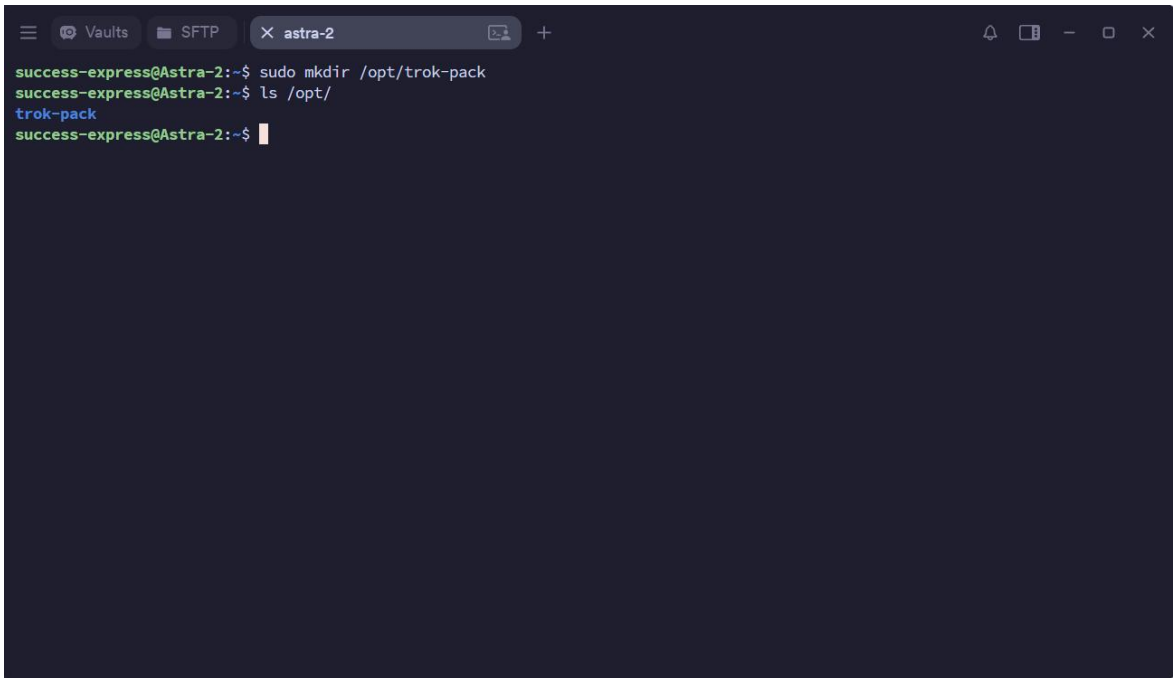



Рисунок 17 – Создание папки для последующего размещения пакетов TROK

Выполните команду для распаковки архива в созданную папку:

```
sudo tar -xzf /mnt/trok_installer/repository/<имя_архива> --overwrite -C /opt/trok-pack/
```

Ключ --overwrite не всегда обязателен, но допустим, если файл уже существует.

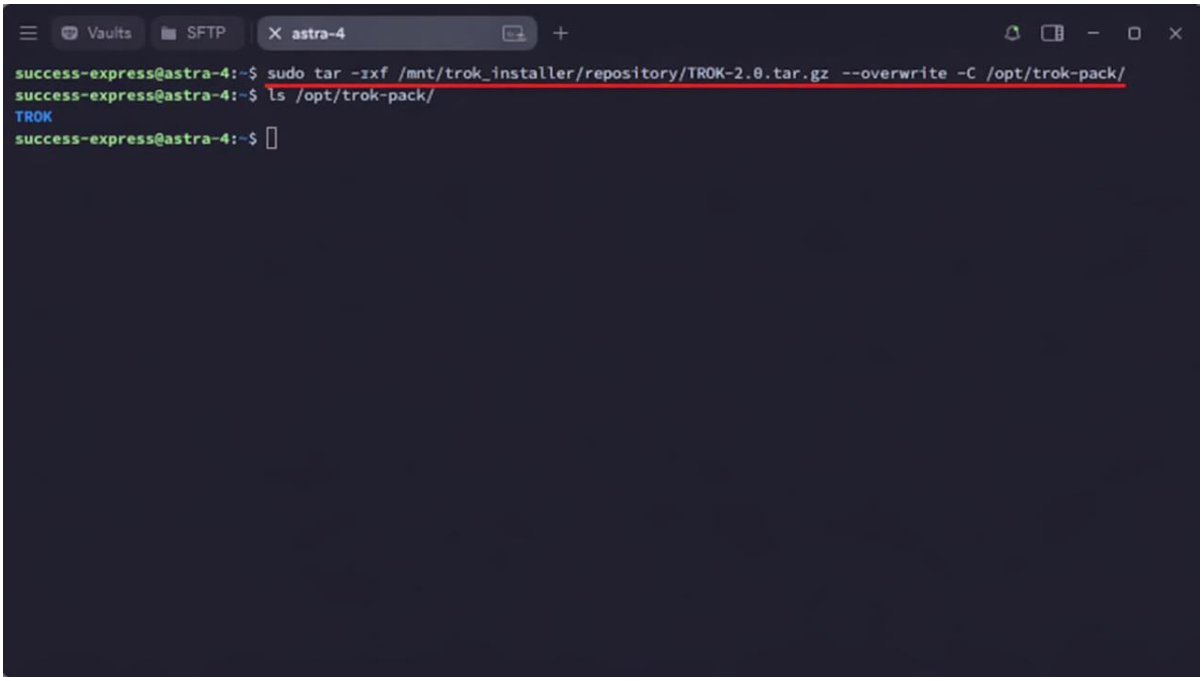



Рисунок 18 – Результат выполнения команды распаковки

Создайте отдельный файл репозитория с помощью команды:

```
sudo vim /etc/apt/sources.list.d/trok_repo.list
```

Добавьте в созданный файл строку репозитория:

```
deb [arch=amd64 signed-by=/usr/share/keyrings/trok-archive-keyring.gpg] file:///opt/trok-pack/SDS/apt-repo stable main
```


Рисунок 19 – Файл репозиториев trok_repo.list после внесения изменений

В файлах смонтированного образа найдите файл trok-archive-keyring.gpg и копируйте его в систему.

Для начала убедитесь, что в директории /usr/share/ есть папка, keyrings. Если ее нет, создайте вручную:

```
sudo mkdir -p /usr/share/keyrings
```

Флаг -p позволяет создать вложенные директории, если их ещё нет, и не выдаёт ошибку, если директория уже существует.

Затем выполните команду копирования:

```
sudo cp /mnt/trok_installer/trok-archive-keyring.gpg /usr/share/keyrings/
```

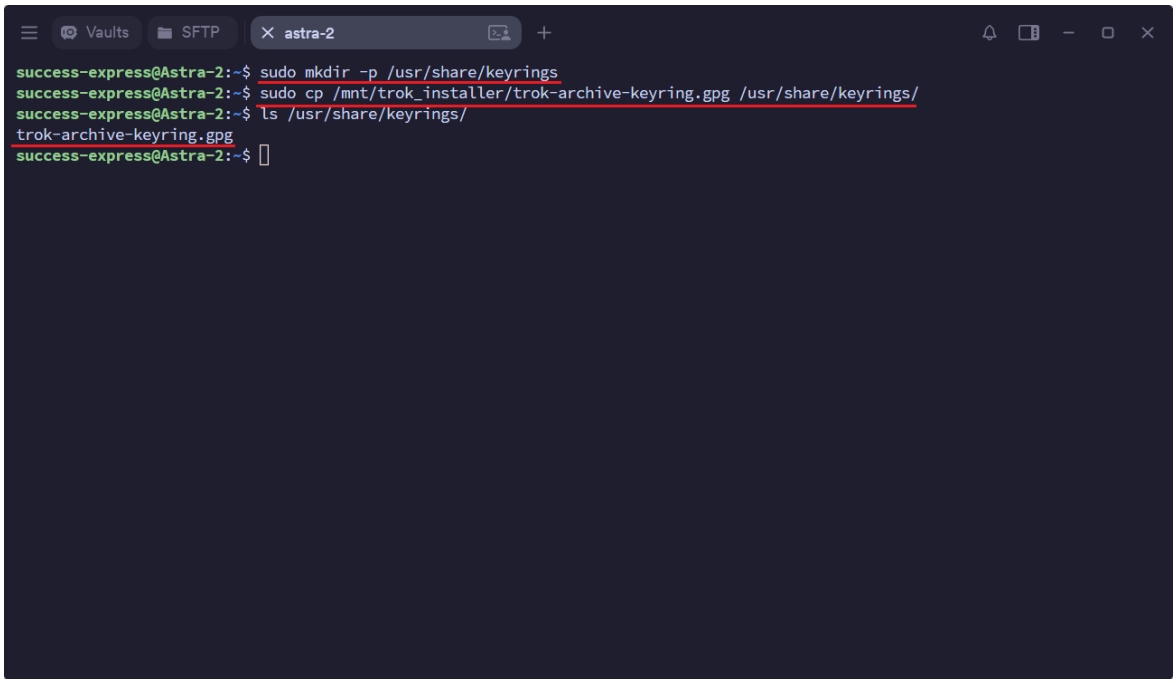



Рисунок 20 – Процесс копирования ключа

Установите корректные права для ключа. Это обеспечит системам безопасности надлежащий доступ к ключу.

```
sudo chmod 644 /usr/share/keyrings/trok-archive-keyring.gpg
```

Создайте файл приоритета:

```
sudo vim /etc/apt/preferences.d/99-trok.pref
```

(Обратите внимание: между 99- и trok.pref не должно быть пробелов).

Добавьте в файл следующее содержимое:

```
Package: *
Pin: release o=TROK
Pin-Priority: 1001
```

Такая запись гарантирует приоритет использования пакетов из вашего локального репозитория при установке.

