



# Sharx Base 6.3

Руководство пользователя. Командная строка

Copyright © 2026 ООО «Шаркс ДЦ». Все права защищены.

Для получения дополнительной информации посетите сайт <https://sharxdc.ru/>. Все товарные знаки, торговые наименования, знаки обслуживания и логотипы, упомянутые в настоящем документе, принадлежат компании ООО «Шаркс ДЦ».

ООО «Шаркс ДЦ» оставляет за собой право вносить изменения без дополнительного уведомления в любые продукты или данные.

## Оглавление

---

1. Руководство пользователя	8
2. Общие сведения	8
2.1. Назначение Sharx Base	8
2.2. Что такое логический кластер?	8
2.3. Разделение логического кластера на виртуальные единицы	9
3. Требования	10
3.1. Требования к узлу	10
3.2. Требования к сетевой инфраструктуре	10
3.2.1. Общие требования к сетевой инфраструктуре	10
3.2.2. Требования и ограничения сегментирования сети	11
3.2.3. Требования к сетевой доступности	12
3.3. Требования к АРМ и мобильному устройству	16
3.4. Требования к интерфейсу командной строки	16
3.4.1. Стандартный sdc-cli	17
3.4.2. Отчуждаемый sdc-cli	17
4. Быстрый старт	17
5. Аутентификация	18
6. Просмотреть и изменить структуру ВЦОД	18
7. Валидация объектов ВЦОД	20
7.1. Управление ключами ВЦОД	20
7.2. Отправка сертификатов пользователям и включение двухфакторной аутентификации	21
7.3. Настройка клиентской машины для входа во ВЦОД с использованием сертификата	22
7.4. Настройка валидации объектов ВЦОД	24
7.5. Оповещение пользователя о событиях КЦ	28

## Sharx Base 6.3. Руководство пользователя. Командная строка

7.6. Рекомендации по действиям Администратора ВЦОД при нарушении контроля целостности	28
8. Уведомления	29
8.1. Создать почтовый сервер	29
8.2. Сформировать список адресов	30
8.3. Задать шаблоны сообщений	30
8.4. Настроить маршрут отправки уведомлений	31
8.5. Настроить уведомления с помощью файла YAML	31
8.6. Контроль отправленных сообщений	32
9. Права доступа и роли	32
9.1. Права доступа	32
9.2. Роли	33
9.2.1. Роли по умолчанию	34
9.2.2. Действия с ролями	35
10. Пользователи	37
10.1. Создать пользователя	37
10.2. Управлять пользователями	38
10.3. Ограничить время активности пользователя	39
10.4. Сессии пользователя	39
11. Время активности пользователя	40
12. Загрузить пользователей с использованием LDAP	42
12.1. Общий принцип работы	42
12.2. Управлять конфигурациями LDAP	43
12.2.1. Добавить конфигурацию	43
12.2.2. Просмотреть и проверить статус конфигурации LDAP-каталога	45
12.2.3. Обновить конфигурацию	46
12.2.4. Удалить конфигурацию	46
12.3. Активировать синхронизацию с LDAP	46
12.4. Проверить активацию LDAP	47

## Sharx Base 6.3. Руководство пользователя. Командная строка

13. Политика безопасности учетных записей	47
13.1. Команды управления базовыми параметрами механизмов безопасности	47
13.2. Параметры безопасности ВЦОД для обычного пользователя	48
13.3. Параметры ВЦОД для привилегированного пользователя	52
14. Двухфакторная аутентификация	54
14.1. Проверка настройки	55
15. Создать виртуальный кластер	56
15.1. Создать виртуальный кластер	56
15.2. Дополнительные команды	58
16. Управлять виртуальным кластером	58
16.1. Изменить параметры виртуального кластера	58
16.2. Управлять узлами виртуального кластера	59
16.3. Удалить виртуальный кластер	60
17. Лимиты пользователей	60
17.1. Включить режим лимитов для пользователей	61
17.2. Настроить лимиты пользователя	61
17.3. Обновить лимиты пользователя	62
17.4. Просмотреть лимиты пользователей	62
17.5. Удалить лимиты пользователей	63
18. Настроить метки виртуального кластера	63
18.1. Ограничения работы с метками в виртуальном кластере	65
18.2. Операции с метками в виртуальном кластере	65
19. Требования к хранилищу	67
19.1. Libvirt	67
19.2. РСХД	68
19.3. NFS	68
20. Журнал безопасности	68
20.1. Просмотреть события в журнале безопасности	69

20.2. Настроить параметры записи событий	70
20.2.1. Настроить параметры регистрации событий вручную	70
20.2.2. Загрузить файл с параметрами регистрации событий	71
21. Виртуальные машины	72
21.1. Аутентифицироваться во ВЦОД	72
21.2. Создать том	73
21.2.1. Libvirt	73
21.2.2. РСХД	75
21.2.3. NFS	78
21.3. Работа с томом	80
21.3.1. Libvirt	80
21.3.2. РСХД	82
21.3.3. NFS	86
21.4. Сетевые интерфейсы и правила доступа	87
21.4.1. VXLAN-подсети	87
21.4.2. Сетевые интерфейсы	89
21.4.3. Статические IP-адреса	90
21.4.4. Правила доступа	91
21.5. Правила размещения виртуальных машин	95
21.5.1. Выбор узла по идентификатору nodeName	95
21.5.2. Использование меток узлов nodeSelector	95
21.5.3. nodeAffinity, vmAffinity и colocation	97
21.6. Оптимизация ресурсов VM	103
21.6.1. Настройка переподписки ЦПУ	104
21.6.2. Настройка переподписки ОЗУ	104
21.7. Операции с виртуальными машинами	105
21.7.1. Создать VM	105
21.7.2. Просмотреть VM	108