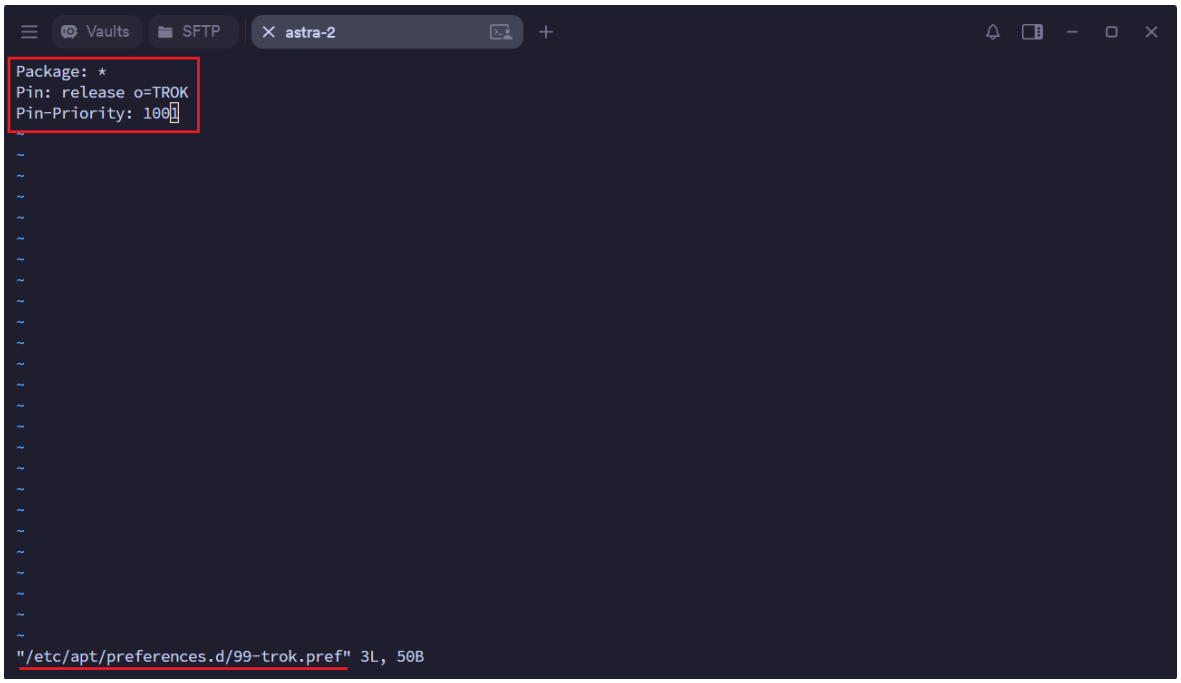



Рисунок 21 – Файл 99-trok.pref после внесения изменений

После выполнения всех подготовительных шагов обновите индекс пакетов. Для этого выполните:

```
sudo apt update
```

Теперь возможно устанавливать программное обеспечение с использованием пакетов.

Если предопределенная роль – выделенный контроллер (controller), начните с установки пакета trok-webui, поскольку управление контроллером осуществляется с его помощью. Обратите внимание, что установка пакетов осуществляется в строгой последовательности. Используйте команду:

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

Веб-сервер nginx устанавливается автоматически во время установки пакета trok-webui.


Рисунок 22 – Уведомление о настройке nginx

Если этого не произошло, воспользуйтесь дополнительной командой:

```
sudo apt install nginx
```

Установите компонент контроллер:

```
sudo apt install trok-controller-meta
```

В качестве завершающего этапа выполните установку пакета trok-harbor-meta:

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

Если предопределенная роль – узел хранения (worker), используйте команду:

```
sudo apt install trok-worker-meta
```

В качестве завершающего этапа выполните установку пакета trok-harbor-meta:

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

После установки обязательно выполните команду:

```
sudo trok-drbd-autoinstall
```

В качестве последнего альтернативного варианта установки выступает комбинированный узел. Для того, чтобы не вводить много однотипных команд, установку пакетов и зависимостей можно произвести одной командой:

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

После установки обязательно выполните команду:

```
sudo trok-drbd-autoinstall
```

Рекомендуется перезагрузить систему, когда все требуемые пакеты будут установлены на узле.

**Важно.** После завершения установки компонента worker, всегда необходимо выполнить команду для запуска программы инсталляции DRBD от имени суперпользователя (root): sudo trok-drbd-autoinstall.

**Важно.** В случае, если в вашей операционной системе возникают проблемы с установкой DRBD, например, как в случае на изображении ниже, установите пакет drbd-dkms.

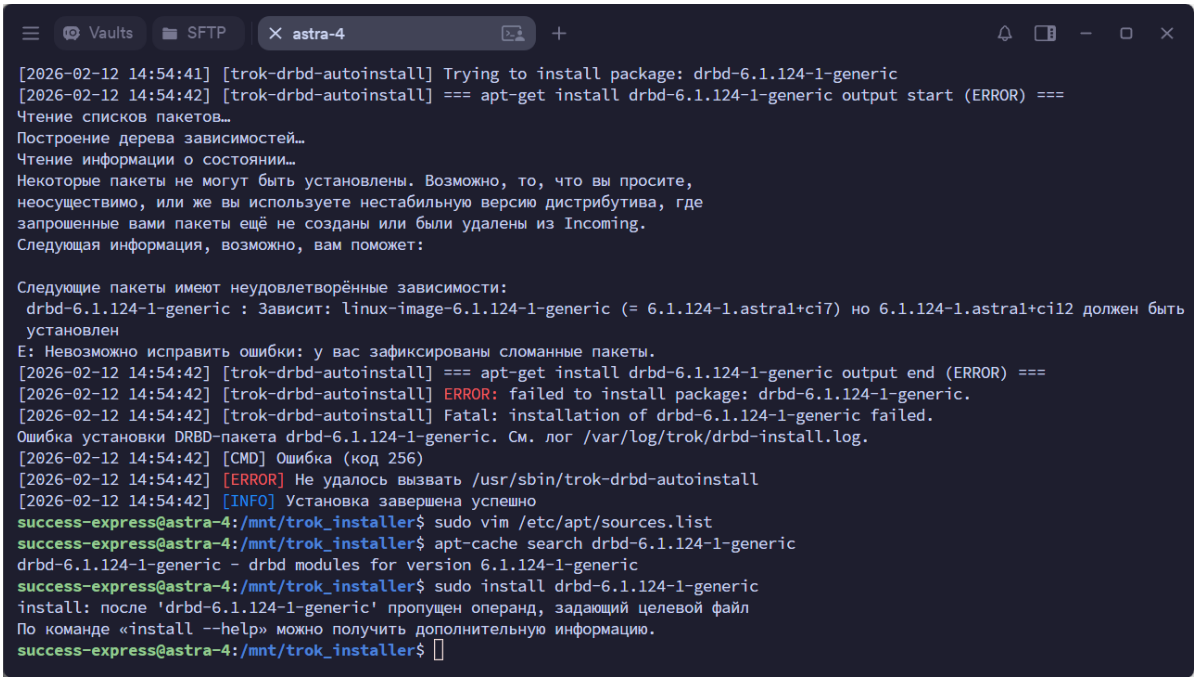



Рисунок 23 – Ошибка совместимости DRBD-пакета с ядром Linux

Пакет drbd-dkms также устанавливается из архива с пакетами TROK после выполнения всех необходимых действий по добавлению локальных пакетов в список репозиториев.

Команда для установки:

```
sudo apt install drbd-dkms
```


Рисунок 24 – Процесс установки пакета drbd-dkms

# 9. ДЕПЛОЙ КЛАСТЕРА БЕЗ HA

Глава описывает развертывание обычного кластера TROK без HA-контроллера для тех, кто знаком с базовой работой Ansible и хочет быстро получить рабочую схему: controller, worker, пул хранения и экспорт по iSCSI, NFS, а также HA Versity S3-шлюз.

## 9.1. Требования к управляющему хосту (узел управления Ansible)

Минимальные требования к ПО:

– Ansible 2.15.0 или выше – инструмент для автоматизации конфигурации и управления инфраструктурой, обеспечивающий выполнение задач на удаленных узлах.

– Python 3 и пакет python3-netaddr – python 3 требуется как базовая среда выполнения для Ansible; пакет python3-netaddr предоставляет дополнительные функции для работы с IP-адресами и сетевыми диапазонами, необходимые для сценариев автоматизации.

– Коллекции Ansible: community.general и ansible.posix – наборы дополнительных модулей и плагинов для Ansible, расширяющие его базовую функциональность. Коллекция community.general содержит множество модулей для работы с различными системами, а ansible.posix предоставляет инструменты для управления POSIX-совместимыми операционными системами (работа с файловыми системами, пользователями, ACL и др.). Коллекции устанавливаются отдельно через ansible-galaxy и подключаются в плейбуках по необходимости.

– SSH-доступ к целевым узлам (controller и worker) обеспечивает безопасное соединение, чтобы Ansible мог выполнять команды удаленно без физического доступа. Требования:

– Заранее создан SSH-ключ на управляющей машине (controller) и установлен на все целевые серверы.

– Заранее создан отдельный пользователь для Ansible (например, ansible) с правами sudo без ввода пароля.

Создать ключ (можно назвать файл по имени сервера/проекта, чтобы не путаться):

```
ssh-keygen -t ed25519 -f ~/.ssh/<имя_ключа> -C "ansible"
```

-t задаёт тип ключа (например, ed25519) эту опцию можно пропустить.

-f задаёт путь/имя файла для сохранения ключа.

-C "ansible" – комментарий, его можно пропустить

Скопировать публичный ключ на целевой узел:

```
ssh-copy-id -i ~/.ssh/<имя_ключа>.pub <имя_пользователя>@<адрес_узла>
```

-i указывает, какой публичный ключ копировать (файл .pub).

Можно использовать комбинацию ключей -fi. -f – «force» (принудительно), пропускает проверки на уже установленные ключи и может привести к дубликатам.

– Заранее определенный набор узлов-экспортёров для iSCSI, NFS и/или S3-шлюза – эти узлы должны быть заранее выделены и описаны в инвентарном файле Ansible в группах iscsi_exporters, nfs_exporters и s3gw_gateways соответственно. Выбор узлов-экспортёров определяется архитектурой кластера и требованиями к отказоустойчивости; на этих узлах в процессе развёртывания будут настроены DRBD-ресурсы, подняты виртуальные IP-адреса и запущены сервисы экспорта.

Для подготовки узла управления к работе с Ansible выполните следующие команды в терминале (команды выполняются с правами суперпользователя (sudo)):

Обновите локальный индекс пакетов системы управления пакетами APT (Advanced Package Tool):

```
sudo apt update
```

Установите Ansible и python3-netaddr:

```
sudo apt install -y ansible python3-netaddr
```

-y автоматически подтверждает установку.

Коллекции community.general и ansible.posix устанавливаются вместе с Ansible по умолчанию. Если требуется обновление или по какой-то причине они отсутствуют, установите/обновите их следующими командами:

```
ansible-galaxy collection install community.general

ansible-galaxy collection install ansible.posix
```

## 9.2. Установка коллекции TROK

Если коллекция astra.trok доступна локально как архив astra-trok-<версия>.tar.gz, произведите ее установку перед деплоем. Коллекция представляет собой набор модулей и ролей Ansible, предназначенный для автоматизации задач развертывания (деплоя) в инфраструктуре.