## Sharx Base 6.3. Руководство пользователя. Командная строка

21.7.3. Обновить VM	109
21.7.4. Изменить статус VM	110
21.7.5. Назначить метку СКЗИ виртуальной машине	110
21.7.6. Мигрировать VM	111
21.7.7. Удалить VM	112
21.8. Запустить виртуальные машины	112
21.9. Снимки VM и томов	113
21.9.1. Снимки VM	113
21.9.2. Снимки томов Libvirt	115
21.9.3. Снимки томов РСХД	116
21.9.4. Снимки томов NFS	117
21.10. Шаблоны виртуальных машин	119
21.11. Резервные копии виртуальных машин	124
21.12. Восстановить виртуальную машину	126
21.12.1. Восстановить VM из резервной копии	126
21.12.2. Восстановить VM из шаблона VM	128
22. Перевести узлы в сервисный режим	129
22.1. Перевести узлы в сервисный режим	129
22.2. Отключить сервисный режим	129
23. Мониторинг	129
23.1. Общие сведения	129
23.2. Внутренний мониторинг Sharx Base	130
23.3. Внешний мониторинг. Sharx ProView	131
24. Система автоматизации процедур checker	132
24.1. Checker	132
24.1.1. Ключевые понятия	132
24.1.2. Как работает Checker?	132
24.1.3. Доступные действия	133

24.1.4. Дополнительная информация	133
24.2. Управление процедурами	133
24.2.1. Просмотреть определение процедуры	134
24.2.2. Планирование выполнения процедуры	134
24.2.3. Ручной запуск процедур	135
24.2.4. Принудительная остановка процедур	136
24.2.5. Статусы запущенных процедур	136
24.2.6. Практические советы	137
24.2.7. Дополнительная информация	138
24.3. Примеры процедур	138
24.3.1. Отслеживание и восстановление состояния VM	138
24.3.2. Дополнительная информация	139

## 1. Руководство пользователя

В данном руководстве описана последовательность действий пользователей через **командную строку** для создания и управления виртуальной инфраструктурой: ВЦОД, виртуальными кластерами, пользовательскими ресурсами, виртуальными машинами.

Сначала действия выполняет пользователь с ролью **Администратор ВЦОД**.

Затем описаны действия пользователя с ролью **Разработчик VM**.

Некоторые функции доступны другим ролям в пределах их [Прав доступа](#).

## 2. Общие сведения

### 2.1. Назначение Sharx Base

Платформа **Sharx Base** (далее – Платформа, Sharx Base) – средство виртуализации, предназначенное для развертывания и централизованного управления виртуальной инфраструктурой, частными и публичными облаками.

Платформа позволяет централизованно в едином пространстве эффективно использовать аппаратные ресурсы клиента, объединяя их в логический кластер. Наличие API позволяет интегрировать Платформу в инфраструктуру клиента.

Sharx Base является гипервизором 1 типа, то есть включает все необходимые системные и прикладные компоненты для развертывания на аппаратных средствах без необходимости установки дополнительного ПО. Доступ к прикладным компонентам предоставляется пользователям и администраторам Sharx Base через графический интерфейс, интерфейс командной строки или вызовы API. Доступ к системным компонентам не предоставляется потребителю. Данный доступ использует только техническая поддержка предприятия-изготовителя в рамках запроса потребителя на техническую поддержку платформы Sharx Base.

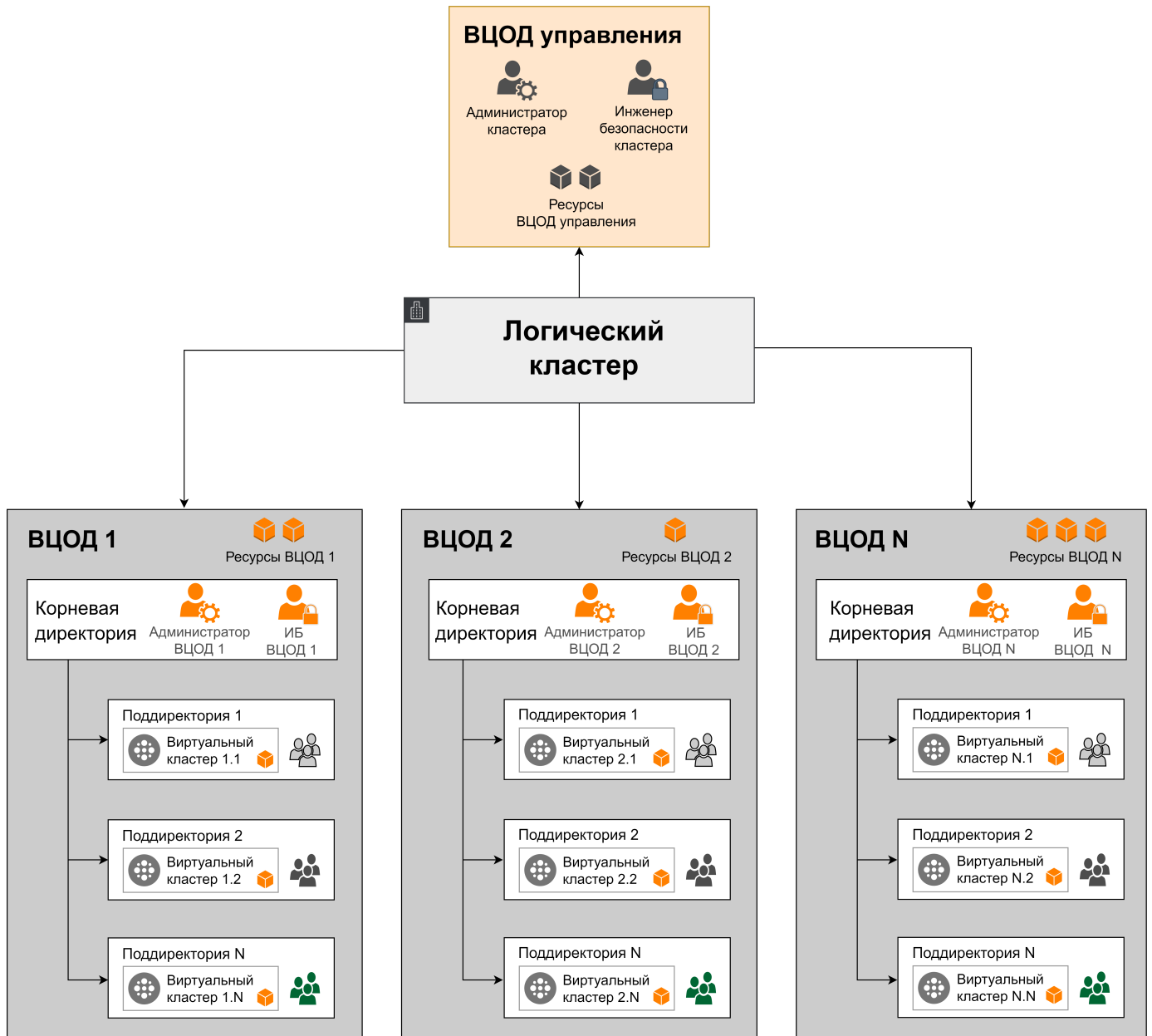
---

### 2.2. Что такое логический кластер?

**Sharx Base** объединяет все физические ресурсы серверных узлов заказчика в единый логический блок – логический кластер. Логический кластер представляет собой пространство с распределенным хранилищем высокой плотности и вычислительными ресурсами. Взаимодействие узлов кластера происходит через телекоммуникационную инфраструктуру и сетевые интерфейсы каждого узла.