Выполните команду, которая устанавливает коллекцию astra-trok из локального файла архива astra-trok-<версия>.tar.gz (относительный путь ./ указывает на текущую директорию):

```
ansible-galaxy collection install ./astra-trok-<версия>.tar.gz
```

В случае повторной установки (например, если коллекция была обновлена или требуется перезапись существующей версии) используйте флаг --force для принудительного выполнения операции, игнорируя возможные конфликты:

```
ansible-galaxy collection install ./astra-trok-<версия>.tar.gz --force
```

После установки рекомендуется выполнить проверку, чтобы подтвердить, что коллекция успешно интегрирована в систему и готова к использованию:

```
ansible-galaxy collection list astra.trok
```

При проблемах проверьте вывод Ansible/логи и, при необходимости, выполните команды с sudo.

## 9.3. Подготовка структуры проекта

Для подготовки среды развертывания (деплоя) системы с использованием Ansible рекомендуется создать организованную структуру каталогов и файлов. Структура рабочего каталога trok-deploy/ должна выглядеть следующим образом:

```
trok-deploy/
  ├── inventory.ini
  ├── group_vars/
  |   └── all.yml
  └── site.yml
```

inventory.ini – файл инвентаря, который определяет список целевых хостов (серверов), их группы и параметры подключения (например, IP-адреса, SSH-ключи).

group_vars/all.yaml – файл переменных в формате YAML, расположенный в подкаталоге group_vars/. Он содержит глобальные переменные, доступные всем хостам, такие как версии компонентов или ключи шифрования.

site.yml – основной плейбук Ansible в формате YAML, расположенный в корневом каталоге. Файл описывает последовательность задач развертывания, включая установку ПО, конфигурацию служб и проверку состояния. Он служит точкой входа для запуска всего процесса автоматизации.

## 9.4. Конфигурация файла инвентаря inventory.ini

Файл inventory.ini служит статическим инвентарем в формате INI, где перечислены серверы, разделенные на группы, с указанием необходимых атрибутов для успешного выполнения плейбуков (какие узлы входят в кластер и какие роли они выполняют). Этот файл обеспечивает Ansible информацией о том, как и где выполнять задачи, минимизируя риски ошибок в сетевом взаимодействии и аутентификации.

Пример конфигурации:

```
[controllers]
ctrl1.local.admin ansible_host=192.0.2.11 ansible_user=ansible ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
```

```
   [workers]
   sat1.local.admin  ansible_host=192.0.2.12  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat2.local.admin  ansible_host=192.0.2.13  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [iscsi_exporters]
   sat1.local.admin  ansible_host=192.0.2.12  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat2.local.admin  ansible_host=192.0.2.13  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [nfs_exporters]
   sat1.local.admin  ansible_host=192.0.2.12  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat2.local.admin  ansible_host=192.0.2.13  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [s3gw_gateways]
   sat1.local.admin  ansible_host=192.0.2.12  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat2.local.admin  ansible_host=192.0.2.13  ansible_user=ansible  ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [all:vars]
   ansible_become=true
```

Пояснения и рекомендации:

– ansible_host – IP-адрес, короткое имя хоста или полное доменное имя (FQDN). Используйте FQDN – тогда Ansible будет обращаться к узлам по тем же именам, которые видны внутри самих машин. Это упрощает диагностику и устраняет путаницу.

– ansible_user – имя пользователя для SSH-подключения к серверу. Ansible подключается к узлам именно под этим пользователем. Обычно это выделенный

технический пользователь (например, ansible), созданный специально для автоматизации.

– ansible_ssh_private_key_file – путь к закрытому SSH-ключу на управляющей машине. Используется для аутентификации при подключении. Ключ должен быть заранее скопирован на все серверы (команда ssh-copy-id)..

– На каждом узле должен быть ровно один активный IPv4, и он должен совпадать с адресом в инвентаре (группы controllers или workers).

– В группу iscsi_exporters включаются только те рабочие узлы (worker), между которыми разрешено размещать и переключать iSCSI-ресурс.

– В группу nfs_exporters включаются только те рабочие узлы, между которыми разрешено размещать и переключать NFS-ресурс.

– В группу s3gw_gateways включаются только те рабочие узлы, между которыми разрешено размещать и переключать ресурс S3-шлюза.

## 9.5. Настройка переменных (group_vars/all.yml)

### 9.5.1. Базовая часть

Файл с глобальными переменными Ansible для всех хостов называется group_vars/all.yml. Он размещается в корне проекта Ansible в подкаталоге group_vars, рядом с файлами inventory.ini и site.yml (например, trok-deploy/group_vars/all.yml). Этот каталог и файл не создаются автоматически, их необходимо создать вручную либо получить из шаблона/репозитория.

Сначала создайте каталог group_vars в корне проекта:

```
mkdir -p trok-deploy/group_vars
```

Откройте файл trok-deploy/group_vars/all.yml в текстовом редакторе: если задана переменная окружения $EDITOR – используйте её, иначе запустите nano или vim напрямую:

```
$EDITOR trok-deploy/group_vars/all.yml
# или
```

```
nano trok-deploy/group_vars/all.yml

# или

vim trok-deploy/group_vars/all.yml
```

Пример содержания файла:

```
controllers:
  - 192.0.2.11

workers:
  - 192.0.2.12
  - 192.0.2.13

drbd_replication_network: "192.0.2.0/24"
enable_gateway: true
ufw_enable: false
dist_upgrade: false
cluster_member: true
sp_cluster_member: true
drbd_auto_reboot: true
```

Допустимы расширения .yml или .yaml.

Имя файла обязательно должно начинаться с all (например, all.yml или all.yaml).

Переменные из этого файла применяются ко всем хостам инвентаря, так как они относятся к группе all.

Ключевые переменные и их назначение:

– enable_gateway: true – эта переменная включена явно, поскольку работа iSCSI и NFS требует наличия gateway-компонентов. Без них эти сервисы не смогут функционировать.

– sp_cluster_member: true – настройка активирует регистрацию пулов хранения в кластере. Это необходимо для того, чтобы узлы могли взаимодействовать с общими хранилищами.

### 9.5.2. Опционально: глобальная настройка DRBD на worker-узлах

Включайте настройку только в том случае, если узел с ролью worker должен управлять конфигурационным файлом DRBD /etc/drbd.d/global_common.conf.

Если узлу не требуется управлять DRBD-настройками глобального уровня, переменную следует оставить выключенной или не указывать.

```
worker_drbd_tuning_enabled: true
worker_drbd_tuning_profile: "25g-ssd"
worker_drbd_tuning_resources_per_node: 20
worker_drbd_tuning_reboot: false
```

Настройка профиля производительности DRBD автоматически рассчитывает оптимальные параметры DRBD на основе двух факторов:

– Выбранного профиля (скорость сети и тип дисков).

– Количества ресурсов, которые будут работать на узле.

Поддерживаемые значения worker_drbd_tuning_profile:

Таблица 5 – Доступные профили настройки DRBD

| Тип конфигурации | Доступные профили |
|---|---|
| Базовый (по умолчанию) | default (соответствует 10g-hdd) |
| Сеть 1 Гбит/с или 10 Гбит/с | 1g-hdd, 10g-hdd – для HDD<br>1g-ssd, 10g-ssd – для SSD<br>1g-nvme, 10g-nvme – для NVMe<br>1g-nvme-aggressive, 10g-nvme-aggressive – агрессивный режим для NVMe |
| Сеть 25 Гбит/с или 40 Гбит/с | 25g-hdd, 40g-hdd – для HDD<br>25g-ssd, 40g-ssd – для SSD<br>25g-nvme, 40g-nvme – для NVMe<br>25g-nvme-aggressive, 40g-nvme-aggressive – агрессивный режим для NVMe |
| Сеть 100 Гбит/с | 100g-hdd – для HDD<br>100g-ssd – для SSD<br>100g-nvme – для NVMe |

| | |
|---|---|
| | 100g-nvme-aggressive – агрессивный режим для NVMe |
| Сеть 200 Гбит/с | 200g-hdd – для HDD<br>200g-ssd – для SSD<br>200g-nvme – для NVMe<br>200g-nvme-aggressive – агрессивный режим для NVMe |

Примечание: Профили с пометкой -aggressive используют более высокие настройки производительности, что даёт прирост скорости, но требует более стабильной сетевой инфраструктуры. Агрессивные профили нельзя использовать, если на узле запланировано 50 и более ресурсов – такие настройки создадут избыточную нагрузку и могут дестабилизировать работу системы.

Настройка количества DRBD-ресурсов на узле worker_drbd_tuning_resources_per_node задаёт, сколько DRBD-ресурсов будет работать на одном узле. От этого зависят параметры производительности – чем больше ресурсов, тем ниже пропускная способность у каждого из них.

Доступные значения:

Таблица 6 – Доступные значения количества ресурсов на узел

| Значение | Рекомендации по применению |
|---|---|
| 1, 10, 20 | Низкая плотность – на узле мало ресурсов, каждый из них может работать с максимальной производительностью. |
| 50, 100 | Средняя плотность – ресурсов уже достаточно много, поэтому производительность каждого умеренно ограничивается в пользу общей стабильности. |
| 200, 500 | Высокая плотность – на узле очень много ресурсов, приоритет отдаётся общей |

| | |
|---|---|
| | устойчивости системы, а не скорости отдельного ресурса. |
| unlimited | Особый режим – подходит для большого количества небольших ресурсов, где каждый потребляет минимум ресурсов системы. |

Общий принцип: чем больше ресурсов на узле, тем более «консервативными» становятся настройки DRBD для каждого из них, чтобы избежать перегрузки системы.

Таблица 7 – Примеры комбинаций

| Сценарий | Профиль | Ресурсов на узел |
|---|---|---|
| Консервативный базовый вариант | 10g-hdd | 20 |
| Обычный быстрый узел | 25g-ssd | 20 |
| Быстрый, но уже плотный узел | 100g-nvme | 100 |
| Сценарий с множеством ресурсов | 200g-nvme | 500 |

### 9.5.3. Пул хранения

Для хранения данных используется LVM-пул. В типичном рабочем кластере (без требований высокой доступности) это хороший вариант по умолчанию, и на практике он часто работает быстрее других решений.

Настройки пулов хранения задаются в глобальном файле переменных: trok-deploy/group_vars/all.yml в корне проекта Ansible.