### 2.3. Разделение логического кластера на виртуальные единицы

Ресурсы логического кластера разделяются на виртуальные единицы: множество ВЦОД – виртуальных центров обработки данных.



Каждый ВЦОД обладает определенными выделенными ресурсами и используется для создания, управления и взаимодействия объектов, принадлежащих исключительно данному пространству. В основе организации виртуальных ресурсов лежит иерархическая структура. По умолчанию у ВЦОД имеется корневая директория, в которой на основе конфигурации имеются управляющие роли: администратор ВЦОД и администратор безопасности ВЦОД, а также другие роли, соответствующие конфигурации.

## 3. Требования

### 3.1. Требования к узлу

#### Отказоустойчивость

Для обеспечения отказоустойчивости Платформы необходимо иметь не менее трех узлов с минимальными параметрами, указанными в таблице ниже

Таблица – Минимальные требования к узлу для установки Sharx Base.

Параметр	Ограничение, не менее
Процессор	2 (не менее 8 ядер в каждом). Архитектура x86_64, частота 2 ГГц
Оперативная память	128 Гбайт
Системный жесткий диск	SSD. Объем не менее 256 Гбайт
Жесткий диск	3 SSD. Объем не менее 1 Тбайт каждый
Сетевая карта	2 порта. Не менее 10 Гбит/с каждый

### 3.2. Требования к сетевой инфраструктуре

#### 3.2.1. Общие требования к сетевой инфраструктуре

Сетевая инфраструктура для соединения узлов в кластер должна отвечать требованиям:

1. Пропускная способность портов на сетевом оборудовании для подключения продуктивных интерфейсов не менее 10 Гбит/с.
2. Количество портов должно превышать количество узлов кластера не менее чем в два раза.
3. Должно быть настроено не менее трех VLAN, один из которых имеет параметр *native*.
4. Для каждого узла кластера должен быть выделен один порт на каждом из двух сетевых коммутаторов.

5. Должен быть выделен отдельный коммутатор сегмента управления для подключения интерфейсов управления узлов кластера.

### 3.2.2. Требования и ограничения сегментирования сети

Чтобы развернуть кластер платформы виртуализации Sharx Base, выполните подготовку сети Клиента в соответствии с настоящей статьей.

#### Важно

В статье указано минимальное количество служебных и продуктивных сетей

Типовой состав подсетей для установки и функционирования Платформы отображает Таблица ниже.

Таблица – Список выделяемых подсетей.

Название подсети	VLAN	Сеть	Шлюз
Подсеть управления Sharx Base	X	X.X.X.0/24	X.X.X.1
Подсеть внеполосного управления	Y	X.X.Y.0/24	X.X.Y.1
Подсеть распределенной системы хранения данных (РСХД)	Z	X.X.Z.0/24	шлюз не требуется
Подсеть транспортная VXLAN	V	X.X.V.0/24	шлюз не требуется
Подсеть продуктивная №1	W	X.X.W.0/24	X.X.W.1
...		...	
Подсеть продуктивная №N	N	X.X.N.0/24	X.X.N.1

**Подсеть внеполосного управления** используется для подключения и дальнейшего управления выделенными портами (access-port) управления аппаратных платформ серверов. Выделите VLAN и IP-адреса для подключения IPMI/BMC-интерфейсов серверов, укажите шлюз сети, настройте маршрутизацию. Трафик в подсети нетегированный (native). Подсеть внеполосного управления работает с MTU 1500 без использования RDMA.

**Подсеть управления Sharx Base, подсеть РСХД и продуктивные подсети** подключаются к магистральным портам (trunk-port) узлов платформы виртуализации.

**Подсеть управления Sharx Base** используется для подключения гипервизоров (узлов) и функционирования кластера Платформы виртуализации. Выделите VLAN и IP-адреса для подключения интерфейсов узлов платформы, виртуальный IP-адрес кластера, укажите шлюз сети, настройте маршрутизацию. Трафик тегированный. Подсеть управления работает с MTU 1500 без использования RDMA.

**Подсеть РСХД** используется для функционирования распределенного хранилища данных. Выделите VLAN и IP-адрес сети. Немаршрутизируемый VLAN. Трафик тегированный. Для функционирования распределенной системы хранения данных (РСХД) используется RDMA и MTU 9000.

Не рекомендуется размещать в подсети РСХД узлы, не относящиеся к системе, поскольку наличие в подсети РСХД MAC-адресов, не относящихся к системе, может приводить к помехам и задержкам при обработке данных.

**Подсеть транспортная VXLAN** используется для передачи трафика между клиентскими подсетями при использовании сетей типа VXLAN.

Данная подсеть немаршрутизируемая, внешнего доступа к ней и от нее не требуется. Размер MTU внутри подсети должен быть не менее 1600 байт.

**Продуктивные подсети** используются для размещения продуктивных ВМ. Число таких подсетей зависит от дизайна информационных систем, размещаемых на Платформе виртуализации. Выделите VLAN и IP-адреса для подключения интерфейсов ВМ, укажите шлюзы сетей, настройте маршрутизацию. Трафик тегированный.

### 3.2.3. Требования к сетевой доступности

#### Примечание

Сообщите IP-адреса доверенных NTP, DNS и LDAP-сервисов клиента технической поддержке Платформы

При использовании у Клиента ACL или политик межсетевого экранирования настройте сетевую доступность. Это обеспечит функционирование и обслуживание кластера

## Sharx Base 6.3. Руководство пользователя. Командная строка

персоналом и технической поддержкой производителя, а также размещение и эксплуатацию продуктивных информационных систем на Платформе.

Далее по тексту термин «подсеть управления» упоминается в контексте подсети управления платформой виртуализации Sharx Base, термин VIP кластера – виртуальный IP-адрес выделенного узла кластера платформы виртуализации.

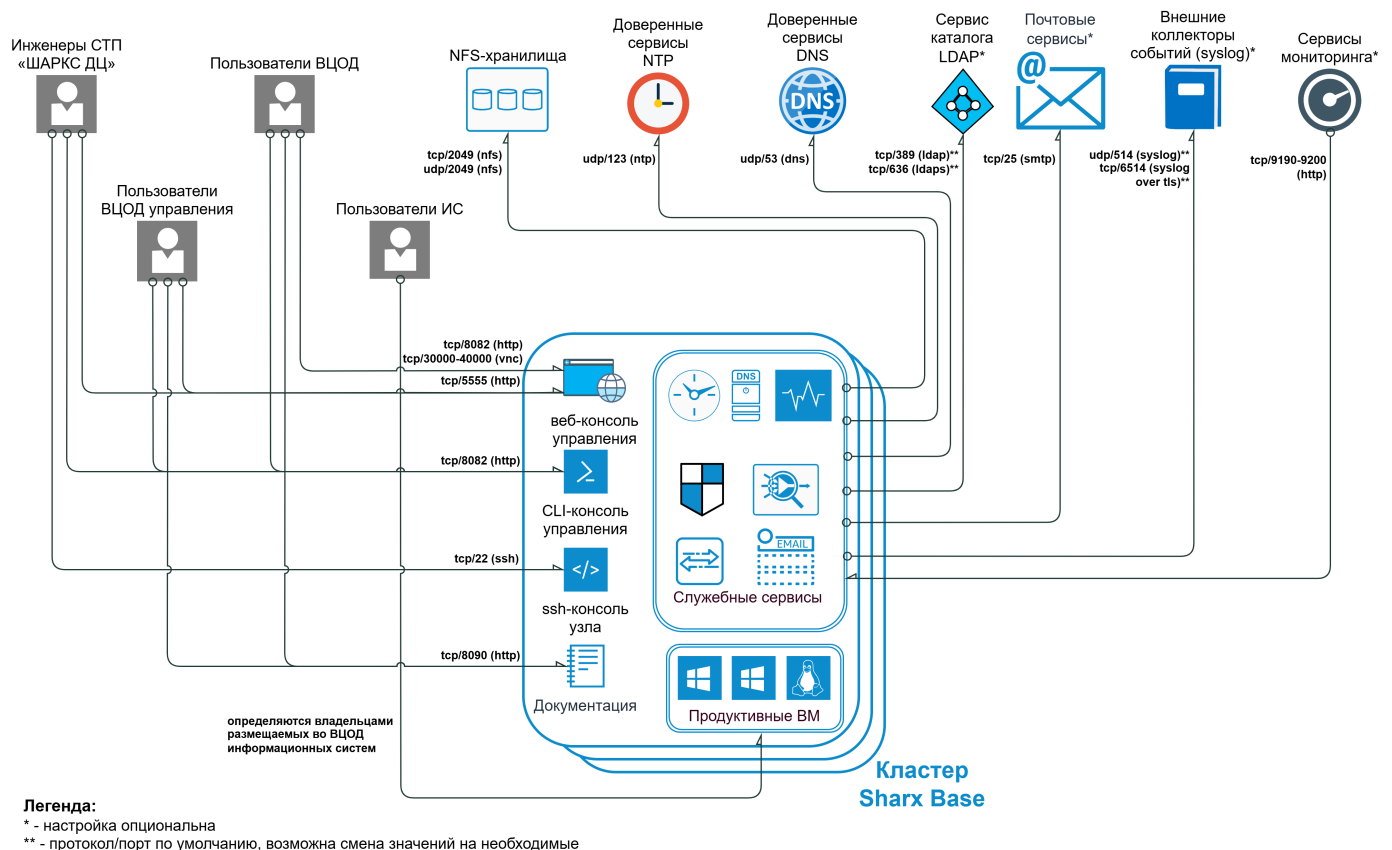


Таблица – Требования к сетевой доступности Платформы.

Описание	Источник	Приемник	Порт и протокол
Доступ персонала клиентов к веб-консоли ВЦОД управления	АРМ пользователей Sharx Base	VIP кластера в подсети управления	tcp/5555 (HTTP)
Доступ персонала	АРМ пользователей ВЦОД		tcp/80 (HTTP)

## Sharx Base 6.3. Руководство пользователя. Командная строка

Описание	Источник	Приемник	Порт и протокол
клиентов к веб-консоли ВЦОД		VIP кластера в подсети управления	
Доступ персонала клиентов к CLI-консоли ВЦОД управления/ВЦОД	АРМ пользователей ВЦОД управления/ВЦОД	VIP кластера в подсети управления	tcp/8082
Доступ службы ТП вендора к консолям управления Base	Сеть службы ТП вендора	VIP кластера в подсети управления	tcp/5555 (HTTP), tcp/8082
Доступ службы ТП вендора к SSH-консоли узлов кластера	Сеть службы ТП вендора	VIP кластера, IP-адреса узлов кластера в подсети управления	tcp/22 (SSH)
Доступ персонала клиентов к консолям управления VM	АРМ администраторов VM/инженеров DevOps	VIP кластера в подсети управления	tcp/30000-40000 (VNC)
Доступ узлов Sharx Base к NTP-серверам	IP-адреса узлов кластера в подсети управления	IP-адреса или FQDN доверенных NTP-служб	udp/123 (NTP)
Доступ узлов Sharx Base к DNS-серверам	IP-адреса узлов кластера в подсети управления	IP-адреса доверенных сервисов DNS-клиентов	udp/53 (DNS)

## Sharx Base 6.3. Руководство пользователя. Командная строка

Описание	Источник	Приемник	Порт и протокол
Доступ узлов Sharx Base к внешним NFS-хранилищам	IP-адреса узлов кластера в подсети управления	IP-адреса внешних хранилищ	tcp/2049 (NFS), udp/2049 (NFS)
Интеграция с каталогами LDAP-клиентов	VIP кластера в подсети управления	IP-адреса сервисов LDAP-серверов клиентов	tcp/389 (LDAP)*, tcp/636 (LDAPS)*
Отправка почтовых уведомлений персоналу клиентов	VIP кластера в подсети управления	IP-адреса SMTP-серверов клиентов	tcp/25 (SMTP)
Передача событий Sharx Base во внешние системы логирования (SIEM)	VIP кластера в подсети управления	IP-адреса внешних syslog-коллекторов клиентов	udp/514 (SYSLOG)*, tcp/6514 (SYSLOG OVER TLS)*
Доступ персонала к документации	АРМ персонала, администраторов безопасности, администраторов ВМ/инженеров DevOps	VIP кластера в подсети управления	tcp/8090 (HTTP)
Передача трафика VXLAN между узлами Sharx Base	IP-адреса узлов Sharx Base в сети VXLAN	IP-адреса узлов Sharx Base в сети VXLAN	udp/4789 (VXLAN)

\* Указаны протоколы и порты по умолчанию. При необходимости их можно изменить в настройках в соответствии с конфигурацией внешних сервисов.