Пример настройки обычного LVM-пула:

```
storage_pool_lvm_pools:
  - name: lvm_pool
    vg_name: my_vg
    disks:
      - /dev/vdb
```

Обязательные параметры:

name – имя пула

vg_name – имя группы томов (Volume Group)

disks – список дисков, которые будут использоваться в пуле (каждый путь должен быть вида /dev/...)

Таблица 8 – Поддерживаемые типы пулов

| Тип хранилища | Переменная для настройки |
|---|---|
| Обычный LVM (рекомендуемый) | storage_pool_lvm_pools |
| LVM с тонким выделением (Thin Provisioning) | storage_pool_lvm_thin_pools |
| ZFS | storage_pool_zfs_pools |

Важно. В плейбуке следует запускать только одну роль подготовки пула – ту, которая соответствует выбранному типу хранилища. Не смешивайте настройки разных типов в одном запуске.

### 9.5.4. iSCSI-экспорт

Настройки iSCSI-целей задаются в глобальном файле переменных: trok-deploy/group_vars/all.yml в корне проекта Ansible.

Пример настройки iSCSI-цели:

```
iscsi_manage_resources: true

iscsi_targets:
  - resource_name: ISCSI-APP
    storage_pool: lvm_pool
    bind_ip: "192.0.2.249"
    bind_port: 3260
    bind_cidr_netmask: 24
    iqn: "iqn.2026-05.trok.sds:iscsi-app"
    username: "chap-user"
    password: "chap-password"
    allowed_initiators: []
    target_ra:
      tid: 10
    lun_ra_defaults:
      vendor_id: "TROK"
      product_id: "TROK_LUN"
      lio_iblock: 0
    volumes:
      - size: "64MiB"
      - size: "96MiB"
```

Таблица 9 – Обязательные параметры

| Параметр | Обязательность |
|---|---|
| resource_name | Всегда |
| bind_ip | Всегда |
| iqn | Всегда |
| volumes | Всегда |
| storage_pool | Если iscsi_manage_resources: true |
| volumes[].size | Если iscsi_manage_resources: true |
| username и password | Задаются только парой (оба или ни один) |

Важные ограничения и правила:

– allowed_initiators: [] – если оставить список пустым, то любой инициатор сможет подключиться к iSCSI-цели (ограничений по IQN не будет). Чтобы разрешить доступ только определённым инициаторам, укажите их IQN-имена в списке, например:

```
      allowed_initiators:      ["iqn.1994-05.com.redhat:client1",      "iqn.1994-
05.com.redhat:client2"]
```

– Группа iscsi_exporters должна содержать только те узлы, на которых разрешено размещать iSCSI-ресурс. Если группа слишком широкая, ресурс может быть размещён на лишних узлах.

– Роль необходимо запускать строго на группе узлов iscsi_exporters, например:

```
  - hosts: iscsi_exporters
    roles:
      - your_iscsi_role
```

Не используйте для запуска этой роли группы workers или all – это может привести к тому, что роль будет применена на узлах, где iSCSI-экспорт не должен настраиваться.

– Если параметр autoplace_nodes явно указывает узлы за пределами группы iscsi_exporters, роль завершится ошибкой – неявного расширения размещения не происходит.

Таблица 10 – Практические рекомендации

| Сценарий | Рекомендация |
|---|---|
| Небольшие и обычные ресурсы | Используйте iscsi_manage_resources: true – этого достаточно. |
| Очень большие ресурсы | Первичная синхронизация может занять много времени. Роль в этом режиме синхронно ждёт завершения синхронизации, что может привести к таймаутам. |

Для больших ресурсов лучше использовать двухэтапную схему:

– Сначала создайте ресурс и дождитесь завершения его синхронизации отдельно (вне роли).

– Затем запускайте настройку iscsi_target с параметром iscsi_manage_resources: false.

Этот подход помогает избежать: таймаута синхронизации (sync_timeout), внешних таймаутов Ansible, обрывов SSH-соединения, таймаутов в CI-заданиях.

### 9.5.5. NFS-экспорт

Настройки NFS-экспортов задаются в глобальном файле переменных: trok-deploy/group_vars/all.yml в корне проекта Ansible.

Пример настройки NFS-экспорта:

```
nfs_manage_resources: true

nfs_exports:
  - resource_name: NFS-APP
    storage_pool: lvm_pool
    size: "128MiB"
    bind_ip: "192.0.2.250"
    bind_cidr_netmask: 24
    mount_point: "/srv/nfs/nfs-app"
```

```
      fsid: 200
      clientspec: "192.0.2.0/24"
      export_options: "rw,sync,no_subtree_check,no_root_squash"
```

Таблица 11 – Обязательные параметры

| Параметр | Обязательность |
|---|---|
| resource_name | Всегда |
| bind_ip | Всегда |
| mount_point | Всегда |
| fsid | Всегда |
| clientspec | Всегда |
| storage_pool | Если nfs_manage_resources: true |
| size | Если nfs_manage_resources: true |

Таблица 12 – Особенности управления NFS-сервисом

| Количество экспортов | Способ управления |
|---|---|
| Один экспорт | Роль управляет NFS-сервером через drbd-reactor (полный цикл: монтирование → IP → запуск NFS) |
| Несколько экспортов | NFS-сервер работает как обычный systemd-сервис на узлах-экспортёрах. drbd-reactor управляет только монтированием ФС, IP-адресом и экспортом (exportfs), но не перезапускает сам NFS-сервер. |

Важные ограничения и правила:

– Группа nfs_exporters должна содержать только те узлы, на которых разрешено размещать NFS-ресурс. Если группа слишком широкая, ресурс может быть размещён на лишних узлах.

– Запускайте роль только через hosts: nfs_exporters, а не через более широкую группу (например, workers или all).

– Если параметр autoplace_nodes явно указывает узлы за пределами группы nfs_exporters, роль завершится ошибкой — неявного расширения размещения не происходит.

Таблица 13 – Практические рекомендации

| Количество экспортов | Способ управления |
|---|---|
| Небольшие и обычные ресурсы | Используйте nfs_manage_resources: true – этого достаточно. |
| Повторное использование существующего DRBD-ресурса | На ресурсе должна уже быть создана файловая система. Если её нет — роль завершится ошибкой (форматирование не выполняется автоматически). |
| Очень большие ресурсы | Первичная синхронизация может занять много времени. Роль в этом режиме синхронно ждёт завершения синхронизации, что может привести к таймаутам. |

Для больших ресурсов лучше использовать двухэтапную схему:

– Сначала создайте DRBD-ресурс и дождитесь завершения его синхронизации отдельно (вне роли).

– Затем запускайте настройку NFS-экспорта с параметром nfs_manage_resources: false.

Этот подход помогает избежать: таймаута синхронизации (sync_timeout), внешних таймаутов Ansible, обрывов SSH-соединения, таймаутов в CI-заданиях.

### 9.5.6. Versity S3-шлюз

Настройки S3-шлюза задаются в глобальном файле переменных: trok-deploy/group_vars/all.yml в корне проекта Ansible.

Пример настройки S3-шлюза:

```
trok_s3_gw_manage_resources: true
trok_s3_gw_instance_name: dev
trok_s3_gw_vip: "192.0.2.251"
trok_s3_gw_vip_cidr_netmask: 24
```

```
trok_s3_gw_storage_pool: lvm_pool
trok_s3_gw_resource_size: "20GiB"
trok_s3_gw_rg_place_count: 2
trok_s3_gw_port_s3: 7070
trok_s3_gw_port_admin: 7080
trok_s3_gw_port_webui: 7071
trok_s3_gw_region: us-east-1
trok_s3_gw_root_access_key_id: "admin"
trok_s3_gw_root_secret_access_key: "{{ vault_s3gw_secret }}"
```

Таблица 14 – Обязательные параметры

| Параметр | Обязательность |
|---|---|
| trok_s3_gw_vip | Всегда (должен быть валидным IPv4-адресом) |
| trok_s3_gw_storage_pool | Если trok_s3_gw_manage_resources: true |
| trok_s3_gw_resource_size | Если trok_s3_gw_manage_resources: true |

S3-шлюз работает как высокодоступный сервис на одном DRBD-ресурсе:

– В каждый момент времени ровно один узел из группы s3gw_gateways держит роль Primary.

– На этом узле автоматически:

  – монтируется ФС,

  – поднимается виртуальный VIP,

  – запускается сервис trok-s3-gw@<instance>.

– Переключение на другой узел выполняется автоматически через drbd-reactor.

Пакет trok-s3-gw устанавливается из APT-репозитория TROK через зависимость pre_install.

Переопределяйте trok_s3_gw_package_name только в том случае, если имя пакета в вашем репозитории отличается от стандартного.

Важные ограничения и правила:

– Количество узлов в группе s3gw_gateways должно быть не меньше значения trok_s3_gw_rg_place_count (иначе негде будет размещать ресурс).

– Запускайте роль только через hosts: s3gw_gateways, а не через более широкую группу (например, workers или all).

Безопасность учётных данных:

– Никогда не храните секретный ключ trok_s3_gw_root_secret_access_key в открытом виде в all.yml.

– Используйте Ansible Vault для шифрования чувствительных данных.

– Значения по умолчанию admin/admin пригодны только для лабораторного стенда – никогда не используйте их в продакшене.

Таблица 15 – Практические рекомендации по управлению ресурсами

| Сценарий | Поведение роли |
|---|---|
| Создание нового ресурса | Роль создаёт DRBD-ресурс, форматирует ФС и настраивает шлюз. |
| Повторный запуск на существующем ресурсе | Роль работает в режиме validate-only (проверяет, но ничего не меняет), если на ресурсе уже есть живой Primary. |
| Очень большие ресурсы | Первичная синхронизация может занять много времени. Используйте двухэтапную схему. |

Для больших ресурсов рекомендуется двухэтапный подход:

– Сначала создайте DRBD-ресурс и дождитесь завершения его синхронизации отдельно (вне роли).

– Затем запускайте настройку S3-шлюза с параметром trok_s3_gw_manage_resources: false.

Это помогает избежать таймаутов синхронизации (sync_timeout), обрывов SSH-соединения и проблем с CI-заданиями.

### 9.6. Плейбук

Плейбук находится в корне проекта Ansible: trok-deploy/site.yml Это главный файл, который запускает все роли для развёртывания кластера. Если файл отсутствует – создайте его вручную.