## 3.3. Требования к АРМ и мобильному устройству

Таблица - Минимальные требования к АРМ клиента и мобильному устройству для установки Sharx Base.

Параметр	Ограничение, не менее
<b>АРМ</b>	
Процессор	Архитектура x86_64, частота 2 ГГц
Оперативная память	512 Мбайт
Свободное пространство на жестком диске	500 Мбайт
Операционная система	Linux (.deb-based или .rpm-based дистрибутив)
<b>Мобильное устройство</b>	
Операционная система	Android 12 и выше
Дополнительные приложения	SharxOTP
Дополнительные характеристики	Наличие фотокамеры

### Примечание

Дистрибутив мобильного приложения SharxOTP в формате .apk недоступен для скачивания и предоставляется [технической поддержкой 000 «Шаркс ДЦ»](#). С инструкцией по установке можно ознакомиться в статье [Установить SharxOTP](#)

## 3.4. Требования к интерфейсу командной строки

**sdc-cli** – интерфейс командной строки. Реализует пользовательский интерфейс механизма управления **Sharx Base**, позволяющий управлять кластером.

### Примечание

Не все функции реализованы в **веб-интерфейсе** Sharx Base. Поэтому `sdc-cli` необходим для работы с кластером на удаленном рабочем месте

Существуют два вида `sdc-cli`: стандартный и отчуждаемый.

#### 3.4.1. Стандартный `sdc-cli`

1. На клиентском АРМ проверьте наличие **python** версии 3.11 или выше.
2. Установите пакет `sdc-cli` – интерфейс командной строки Sharx Base.
3. Установите пакет **`sdc-pyenv3`** – набор библиотек, необходимых для работы кластера.

#### 3.4.2. Отчуждаемый `sdc-cli`

Установите пакет `sdc-cli` – интерфейс командной строки Sharx Base.

Отчуждаемый `sdc-cli` содержит `sdc-pyenv3`, поэтому дополнительная установка не требуется.

## 4. Быстрый старт

Чтобы настроить ВЦОД, выполните следующие действия:

1. [Аутентифицируйтесь во ВЦОД.](#)
2. [Просмотрите и при необходимости измените структуру ВЦОД.](#)
3. [Настройте валидацию объектов ВЦОД.](#)
4. [Настройте уведомления.](#)
5. [Создайте пользователей или измените их параметры.](#)
6. [Включите двухфакторную аутентификацию\(опционально\).](#)
7. [Создайте виртуальный кластер.](#)
8. [Проверьте, что хранилище подготовлено Администратором кластера.](#)
9. [Настройте журнал безопасности.](#)
10. [Создайте виртуальную машину.](#)

## 5. Аутентификация

### Важно

Для первичной аутентификации необходимо наличие:

- логина и пароля;
- управляющего IP-адреса кластера;
- поддерживаемой ОС;
- наличие пакета `sdc-cli` на APM пользователя.

Действия для аутентификации:

1. Введите в командной строке APM пользователя или со стороннего аппаратного средства (в случае удаленного подключения)

```
sdc-cli --host <HOST> [--tls_verify <TLS_VERIFY>] [--tls <yes|no>]
```

где

- `host` — управляющий IP-адрес кластера. IP-адрес имеет вид `a.b.c.d`;
- `tls-verify` — путь к сертификату TLS. Обязателен при включенном TLS-соединении;
- `tls` — флаг использования TLS. Значение по умолчанию — `no`. При включенном TLS-соединении обязательно использовать значения `yes`.

2. Аутентифицируйтесь во ВЦОД

```
login <LOGIN> --cluster <CLUSTER> --ns <NS>
```

где

- `login` — логин администратора ВЦОД. Значение по умолчанию — `vcod_admin`;
- `cluster` — имя логического кластера;
- `ns` — имя ВЦОД.

## 6. Просмотреть и изменить структуру ВЦОД

По умолчанию у ВЦОД имеется корневая директория, в которой размещены **Администратор ВЦОД** и **Инженер безопасности ВЦОД**. Созданный ВЦОД обладает определенными выделенными ресурсами и используется для создания, управления и взаимодействия объектов, принадлежащих исключительно данному пространству.

**Директории во ВЦОД** – это средство управления доступом пользователей к объектам, находящимся внутри этих директорий: виртуальных кластеров, VM и иных объектов виртуализации.

### Примечание

Пользователи, находящиеся в определенной директории, имеют доступ к этой директории и всем директориям, расположенным ниже по иерархии.

Подробнее о расположении пользователей во ВЦОД и их доступе читайте в статье [Пользователи](#)

Далее в пределах ВЦОД создаются поддиректории, изолированные друг от друга. В каждой из них инициализируются пользователи, имеющие доступ только к объектам данной поддиректории. При этом Администратор ВЦОД, находясь в корневой директории ВЦОД, имеет доступ ко всем ресурсам пользователей, находящихся в поддиректориях.

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

1. Чтобы посмотреть структуру ВЦОД, введите

```
aaa namespace path show
```

### Внимание

В текущей версии Sharx Base некорректно обрабатывается путь до директории, в имени которой имеется знак `.`. Поэтому название директории не должно содержать в себе знак `.`

2. Внести изменения в структуру директорий ВЦОД

```
aaa namespace path update --paths <PATHS>
```

где `paths` – имя корневой директории ВЦОД в формате JSON. По умолчанию имеет значение `"/`.

### Пример paths

```
"{\\"/\\" : {\"folder\" : null, \"folder2\" : null}}"
```

где

- / — имя корневой директории ВЦОД;
- folder, folder2 — имя поддиректории ВЦОД.

## 7. Валидация объектов ВЦОД

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

Подпись и валидация объектов используются для обеспечения [контроля целостности \(КЦ\)](#) объектов виртуальной инфраструктуры в **Sharx Base**.

Механизм работы основан на применении цифровой подписи.

При создании любого **ВЦОД** Sharx Base автоматически генерирует ключ цифровой подписи ВЦОД, который используется для контроля целостности объектов виртуальной инфраструктуры.

При создании любого **пользователя** — ключи и сертификат пользователя, которые используются для двухфакторной аутентификации пользователей.

### 7.1. Управление ключами ВЦОД

1. Чтобы посмотреть список ключей в созданном ВЦОД, введите команду

```
signer key list
```

2. Посмотреть подробную информацию о конкретном ключе можно с помощью команды

```
signer key show --keyid <KEYID>
```

где `keyid` — идентификатор ключа. Обязательный аргумент.

### 7.2. Отправка сертификатов пользователям и включение двухфакторной аутентификации

При создании пользователя внутри ВЦОД ему автоматически генерируется сертификат, который в будущем пользователь сможет использовать для двухфакторной аутентификации.

#### Важно

Перед включением двухфакторной аутентификации **Администратор ВЦОД** обязан выслать сертификаты пользователям на почту

Чтобы включить двухфакторную аутентификацию, Администратору необходимо выполнить следующие шаги:

1. Создать маршрут, в котором указать шаблон сообщения (пример [signer\\_template.yaml](#)) и почтовые адреса пользователей.

Для корректной отправки сертификатов почтовые адреса в маршруте должны совпадать с почтовыми адресами пользователей, указанными при их создании.

2. Добавить маршрут

```
aaa param update --notif_route <NOTIF_ROUTE>
```

где `notif_route` – имя маршрута отправки уведомлений о событии.

3. Отправить пользователям их сертификаты

```
aaa user cert notify --login <LOGIN>
```

где `login` – логин пользователя. Обязательный аргумент.

4. Включить вход во ВЦОД по сертификату

```
aaa param update --cert yes
```

где `cert` – включение или выключение механизма входа пользователя по сертификату.

По умолчанию – `no`. Возможные значения:

- `yes` – механизм входа пользователя по сертификату включен;
- `no` – механизм входа пользователя по сертификату выключен;

### Примечание

После включения данного параметра при аутентификации пользователя необходимо будет дополнительно указывать параметр `--cert` — идентификатор ключа сертификата пользователя, который получен пользователем в письме от Администратора.

Пример

```
login vcod_admin --cluster test --ns test --cert <CERT_KEY>
```

5. Чтобы обновить сертификат пользователя, Администратор должен ввести следующую команду

```
aaa user update --login <LOGIN> --cert yes
```

где

- `login` — логин пользователя. Обязательный аргумент.
- `cert` — включение или выключение механизма обновления сертификата пользователя.

По умолчанию — `no`. Возможные значения:

- `yes` — механизм обновления сертификата пользователя включен, старый сертификат будет удален,
- `no` — механизм обновления сертификата пользователя выключен, сертификат пользователя не изменится;

### Важно

После обновления сертификата пользователя необходимо:

- выполнить повторную отправку сертификата на почту пользователя;
- пользователю обновить на локальной машине свои сертификаты

## 7.3. Настройка клиентской машины для входа во ВЦОД с использованием сертификата

### Примечание

Действия выполняет Клиент на своем АРМ

Перед началом работы с сертификатами пользователь должен убедиться, что у него установлены следующие пакеты:

- `libksba8_1.6.3`;
- `gpgsm_2.2.40`;
- `dirmngr_2.2.40`.

Письмо от Администратора, полученное на почтовый адрес, содержит идентификатор ключа сертификата пользователя и 3 файла:

- `ns_cert` – сертификат ВЦОД;
- `user_cert` – сертификат пользователя;
- `secret` – закрытый ключ пользователя.

### Примечание

Для аутентификации пользователя во ВЦОД можно использовать идентификатор ключа `<CERT_KEY>`. По данному ключу сверяется сертификат на машине клиента с сертификатом, который есть во ВЦОД. Если они совпадают, то вход разрешается. Если нет, то выдается ошибка

После копирования приведенных выше файлов на свою машину необходимо выполнить следующие действия:

#### 1. Импортировать сертификат ВЦОД

```
gpgsm --import ns_cert
```

Посмотреть список загруженных сертификатов можно, используя команду

```
gpgsm --list-keys
```

### Примечание

Подробнее ознакомиться с аргументами команды `gpgsm` можно в [официальной документации](#)

#### 2. Создать файл `~/gnupg/trustlist.txt`, в который занести `fingerprint` сертификата ВЦОД. В конце через пробел добавить букву `S`.

### Пример

```
A5:0D:83:06:4A:E8:8E:D1:C9:EB:27:B4:5C:F6:A0:43:6F:8F:0C:59 S
```

#### 3. Импортировать сертификат пользователя

```
gpgsm --import user_cert
```

#### 4. Импортировать закрытый ключ пользователя

```
gpgsm --import secret
```

После выполнения вышеперечисленных шагов пользователю можно перейти к авторизации во ВЦОД.

### 7.4. Настройка валидации объектов ВЦОД

Для обеспечения контроля целостности на уровне виртуальной инфраструктуры ВЦОД необходимо настроить и включить валидацию объектов соответствующего ВЦОД.

#### 1. Задайте параметры для настройки валидации ВЦОД

```
signer param add [--sign_enable <SIGN_ENABLE>]
                 [--validation_action <VALIDATION_ACTION>]
                 [--sign_models <SIGN_MODELS>]
                 [--sign_objects <SIGN_OBJECTS>]
                 [--sign_objects_exclude <SIGN_OBJECTS_EXCLUDE>]
```

где

- `sign_enable` — включение или выключение механизма валидации для объектов в рамках ВЦОД при обращении к ним. По умолчанию — `no`. Возможные значения:
  - `yes` — механизм валидации включен,
  - `no` — механизм валидации выключен;
- `validation_action` — тип действия при обнаружении невалидного объекта. По умолчанию — `ERROR`. Возможные значения:
  - `ERROR` — запрет на совершение операции и вывод сообщения об ошибке,
  - `WARN` — отправка уведомления с возможностью дальнейшей работы;
- `sign_models` — набор моделей, с которыми будет работать плагин `sdcp1gn-signer`;
- `sign_objects` — набор объектов, которые существуют в рамках ВЦОД и валидацию которых нужно осуществлять;
- `sign_objects_exclude` — набор объектов, которые нужно исключить из подписываемой модели и валидацию которых осуществлять не нужно.