Полный пример site.yml:

```
---
- name: Развернуть контроллеры
  hosts: controllers
  gather_facts: true
  become: true
  become_user: root
  roles:
    - { role: astra.trok.controller }

- name: Развернуть рабочие узлы
  hosts: workers
  gather_facts: true
  become: true
  become_user: root
  roles:
    - { role: astra.trok.worker }

- name: Подготовить LVM-пул хранения
  hosts: workers
  gather_facts: true
  become: true
  become_user: root
  roles:
    - { role: astra.trok.storage_pool_lvm }

- name: Настроить iSCSI-экспорт
  hosts: iscsi_exporters
  gather_facts: true
  become: true
  become_user: root
  roles:
    - { role: astra.trok.iscsi_target }

- name: Настроить NFS-экспорт
  hosts: nfs_exporters
  gather_facts: true
  become: true
  become_user: root
  roles:
    - { role: astra.trok.nfs_export }

- name: Развернуть Versity S3-шлюз
  hosts: s3gw_gateways
  gather_facts: true
  become: true
  become_user: root
  roles:
    - { role: astra.trok.s3gw }
```

Таблица 16 – Замечания

| Правило | Пояснение |
|---|---|

| | |
|---|---|
| Порядок ролей важен | Сначала разворачиваются контроллеры и рабочие узлы, затем создаётся пул хранения, и только после этого настраиваются экспорты (iSCSI, NFS, S3). |
| Запуск на конкретных группах | Каждая роль запускается только на своей группе хостов (например, iscsi_target – только на iscsi_exporters). Это предотвращает случайное применение ролей на неподходящих узлах. |

Если используется другая подсистема хранения:

Если вы не используете LVM, а выбираете другое хранилище (например, LVM Thin или ZFS):

- Замените роль astra.trok.storage_pool_lvm на соответствующую:
  - astra.trok.storage_pool_lvm_thin – для LVM с тонким выделением
  - astra.trok.storage_pool_zfs – для ZFS
- Обновите имена пулов в настройках экспортов:
  - В iscsi_targets укажите storage_pool, соответствующее созданному пулу.
  - В nfs_exports укажите storage_pool, соответствующее созданному пулу.

Пример: если вы создали ZFS-пул с именем zfs_pool, то в настройках iSCSI и NFS должно быть:

```
storage_pool: zfs_pool
```

## 9.7. Запуск деплоя

Убедитесь, что вы находитесь в корневой папке проекта Ansible – там, где расположены файлы:

- inventory.ini — файл с хостами,
- site.yml — главный плейбук,
- папка group_vars/ с глобальными настройками,
- папка roles/ с ролями.

Если вы ещё не перешли в эту папку – сделайте это:

```
cd /путь/к/вашему/project/trok-deploy
```

Все команды ниже выполняются именно из этой директории.

Если вы хотите развернуть всё сразу, выполните:

```
ansible-playbook -i inventory.ini site.yml
```

Если вам нужно выполнить развёртывание по шагам (например, для проверки или поэтапного ввода), используйте флаг --limit с указанием группы хостов:

Разверните контроллеры:

```
ansible-playbook -i inventory.ini site.yml --limit controllers
```

Разверните рабочие узлы:

```
ansible-playbook -i inventory.ini site.yml --limit workers
```

Настройте iSCSI-экспорт:

```
ansible-playbook -i inventory.ini site.yml --limit iscsi_exporters
```

Настройте NFS-экспорт:

```
ansible-playbook -i inventory.ini site.yml --limit nfs_exporters
```

Разверните S3-шлюз:

```
ansible-playbook -i inventory.ini site.yml --limit s3gw_gateways
```

Таблица 17 – Важные нюансы

| Что нужно знать | Пояснение |
|---|---|
| Порядок запуска обязателен | Сначала контроллеры, потом рабочие узлы, затем пулы хранения, и только после этого — экспорты (iSCSI, NFS, S3). Нарушение порядка приведёт к ошибкам. |

| | |
|---|---|
| Где выполнять команды | Все команды выполняются на вашей управляющей машине (где установлен Ansible), не на узлах кластера. |
| Рабочая директория | Обязательно находитесь в папке проекта (trok-deploy), иначе Ansible не найдёт файлы инвентаря и плейбука. |
| Флаг --limit | Он ограничивает выполнение только указанной группой узлов. Это удобно, чтобы не запускать всё подряд и не тратить время на лишние этапы. |
| Если что-то пошло не так | Ansible покажет ошибку с указанием проблемы. Чаще всего это: неправильный путь к файлам, недоступные узлы, недостаток прав (запускайте с sudo или от root). |
| Права на выполнение | В плейбуке указано become: true и become_user: root, поэтому Ansible будет повышать привилегии автоматически. Убедитесь, что на узлах настроен sudo без пароля для пользователя, от которого вы подключаетесь. |
| Файл инвентаря | В примерах используется inventory.ini — если ваш файл называется иначе (например, hosts.ini), подставьте своё имя. |

## 9.8. Проверка после деплоя

### 9.8.1. Базовые сервисы

Выполните на каждом контроллере:

```
sudo systemctl status trok-controller

sudo systemctl status trok-cp-endpoint
```

```
sudo systemctl status trok-auth
```

На рабочих узлах (узлах с ролью worker)

Выполните на каждом рабочем узле:

```
sudo systemctl status trok-worker
```

Если сервис не запустился:

Посмотрите логи:

```
sudo journalctl -u trok-worker -n 50 --no-pager
```

Попробуйте перезапустить сервис:

```
sudo systemctl restart trok-worker
```

Если перезапуск не помог – проверьте:

– наличие конфигурационных файлов,

– доступность дисков и сетевых путей,

– наличие ошибок в логах (journalctl -xe).

### 9.8.2. Пул хранения

После настройки хранилища необходимо убедиться, что пул создан корректно и доступен для использования. Откройте веб-интерфейс TROK и выполните следующие проверки:

– Узлы кластера – убедитесь, что в интерфейсе отображаются все рабочие узлы (workers), которые вы указали в инвентаре. Они должны быть в статусе Online или Active.

– Пул хранения – проверьте, что созданный вами пул (lvm_pool или другой) присутствует в списке пулов хранения.

– Привязка пула к узлам – убедитесь, что созданный пул привязан именно к тем узлам, где вы планируете его использовать (обычно это все узлы из группы workers или их подмножество).

Если сущности не отображаются:

Таблица 18 – Поиск проблем

| Проблема | Возможное решение |
|---|---|
| Узел не отображается или в статусе Offline | Проверьте сеть, запущен ли trok-worker на узле, доступен ли SSH. |
| Пул отсутствует в списке | Убедитесь, что роль storage_pool_lvm (или другая) была успешно выполнена. Проверьте логи Ansible на наличие ошибок. |
| Пул не привязан к ожидаемым узлам | Возможно, вы указали неправильные узлы в группе workers или параметрах пула. Проверьте переменные в all.yml. |

### 9.8.3. iSCSI

На каждом узле-экспортёре (где работает iSCSI-цель) выполните следующие команды.

Установите iSCSI-инициатор (пакет open-iscsi), если он еще не установлен:

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

Проверьте роль DRBD-ресурса (Primary/Secondary):

```
sudo drbdadm role ISCSI-APP
```

Проверьте статус drbd-reactor (управление сервисами):

```
sudo drbd-reactorctl status
```

Посмотрите конфигурацию iSCSI-целей (LIO)

```
sudo targetcli ls
```

Что должно отображаться:

– drbdadm role – ровно один узел в кластере должен показывать Primary, остальные — Secondary.

– drbd-reactorctl status – сервисы (Filesystem, IPaddr2 и др.) должны быть в статусе Started на Primary-узле и Stopped на Secondary.

– targetcli ls – должна отображаться iSCSI-цель с IQN, указанным в настройках, и LUN-ы с нужным размером.

Выполните на клиентской машине, которая будет использовать iSCSI-диск (узел-инициатор или клиент, который подключается к iSCSI) следующие команды. Обнаружьте цель на экспортёре:

```
sudo iscsiadm -m discovery -t sendtargets -p <ip_цели>:3260
```

Настройте аутентификацию CHAP (если используется):

```
sudo iscsiadm -m node -T iqn.2026-05.trok.sds:iscsi-app -p <ip_цели>:3260 -o update -n node.session.auth.authmethod -v CHAP

sudo iscsiadm -m node -T iqn.2026-05.trok.sds:iscsi-app -p <ip_цели>:3260 -o update -n node.session.auth.username -v chap-user

sudo iscsiadm -m node -T iqn.2026-05.trok.sds:iscsi-app -p <ip_цели>:3260 -o update -n node.session.auth.password -v chap-password
```

Подключитесь к цели:

```
sudo iscsiadm -m node -T iqn.2026-05.trok.sds:iscsi-app -p <ip_цели>:3260 --login
```

Ожидаемый результат:

После --login подключение должно быть успешным.

Проверьте, что на инициаторе появился новый диск:

```
lsblk
```

Должен отобразиться новый блочный диск (например, /dev/sdb или /dev/sdc).

Таблица 19 – Ожидаемое состояние системы

| Компонент | Ожидание |
|---|---|
| DRBD-ресурс | Ровно один узел-экспортёр в статусе Primary, остальные – Secondary |
| iSCSI-цель | Доступна по IP-адресу, который вы указали в bind_ip) на порту 3260 |

| | |
|---|---|
| Подключение инициатора | Инициатор успешно обнаруживает цель и подключается по указанному IQN |
| Аутентификация | Если используется CHAP — подключение проходит только с правильными логином и паролем |

Таблица 20 – Поиск проблем

| Проблема | Проверьте |
|---|---|
| Нет Primary на экспортёре | Проверьте DRBD-статус: sudo drbdadm status |
| Цель не видна по IP | Проверьте, поднят ли плавающий VIP на экспортёре: ip a |
| Инициатор не подключается | Убедитесь, что CHAP-учётные данные совпадают, и порт 3260 открыт в файерволе |
| targetcli ls пуст | Проверьте, была ли создана цель: возможно, роль iscsi_target не завершилась успешно, посмотрите логи Ansible |

### 9.8.4. NFS

Выполните на каждом узле из группы nfs_exporters следующие команды/

Установите NFS-клиент (пакет nfs-common), если он еще не установлен:

```
sudo apt update
sudo apt install -y nfs-common
```

Проверьте роль DRBD-ресурса (Primary/Secondary):

```
sudo drbdadm role NFS-APP
```

Проверьте статус drbd-reactor (управление сервисами):

```
sudo drbd-reactorctl status
```

Посмотрите активные NFS-экспорты:

```
sudo exportfs -v
```

Что должно отображаться:

– drbdadm role – ровно один узел в кластере должен показывать Primary, остальные – Secondary.

– drbd-reactorctl status – сервисы (Filesystem, IPaddr2, exportfs и др.) должны быть в статусе Started на Primary-узле и Stopped на Secondary.