### Примечание

Если ввести команду без дополнительных аргументов — `signer param add`, будут созданы пустые параметры со значениями аргументов по умолчанию

2. Сгенерируйте ключ, которым будет осуществляться подпись объектов внутри ВЦОД. Для этого включите механизм валидации

```
signer param update --sign_enable yes
```

где `sign_enable` — включение или выключение механизма валидации для объектов в рамках ВЦОД при обращении к ним. По умолчанию — `no`. Возможные значения:

- `yes` — механизм валидации включен,
- `no` — механизм валидации выключен.

### Важно

В случае необходимости **изменения параметров валидации** вы можете это сделать с помощью команды `signer param update`, указав изменяемые параметры. Эти параметры такие же, как и для команды `signer param add`.

Например, для **выключения валидации**, в том числе уже подписанных объектов, необходимо ввести команду

```
signer param update --sign_enable no
```

При **повторном включении валидации** будет сгенерирован новый ключ. После этого повторно добавьте объекты в отслеживаемые для обновления их подписи

3. Добавьте модели в список отслеживаемых

а. Посмотрите список доступных моделей

```
signer sign model list
```

б. Добавьте модели в список отслеживаемых

```
signer sign model add --models <MODELS>  
                    [--force_sign <FORCE_SIGN>]
```

где

- `models` — список моделей, которые будут отслеживаться. Обязательный аргумент. При `models`, равном `*`, все модели автоматически будут добавлены в отслеживаемые;
- `force_sign` — принудительная валидация уже существующих и создаваемых позже объектов. По умолчанию — `no`. Возможные значения:

- `yes` – принудительная валидация включена,
- `no` – принудительная валидация выключена.

### Важно

Если объекты в модели существуют до того, как включается валидация, то дополнительно в команде укажите флаг `force_sign` со значением `yes`. Это принудительно подпишет уже существующие объекты.

Если внутри модели объектов нет, данный флаг указывать не нужно, все новые объекты будут отслеживаться автоматически.

При обновлении ключа подписи (повторном включении механизма валидации) все модели, которые входят в параметры, необходимо повторно добавить с использованием флага `force_sign`

с. Чтобы удалить модели из отслеживаемых, введите команду

```
signer sign model del --models <MODELS>
```

где `models` – список моделей, которые больше не будут отслеживаться. Обязательный аргумент. При `models`, равном `*`, все модели автоматически не будут отслеживаться.

4. Добавьте объекты внутри моделей в список отслеживаемых.

Плагин позволяет добавлять в список не только модели целиком, но и их отдельные части – объекты.

а. Посмотрите список всех доступных объектов внутри модели

```
signer sign entity list --model <MODEL>
```

где `model` – имя модели. Обязательный аргумент.

б. Добавьте объекты в список отслеживаемых

```
signer sign entity enable --pk <PK> --force_sign yes
```

где

- `pk` – информация об объекте в формате JSON. Обязательный аргумент;
- `force_sign` – принудительная валидация уже существующих и создаваемых позже объектов. По умолчанию – `no`. Возможные значения:
  - `yes` – принудительная валидация включена,
  - `no` – принудительная валидация выключена.



### Пример команды для добавления тома Libvirt в список отслеживаемых

```
signer sign entity enable --pk '{"classpath": "storage_libvirt_vols", "pk":
{"ctx_cluster": "sdcci6569", "ctx_ns": "vdc1", "ctx_path": "/", "name": "2.qcow2", "pool":
"test", "uuid": "7206ed44-3d52-4b30-9224-cee888e8a719"}' --force_sign yes
```

с. Чтобы удалить объект из списка отслеживаемых, введите

```
signer sign entity del --pk <PK>
```

где `pk` – информация об объекте в формате JSON. Обязательный аргумент.

5. Добавьте объекты в список исключений.

Плагин позволяет внести объект в список исключений для подписи.

Это означает, что все объекты модели при ее добавлении будут подписаны, а указанные в исключении объекты подписаны не будут и не будут отслеживаться.

а. Чтобы добавить объект в список исключений, введите команду

```
signer sign entity disable --pk <PK>
```

где `pk` – информация об объекте в формате JSON. Обязательный аргумент.

б. Чтобы удалить объект из списка исключений, введите команду

```
signer sign entity del --pk <PK>
```

где `pk` – информация об объекте в формате JSON. Обязательный аргумент.

6. Информирование о нарушении валидации.

При нарушении валидации отслеживаемого объекта автоматически в журнал безопасности будет занесено соответствующее событие.



### Пример

```
{
  "ctx_cluster": "sdcci6569",
  "ctx_ns": "vdc1",
  "data": "{\"pk\": {\"classpath\": \"scheduler_request\", \"pk\": {\"ctx_cluster\":
\"sdcci6569\", \"ctx_ns\": \"vdc1\", \"ctx_path\": \"/\", \"vcluster\": \"vcl1\", \"kind\": \"
vm\", \"name\": \"test1\", \"uuid\": \"db8ef9f0-035c-4162-8aba-79026618a0fe\"}}}\",
  "error": "Object was illegally modified. Signature is not valid",
  "event_time": "2024-09-09T08:52:04.050000+00:00",
  "group": "default",
  "level": "INFO",
  "login": null,
  "operation": "Object was illegally modified. Signature is not valid",
  "prop": null,
  "result": null,
  "session": null,
  "uuid": "9063f5ff-1194-4f26-8f40-3f222d70e48a"
},
```

Повторная автоматическая проверка произойдет через 24 часа после предыдущего обнаружения ошибки.

Каждое обращение пользователя к объекту будет приводить к дополнительной записи в журнал безопасности с информацией о том, что было обращение к объекту, у которого нарушена валидация.

7. Чтобы удалить параметры валидации ВЦОД, введите команду

```
signer param del
```

### Важно

При удалении параметров, также как и при отключении валидации, ключ подписи удаляется.