– exportfs -v – на Primary узле должен отображаться экспортируемый каталог (/srv/nfs/nfs-app), доступный для клиентов с указанными параметрами (например, rw,sync,no_subtree_check,no_root_squash).

Выполните на клиентской машине, которая будет использовать NFS-шару следующие команды:

Создайте точку монтирования (если её нет):

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

Примонтируйте NFS-шару с версией протокола 3:

```
sudo mount -t nfs -o vers=3 192.0.2.250:/srv/nfs/nfs-app /mnt
```

Проверьте возможность записи – создайте тестовый файл:

```
sudo touch /mnt/probe.txt
```

Отмонтируйте шару:

```
sudo umount /mnt
```

Ожидаемый результат:

– Монтирование должно пройти без ошибок.

– Файл /mnt/probe.txt должен создаться без ошибок.

– Отмонтирование должно пройти без ошибок.

Таблица 21 – Ожидаемое состояние системы

| Компонент | Ожидание |
|---|---|

| | |
|---|---|
| DRBD-ресурс | Ровно один узел-экспортёр в статусе Primary, остальные – Secondary. |
| NFS-экспорт | Экспорт доступен по IP-адресу, который вы указали в bind_ip. |
| Подключение клиента | Монтирование и запись проходят успешно. |
| Повторное монтирование | После переключения роли экспорт поднимается автоматически, и клиент может подключиться снова без ручных действий. |

Таблица 22 – Поиск проблем

| Проблема | Проверьте |
|---|---|
| Нет Primary на экспортёре | Проверьте DRBD-статус: sudo drbdadm status NFS-APP. |
| Экспорт не отображается (exportfs -v пуст). | Проверьте статус drbd-reactorctl status – возможно, сервис exportfs не запустился. |
| Клиент не может примонтировать | Проверьте, поднят ли плавающий VIP на экспортёре: ip a; убедитесь, что порт NFS (2049) открыт в файерволе. |
| Нет прав на запись | Проверьте права на каталог /srv/nfs/nfs-app на экспортёре (должны быть доступны для записи). |
| После переключения экспорт не доступен | Проверьте, что drbd-reactor корректно управляет сервисами: логи sudo journalctl -u drbd-reactor. |

### 9.8.5. Versity S3-шлюз

Выполните на каждом узле, входящем в группу s3gw_gateways, команды:

Проверьте роль DRBD-ресурса (Primary/Secondary):

```
sudo drbdadm role trok-s3-gw-root
```

Проверьте статус управления сервисами через drbd-reactor:

```
sudo drbd-reactorctl status trok-s3-gw
```

Проверьте, активен ли сервис S3-шлюза:

```
sudo systemctl is-active trok-s3-gw@dev
```

Проверьте, смонтирована ли файловая система шлюза:

```
sudo findmnt --mountpoint /srv/trok-s3-gw
```

Что должно отображаться:

– drbdadm role – ровно один узел в кластере должен показывать Primary, остальные – Secondary.

– drbd-reactorctl status – сервисы (Filesystem, IPaddr2, trok-s3-gw@dev и др.) должны быть в статусе Started на Primary-узле и Stopped на Secondary.

– systemctl is-active – на Primary-узле должен вернуть active, на Secondary – inactive.

– findmnt – на Primary-узле должна отображаться точка монтирования /srv/trok-s3-gw (с указанием устройства DRBD). На Secondary монтирования быть не должно.

Для проверки доступности сервиса, с любого хоста, который имеет сетевой доступ к плавающему VIP-адресу (включая сами gateway-узлы), выполните:

```
sudo findmnt --mountpoint /srv/trok-s3-gw
```

# 10. СЦЕНАРИЙ ДЛЯ ДЕПЛОЯ НА КЛАСТЕРА

Эта глава описывает переход от обычного кластера TROK к высокодоступной (HA) схеме с использованием роли ha_controller для тех, кто уже знаком с базовым развертыванием TROK и хочет обеспечить отказоустойчивость управляющего слоя.

Что нужно иметь перед переходом к HA:

– Базовый кластер TROK (controller, worker) – развернутый и работающий.

– Все будущие HA-узлы уже входят в обычный кластер TROK.

– Готовый пул хранения (LVM, ZFS или другой) – созданный и доступный на узлах.

– Значение ha_pool_name известно и совпадает с именем этого пула.

– Virtual-IP-адрес заранее выделен и доступен в нужной сети.

– Этап настройки iSCSI можно пропустить, если он не требуется в вашей схеме.

Рекомендуется не запускать HA первым шагом. Сначала разверните обычный кластер TROK. Затем создайте пул хранения. И только после этого запускайте роль astra.trok.ha_controller.

Что получится в итоге:

– База данных TROK controller будет вынесена в отдельный HA-ресурс на DRBD.

– На узлах, отведённых под HA, будет настроен drbd-reactor для автоматического переключения.

– Клиенты и сервисы смогут обращаться к контроллеру по плавающему VIP-адресу (ha_controller_vip), а не к фиксированному узлу.

## 10.1. Общая информация о ha_controllers

Группа ha_controllers – это группа инвентаря, в которой перечислены точные узлы, участвующие в HA-кластере. Именно по этому списку роль определяет, где можно размещать HA-ресурс базы данных.

Важно. Не добавляйте в ha_controllers лишние узлы. Это может привести к тому, что HA-ресурс будет размещён не там, где вы ожидаете.

Первый IP-адрес в списке controllers (controllers[0]) считается основным для подготовки HA. На этом узле выполняются ключевые шаги начальной настройки.

Важно. Если первым указать не тот узел, настройка HA будет выполнена на неверном узле, и кластер может работать некорректно.

Роль ha_controller не создаёт пул хранения самостоятельно. Она ожидает, что нужный пул уже существует (например, после выполнения роли astra.trok.storage_pool_lvm_thin).

Важно. ha_pool_name должен совпадать с уже созданным пулом. Если ha_pool_name указывает на несуществующий пул, создание HA-ресурса завершится ошибкой.

ha_controller_vip – это виртуальный IP-адрес, который всегда находится на активном HA-узле. Клиенты и сервисы должны подключаться именно к этому адресу, а не к конкретному узлу.

Важно. Если VIP не задан, конфликтует с другими адресами или неправильно маршрутизируется в вашей сети, доступность контроллера для клиентов будет нарушена.

## 10.2. Конфигурация файла инвентаря inventory.ini

Пример конфигурации:

```
   [controllers]
   cmbnd     ansible_host=192.0.2.11     ansible_user=ansible     ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [workers]
   cmbnd     ansible_host=192.0.2.11     ansible_user=ansible     ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat1      ansible_host=192.0.2.12      ansible_user=ansible     ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
```

```
   sat2      ansible_host=192.0.2.13      ansible_user=ansible      ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [ha_controllers]
   cmbnd      ansible_host=192.0.2.11      ansible_user=ansible      ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat1      ansible_host=192.0.2.12      ansible_user=ansible      ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"
   sat2      ansible_host=192.0.2.13      ansible_user=ansible      ansible_port=22
ansible_connection=ssh              ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args="-o              StrictHostKeyChecking=no              -o
UserKnownHostsFile=/dev/null"

   [all:vars]
   ansible_become=true
```

## 10.3. Настройка переменных (group_vars/all.yml)

### 10.3.1. База для HA

Пример содержания файла:

```
   controllers:
     - 192.0.2.11

   workers:
     - 192.0.2.11
     - 192.0.2.12
     - 192.0.2.13

   drbd_replication_network: "192.0.2.0/24"
   enable_gateway: true
   ufw_enable: false
   dist_upgrade: false
   cluster_member: true
   sp_cluster_member: true
   drbd_auto_reboot: true
```

### 10.3.2. Пул хранения для HA

Пример содержания файла:

```
   storage_pool_lvm_thin_pools:
     - pool_name: thinpool
       vg_name: drbdpool
       disks:
```

```
        - /dev/vdb
      size: "8.5G"
```

### 10.3.3. HA-переменные

```
ha_db_resource_group: "HA-TROK-DB-GRP"
ha_db_resource_name: "HA-TROK-DB"
ha_db_size: "500M"
ha_db_sync_timeout: 1800
ha_pool_name: "thinpool"
ha_controller_vip: "192.0.2.248"
ha_controller_vip_netmask: "24"
```

### 10.3.4. Опциональная глобальная настройка DRBD на HA worker-узлах

Эта настройка по-прежнему относится к роли worker и применяется на этапе развёртывания worker-узлов до того, как ha_controller настраивает HA-ресурс.

```
worker_drbd_tuning_enabled: true
worker_drbd_tuning_profile: "25g-ssd"
worker_drbd_tuning_resources_per_node: 100
worker_drbd_tuning_reboot: false
```

Правила выбора параметров настройки DRBD:

worker_drbd_tuning_profile – определяет, какая комбинация сетевого оборудования и типа дисков используется на узле (например, 10g-hdd или 25g-ssd). Это влияет на общую производительность DRBD-репликации.

worker_drb d_tuning_resources_per_node – ограничивает производительность каждого отдельного DRBD-ресурса на узлах с высокой плотностью (где одновременно работает много ресурсов). Чем больше ресурсов, тем более «консервативными» становятся настройки для каждого из них, чтобы избежать перегрузки системы.

## 10.4. Плейбук site.yml

Пример содержания файла site.yml:

```
- name: Развернуть controller
  hosts: controllers
  gather_facts: true
  become: true
  become_user: root
```

```
    roles:
      - { role: astra.trok.controller }

  - name: Развернуть workers
    hosts: workers
    gather_facts: true
    become: true
    become_user: root
    roles:
      - { role: astra.trok.worker }

  - name: Подготовить LVM Thin pool
    hosts: workers
    gather_facts: true
    become: true
    become_user: root
    roles:
      - { role: astra.trok.storage_pool_lvm_thin }

  - name: Настроить HA controller
    hosts: ha_controllers
    gather_facts: true
    become: true
    become_user: root
    roles:
      - { role: astra.trok.ha_controller }
```

Если HA-ресурс должен использовать другую подсистему хранения:

– замените роль подготовки пула на нужную;

– укажите в ha_pool_name имя созданного пула.

Запуск:

```
ansible-playbook -i inventory.ini site.yml
```

Повторный запуск:

– Повторный запуск site.yml допустим после успешного развёртывания HA.

– Это корректно работает даже после переключения на резервный узел. Активный узел может отличаться от controllers[0] – это штатная ситуация.

Важное ограничение:

– Если первый запуск HA был прерван (оборвался на полпути), обычный повторный запуск не исправит ситуацию.

– Для восстановления используйте отдельный сценарий восстановления, а не повторный запуск site.yml.

## 10.5. Восстановление после оборванного первого первичного запуска HA

### 10.5.1. Граница обычного повторного запуска

Повторный запуск site.yml безопасен только после того, как первый запуск HA завершился успешно.

Исключение: если первый запуск HA был прерван после создания маркерных файлов, роль ha_controller при повторном запуске обнаружит уже существующие маркеры и пропустит этап переноса базы данных (move_database.yaml).

Маркерные файлы:

- /etc/drbd-reactor.d/<ha_db_resource_name>.toml
- /etc/systemd/system/trok-controller.service.d/10-ha-db-mount.conf
- /etc/systemd/system/trok-auth.service.d/10-ha-db-mount.conf

Пока эти файлы существуют, controller считает, что узел уже управляется через HA, и не перезапускает локальные сервисы: trok-controller, trok-auth, trok-cp-endpoint.

### 10.5.2. Карта стадий для первого первичного запуска HA

В процессе развёртывания HA-контроллера система может находиться в одном из нескольких состояний. В таблице ниже перечислены признаки каждого состояния и допустимые действия для восстановления нормальной работы.

Таблица 23 – Состояния системы HA и способы восстановления

| Состояние системы | Признаки | Действие |
|---|---|---|
| HA-маркеров нет | На узле controllers[0] отсутствуют файлы-маркеры. | Исправьте ошибку, из-за которой прервался запуск, и запустите site.yml заново. |
| Маркеры есть, но миграция данных | Присутствуют один или несколько маркерных файлов. Каталоги | Откатите узел controllers[0] во временное состояние без |

| | | |
|---|---|---|
| не началась или не завершилась | /var/lib/trok-controller.orig и /var/lib/trok-auth.orig могут как существовать, так и отсутствовать. Команда blkid /dev/drbd7000 /dev/drbd7001 не показывает файловую систему ext4. | HA, затем повторите запуск заново. |
| Данные перенесены, но рабочее окружение HA не поднялось | Маркерные файлы есть. На DRBD-устройствах уже создана ext4. Каталоги .orig существуют. Наблюдаются сбои: drbd-reactor не работает, юниты монтирования или сервисы контроллера падают, VIP отсутствует. | Сначала устраните проблему в рабочем окружении HA. Откат выполняйте только в случае повреждения файловой системы или данных. |
| HA полностью работоспособен | Ровно один узел в статусе Primary, остальные — Secondary. VIP активен на текущем Primary-узле. Сервисы контроллера запущены только на активном узле. | Повторный запуск site.yml безопасен и поддерживается. |

### 10.5.3. Диагностика

Диагностические команды приведены для имени ресурса по умолчанию HA-TROK-DB и устройств /dev/drbd7000 и /dev/drbd7001. Если вы изменили ha_db_resource_name или номера minor DRBD-устройств, подставьте свои значения.

Проверка прохождения Ansible-задач. Запустите плейбук с повышенной подробностью:

```
ansible-playbook -i inventory.ini site.yml -v
```

Таблица 24 – Интерпретация

| Что увидели | Значение |
|---|---|
| Skipping move_database.yaml because this node already has post-HA markers | Система уже прошла этап первичной миграции, повторный запуск не выполняет её заново. |
| Задача move_database.yaml не выполнялась | Первый запуск HA не дошёл до этой стадии. Исправьте исходную ошибку и запустите site.yml снова. |

Проверка маркерных файлов и состояния данных на controllers[0]. Проверка маркерных файлов HA:

```
sudo ls -l \
  /etc/drbd-reactor.d/HA-TROK-DB.toml \
  /etc/systemd/system/trok-controller.service.d/10-ha-db-mount.conf \
  /etc/systemd/system/trok-auth.service.d/10-ha-db-mount.conf
```

Проверка состояния каталогов данных:

```
sudo ls -ld \
  /var/lib/trok-controller /var/lib/trok-controller.orig \
  /var/lib/trok-auth /var/lib/trok-auth.orig
```

Таблица 25 – Интерпретация

| Признак | Значение |
|---|---|
| Маркеры есть + миграция пропущена | Первый запуск пересёк защитную границу HA. |

| | |
|---|---|
| Каталоги .orig существуют | Миграция данных началась. |
| Каталоги .orig отсутствуют | Без дополнительных проверок DRBD и ФС это не гарантирует безопасность. |

Проверка DRBD и drbd-reactor на всех узлах ha_controllers:

```
sudo systemctl status drbd-reactor --no-pager

sudo journalctl -u drbd-reactor -n 100 --no-pager

sudo drbdadm role HA-TROK-DB

sudo drbdsetup status HA-TROK-DB --verbose --statistics
```

Таблица 26 – Интерпретация

| Признак | Значение |
|---|---|
| Нет Primary ни на одном узле | Владение ресурсом не стабилизировалось. |
| Цикл promote → dependency failed → demote | Проблема с юнитами монтирования или зависимостями сервисов. |
| Один Primary, остальные Secondary | DRBD в порядке. Проблема в systemd или рабочем окружении. |

Проверка файловой системы и юнитов монтирования на узле, который становится Primary:

```
sudo blkid /dev/drbd7000 /dev/drbd7001

sudo systemctl status var-lib-trok\\x2dcontroller.mount var-lib-trok\\x2dauth.mount --no-pager

sudo journalctl -u var-lib-trok\\x2dcontroller.mount -u var-lib-trok\\x2dauth.mount -n 100 --no-pager
```

Таблица 27 – Интерпретация

| Признак | Значение |
|---|---|
| Нет ext4 на /dev/drbd7000 или /dev/drbd7001 | move_database.yaml не был завершён. |

| | |
|---|---|
| Dependency failed или Unit not found | Проблема стыковки юнитов в systemd. |
| wrong fs type или bad superblock | Проблема на уровне хранилища или ФС. |

Проверка рабочего окружения контроллера и VIP:

```
sudo systemctl status trok-controller trok-auth trok-cp-endpoint --no-pager
ip addr show
```

Таблица 28 – Интерпретация

| Признак | Значение |
|---|---|
| Сервисы inactive везде | HA не поднял рабочее окружение контроллера. |
| VIP отсутствует везде | У клиентов нет точки входа в HA. |
| Сервисы активны только на Primary + VIP на том же узле | Ожидаемое здоровое состояние HA. |

### 10.5.4. Путь восстановления по стадиям

**Сценарий 1.** Откат во временное состояние без HA

Условия применения: HA-маркеры уже существуют, но первый запуск HA не завершил перенос данных.

Остановите управление HA на всех узлах ha_controllers:

```
sudo systemctl stop drbd-reactor
sudo drbdadm secondary HA-TROK-DB
```

Важно: Демонтируйте ФС только на узле, где ресурс находится в роли Primary.

Удалите маркерные файлы на controllers[0]:

```
sudo rm -f /etc/drbd-reactor.d/HA-TROK-DB.toml
sudo rm -f /etc/systemd/system/trok-controller.service.d/10-ha-db-mount.conf
sudo rm -f /etc/systemd/system/trok-auth.service.d/10-ha-db-mount.conf
```

```
sudo rm -f /etc/systemd/system/var-lib-trok\\x2dcontroller.mount
sudo rm -f /etc/systemd/system/var-lib-trok\\x2dauth.mount
sudo rm -f /etc/systemd/system/trok-db-mountpoint-immutability.service
sudo systemctl daemon-reload
```

Восстановите исходные каталоги данных (если были переименованы):

```
if [ -d /var/lib/trok-controller.orig ]; then
  sudo rm -rf /var/lib/trok-controller
  sudo mv /var/lib/trok-controller.orig /var/lib/trok-controller
fi

if [ -d /var/lib/trok-auth.orig ]; then
  sudo rm -rf /var/lib/trok-auth
  sudo mv /var/lib/trok-auth.orig /var/lib/trok-auth
fi
```

Важно. Продолжать миграцию с середины не поддерживается. Сначала верните исходное состояние, затем запускайте HA заново.

Запустите контроллер на controllers[0]:

```
sudo systemctl start trok-controller trok-auth trok-cp-endpoint
```

Проверьте: вход должен работать через исходный IP контроллера, а не через VIP.

Повторите первичный запуск HA

```
ansible-playbook -i inventory.ini site.yml
```

После завершения проверьте маркеры, роли DRBD, сервисы контроллера и VIP.

**Сценарий 2.** Исправление рабочего окружения (без отката)

Условия применения: Данные на DRBD есть (ext4 создана), первичный перенос завершён, но рабочее окружение HA не поднялось.

В этом случае:

– Оставьте HA-маркеры на месте.

– Оставьте мигрированные данные на DRBD.

– Исправьте проблему: сбойный юнит монтирования, ФС, зависимости drbd-reactor.

Типичные причины:

– systemd не может смонтировать /var/lib/trok-controller или /var/lib/trok-auth

– Ошибки зависимостей сервисов.

– Сбой drbd-reactor при поднятии ресурса.

Важно: Откат выполняйте только если повреждена ФС или сами данные на DRBD.

### 10.6. Проверка корректности поднятия HA

На каждом HA-узле выполните:

```
sudo systemctl status drbd-reactor

sudo drbdadm role HA-TROK-DB
```

Ожидаемый результат:

– drbd-reactor активен (active).

– Ровно один узел в статусе Primary, остальные – Secondary.

Примечание. При повторном запуске site.yml активный узел может остаться тем же или измениться после переключения — это нормально.

На активном HA-узле выполните:

```
sudo systemctl status trok-controller

sudo systemctl status trok-cp-endpoint

sudo systemctl status trok-auth

ip addr show
```

Ожидаемый результат:

– Сервисы trok-controller, trok-cp-endpoint, trok-auth активны только на активном узле.

– Виртуальный IP (ha_controller_vip) присутствует на активном узле.

Важное замечание:

Если после повторного запуска активная роль перешла с controllers[0] на другой HA-узел, это не ошибка. Базовые роли controller и worker не пытаются запускать сервисы локально на пассивных узлах. Управление сервисами контроллера остаётся за drbd-reactor.

Для проверки с клиентской стороны откройте в браузере:

```
http://<ha_controller_vip>
```

Ожидаемый результат: вход в TROK и работа через VIP доступны.

### 10.7. Возможные ошибки конфигурации

При настройке HA-контроллера некорректные значения переменных или неверный состав групп инвентаря могут привести к сбоям развёртывания. В таблице ниже перечислены основные проблемные сценарии и их влияние на систему.

Таблица 29 – Типовые ошибки конфигурации HA

| Проблема | Описание |
|---|---|
| Избыточный состав группы ha_controllers | Включение в группу узлов, не предусмотренных архитектурой, приводит к попытке размещения HA-ресурса на нецелевых хостах, что нарушает предполагаемую топологию. |
| Несоответствие ha_pool_name фактическому пулу | Указание имени пула, отсутствующего в системе, влечёт за собой ошибку на этапе создания HA-ресурса, поскольку роль не выполняет создание пула самостоятельно. |
| Некорректный порядок узлов в группе controllers | Первый узел в списке (controllers[0]) используется как основной для подготовки HA. Неверный выбор этого узла приводит к |

| | |
|---|---|
| | выполнению критических шагов на нецелевом хосте. |

## 10.8. Дальнейшие шаги после развёртывания HA

Развёртывание HA-контроллера завершает этап обеспечения отказоустойчивости управляющего слоя. Настройка сервисов экспорта данных (iSCSI, NFS, S3) является следующим, независимым этапом и не входит в состав процедуры установки HA.

Порядок действий для развёртывания сервисов экспорта:

– Убедитесь в успешном завершении развёртывания HA-контроллера.

– Активируйте поддержку шлюзов: enable_gateway: true.

– Определите группы узлов-экспортёров в инвентаре: iscsi_exporters, nfs_exporters, s3gw_gateways.

– Запустите соответствующие роли:

  – astra.trok.iscsi_target;

  – astra.trok.nfs_export;

  – astra.trok.s3gw.

Конфигурация сервисов экспорта выполняется с использованием тех же переменных (iscsi_targets, nfs_exports, trok_s3_gw_*), что и в сценарии обычного кластера.

# 11. ЗАВЕРШЕНИЕ УСТАНОВКИ

После завершения установки необходимо проверить, что компонент DRBD корректно интегрирован в систему. Ниже приведены ключевые команды проверки с краткими пояснениями для каждой строки или блока. Успешное выполнение команд указывает на готовность модуля DRBD к использованию.

Процедура верификации модуля DRBD применяется исключительно на узлах хранения данных (trok-worker, trok-combined). Если на узле размещается только trok-controller, наличие модуля DRBD не обязательно.

Выполните проверку версии ядра ОС (kernel release):

```
sudo uname -r
```

Пример вывода подтверждает загруженную версию ядра (здесь: 6.1.152-1-generic):

```
6.1.152-1-generic
```

DRBD – это модуль ядра Linux. Он должен быть скомпилирован и загружен именно для той версии ядра, которая сейчас активна. Несовпадение версии ядра и версии DRBD может привести к неработоспособности хранения, поэтому следующим шагом проверьте версию DRBD, чтобы сопоставить данные.

Для этого выполните команду, которая отобразит перечень установленных пакетов, применяя фильтр для вывода только тех, в наименовании которых присутствует «drbd»:

```
sudo apt list --installed | grep drbd
```

Пример вывода:

```
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.
```

Анализ представленного вывода подтверждает, что все необходимые компоненты (ядро DRBD, утилиты, интеграционные модули, сертификаты) установлены корректно. В случае отсутствия какого-либо из указанных элементов в списке результатов, требуется их установка. Судя по префиксу «drbd-6.1.152-1-generic…», используется модуль DRBD версии 9.2.12. Он подходит к текущему ядру, потому что его версия совпадает с тем, что показывает команда sudo uname -r.

Выполните команду для отображения информации о версии и сборке DRBD Admin utility:

```
   sudo drbdadm --version
```

Пример вывода:

```
   DRBDADM_BUILDTAG=GIT-hash:\ ba2ce9037989b6141222c7901d1219cf852949f1configure.ac\ build\ by\ root@linux\,\ 2025-01-17\ 18:32:01 # Метка сборки с хэшем Git и датой компиляции.

   DRBDADM_API_VERSION=2 # Версия API drbdadm (2).

   DRBD_KERNEL_VERSION_CODE=0x09020c # Код версии ядра DRBD (9.2.12).

   DRBD_KERNEL_VERSION=9.2.12 # Человеко-читаемая версия ядра DRBD.

   DRBDADM_VERSION_CODE=0x091c00 # Код версии drbdadm (9.28.0).

   DRBDADM_VERSION=9.28.0 # Версия утилиты drbdadm.
```

Вывод показывает, что утилита для управления DRBD (drbdadm) установлена и работает. Любая рассинхронизация или отсутствие – свидетельство неправильной установки и источник будущих проблем кластера.

Если версии не совпадают или утилита не работает – требуется устранить рассогласования в установке.

Проверьте загруженные модули ядра, фильтруя по «drbd»:

```
   sudo lsmod | grep -i drbd
```

Пример вывода:

```
   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.
```

DRBD должен быть не просто установлен, а также загружен в ядро ОС как модуль и готов к работе. Без этого система хранения работать не сможет – даже если все пакеты формально установлены.

Если 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.

Для подтверждения успешной установки компонентов кластера TROK рекомендуется выполнить команду на узлах контроллера:

```
   sudo systemctl status trok-auth.service trok-controller.service trok-cp-endpoint.service trok-worker.service
```

Если на узле контроллера вы не устанавливали компонент trok-worker, удалите этот аргумент из запроса:

```
   sudo systemctl status trok-auth.service trok-controller.service trok-cp-endpoint.service
```

Для узлов хранения (типа worker) команда будет иметь меньшее количество аргументов:

```
   sudo systemctl status trok-worker.service
```

Все основные компоненты (аутентификация, контроллер, endpoint, worker) должны запускаться как сервисы. Если сервис не стартовал/имел ошибку на запуске – система хранения не будет работать, как ожидается.

Если сервисы имеют статус active (running), нет ошибок типа failed, inactive и т.п., результат удачный.

Отрицательный сценарий: ошибки загрузки, сервисы в состоянии failed. Требуется диагностика: изучите логи, проверьте шаги установки/настройки.


Рисунок 25 – Результат вывода команды проверки статуса установленных служб (комбинированный тип узла)

# 12. ОСОБЕННОСТИ ПЕРЕУСТАНОВКИ КОМПОНЕНТОВ

Если при переустановке пакета trok-webui возникают ошибки (например, конфликты конфигураций или зависимостей), следуйте этой последовательности команд для полной очистки и повторной установки:

Удалите пакет с полным очищением:

```
sudo apt remove --purge trok-webui
```

Команда remove --purge удаляет пакет и все его конфигурационные файлы, а не только исполняемые файлы. Такой способ удаления предотвращает конфликты при повторной установке.

Удалите ненужные зависимости:

```
sudo apt autoremove
```

Это полезная команда для регулярного обслуживания системы, так как она эффективно очищает неиспользуемые файлы любого программного обеспечения и поддерживает оптимальную производительность.

Переустановите пакет:

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

Команды следует выполнять с правами администратора (sudo).

## 13. ЛОГИРОВАНИЕ

Логи сохраняются в директорию /var/log/trok-installer (или в указанную через --log). Имя файла лога генерируется автоматически на основе текущей даты и времени.

Пример:

/var/log/trok-installer/install_1750759134.log

Система осуществляет автоматическую регистрацию событий в соответствии со следующей схемой:

- Хранилище журналов
  - Основной каталог: /var/log/trok-installer/
  - Альтернативный путь: определяется параметром --log при запуске (подробности в пункте 5)
- Генерация имен файлов

  Формируется по шаблону: install_<timestamp>.log

  где: <timestamp> – это количество секунд, прошедших с начала эпохи Unix (01.01.1970). Например, install_1750759134.log

## 14. КРИТИЧЕСКИЕ ПРЕДУПРЕЖДЕНИЯ

Перед началом удаления обратите внимание на следующие особенности:

– Порядок удаления: Нарушение последовательности (например, удаление пулов до ресурсов) приведет к ошибкам InUse.

– Тестирование: Перед выполнением в продакшене протестируйте процедуру на стенде.

– Служебные данные: Не удаляйте директории /srv/ha/internal/ до полной остановки сервисов — это может вызвать ошибки NFS 1.

# 15. УДАЛЕНИЕ СТРУКТУРНЫХ КОМПОНЕНТОВ SDS

Удаление структурных компонентов SDS производится с использованием web-интерфейса TROK.

Очередность удаления компонентов SDS:

– 1 шаг – ресурсы;

– 2 шаг – шаблоны томов;

– 3 шаг – шаблоны ресурсов;

– 4 шаг – группы ресурсов;

– 5 шаг – пулы хранения;

– 6 шаг – узлы;

Более подробная информация указана в документе «TROK Руководство по работе с графическим интерфейсом».

# 16. ПОЛНОЕ УДАЛЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 16.1. Удаление пакетов формата deb

Для удаления установленных пакетов выполните команду:

**Внимание!** Следующая команда полностью удалит компоненты TROK и веб-сервер **nginx**. Если **nginx** использовался для других целей, не включайте его в команду удаления.

```
sudo apt purge trok-controller-meta trok-worker-meta trok-harbor-meta trok-cp-endpoint trok-auth trok-webui nginx
```

Затем удалите неиспользуемые зависимости:

```
sudo apt autoremove
```

Если необходимо удалить пакеты по отдельности, используйте команду:

```
sudo apt purge <имя_пакета>
```

где <имя_пакета> — название конкретного пакета для удаления.

Если необходимо удалить deb-пакеты по отдельности, следует учитывать, какие пакеты устанавливаются вместе с каждым компонентом системы. Полный перечень устанавливаемых пакетов представлен в примечаниях к выпуску (Release Notes) для каждой конкретной версии программного обеспечения.

## 16.2. Очистка LVM-томов и блочных устройств

Перед удалением LVM-томов и блочных устройств необходимо остановить службу trok-controller командой:

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

На узле типа worker удалите описания ресурсов:

```
sudo rm -r /etc/drbd.d/*.res
```

Перезагрузите сервер:

```
sudo reboot
```

Проверьте имеющиеся группы томов:

```
sudo lvs
```

Команда выводит текущее состояние логических томов (LV), их VG, размер и состояние.

Удалите все логические тома в вашей группе томов:

```
sudo lvremove -y /dev/<имя_группы_томов>/*
```

Команда без запроса подтверждения удаляет все логические тома внутри группы томов. Процесс полностью и безвозвратно стирает тома и связанные данные. Может потребоваться предварительно отмонтировать тома и убедиться, что данные не нужны.

Для полной очистки содержимого блочного устройства (например, диска) применяется команда:

```
sudo dd if=/dev/zero of=/dev/sd<буква_тома> bs=1G status=progress
```

где /dev/sd<буква_тома> – имя целевого устройства. Эта операция перезаписывает устройство нулями, что гарантирует удаление всех данных. Используйте с осторожностью.