При повторном создании параметров будет сгенерирован новый ключ

## 7.5. Оповещение пользователя о событиях КЦ

Платформа позволяет настроить оповещение Администратора ВЦОД о регистрации заданных типов событий, в том числе о событиях нарушения контроля целостности. Оповещение осуществляется по электронной почте, подробнее настройка оповещений описана в разделе [Уведомления](#) Руководства пользователя Sharx Base.

## 7.6. Рекомендации по действиям Администратора ВЦОД при нарушении контроля целостности

При получении сообщения о нарушении целостности объекта виртуальной инфраструктуры Администратору рекомендуется:

1. Перевести контроль целостности на уровне виртуальной инфраструктуры в уведомительный режим (WARN).
2. Удалить объект из перечня отслеживаемых.
3. Провести анализ причин срабатывания контроля целостности на нарушение целостности объекта.
4. В случае необходимости – пересоздать объект. Рекомендуется восстановить объект из резервной копии, если имеется актуальная резервная копия объекта.

### Важно

Восстановить виртуальную машину из резервной копии пользователь Sharx Base с соответствующими правами может по инструкции в разделе [Восстановить виртуальную машину](#) Руководства пользователя.

Другие объекты можно восстановить из резервных копий, обратившись в [Техническую поддержку](#) ООО «Шаркс ДЦ»

## 8. Уведомления

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

Уведомления в **Sharx Base** используются для информирования пользователей о важных событиях в системе: отправки сертификатов при включенной двухфакторной аутентификации, настройке журнала безопасности, наличии ошибок при выполнении процессов и т. д.

### Внимание

Предварительно настройте сторонний SMTP-сервер, по которому будет осуществляться подключение

Чтобы настроить уведомления, необходимо сформировать **маршрут отправки**.

Создание маршрута состоит из следующих шагов:

#### 1. Создать **Почтовый сервер**.

Возможно существование нескольких типов почтовых серверов. Самый распространенный – SMTP-сервер для отправки почты.

#### 2. Сформировать **Адреса** для отправки сообщения. Возможно задать набор адресов для получения сообщения.

#### 3. Задать **Шаблоны**. Данные для шаблона передаются как часть сообщения, которое приходит в маршрутизатор.

#### 4. Создать **Маршрут отправки**.

### 8.1. Создать почтовый сервер

## Sharx Base 6.3. Руководство пользователя. Командная строка

1. Чтобы добавить почтовый сервер, введите в командной строке

```
notif endpoint add --descr <endpoint_description> --name <endpoint_name> --passwd <service_server_passwd> --tls <yes_or_no> --type SMTP --url <server_url> --user <service_server_user>
```

2. Запрос информации о всех почтовых серверах

```
notif endpoint list
```

3. Запрос информации об определенном почтовом сервере

```
notif endpoint show --name <endpoint_name>
```

---

## 8.2. Сформировать список адресов

1. Чтобы добавить адреса, введите в командной строке

```
notif mailing add --addresses <list_of_addresses> --descr <description> --name <mailing_name>
```

где `addresses` — адреса, существующие на SMTP-сервере и на кластере **Sharx Base**.

2. Запрос информации о всех адресах

```
notif mailing list
```

3. Запрос информации об определенном адресе

```
notif mailing show --name <mailing_name>
```

---

## 8.3. Задать шаблоны сообщений

### Примечание

Добавляйте шаблон с помощью файла YAML, так как часто возникают ошибки при внесении данных.

Файл с примером шаблона [notif.yaml](#)

1. Чтобы добавить шаблон, введите в командной строке

```
notif template add --data <template_data> --descr <template_description> --name <name>
```

2. Запрос информации о всех шаблонах сообщений

```
notif template list
```

### 3. Запрос информации об определенном шаблоне

```
notif template show --name <name>
```

## 8.4. Настроить маршрут отправки уведомлений

Маршрут объединяет в себе ранее созданные сущности `endpoint`, `mailing`, `template` и является точкой входа и отправки сообщения по конкретному пути.

### 1. Чтобы настроить маршрут, введите в командной строке

```
notif route add --name <route_name> --descr <route_description> --endpoint <endpoint_name>
--mailing <mailing_name> --sender <message_sender_name> --subject <message_subject> --
template <template>
```

### 2. Чтобы запросить информацию о всех маршрутах, введите

```
notif route list
```

### 3. Запросить информацию об определенном маршруте

```
notif route show --name <route_name>
```

## 8.5. Настроить уведомления с помощью файла YAML

### 1. Создайте файл формата YAML.

В файле опишите все необходимые параметры и команды для настройки отправки уведомлений. Пример файла автоматической настройки уведомлений [notif.yaml](#)



#### Примечание

Пример файла содержит шаблон для уведомления о невалидном объекте (неправомерном изменении объекта)

### 2. Сохраните файл настройки уведомлений.

### 3. Загрузите файл на Платформу.

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` – расположение YAML-файла.

При локальном расположении на APM пользователя введите путь до файла.

При расположении в стороннем репозитории укажите ссылку на данный файл.

### 4. Проверьте загрузку файла

```
notif <arg_notif> list
```

где `arg_notif` – аргумент для проверки конкретного параметра настройки уведомлений.

Возможные значения:

- `endpoint` – конечный сервис или сервер для отправки сообщений;
- `mailing` – список адресантов или рассылки для отправки сообщения;
- `template` – шаблон сообщения;
- `route` – маршрут отправки уведомлений.

Проверяемые параметры уведомлений должны отобразиться в общем списке.

---

## 8.6. Контроль отправленных сообщений

### 1. Чтобы проверить перечень отправленных сообщений, введите команду

```
notif message list
```

### 2. Отобразить подробную информацию об отправленном сообщении

```
notif message show --uuid <message_uuid>
```

где `uuid` – идентификатор сообщения.

## 9. Права доступа и роли

### 9.1. Права доступа

**Права доступа** – набор действий, разрешенных для выполнения субъектам над объектами данных. Права доступа определяют правила, которые затем объединяются в роли. Создание, изменение и удаление прав доступа невозможно. Список прав доступа единый для всей системы и может меняться только при ее обновлении.

Существуют следующие виды прав доступа:

## Sharx Base 6.3. Руководство пользователя. Командная строка

- **Командные** – определяют право доступа пользователя к определенной команде. Например, `cluster_list`, `aaa_user_show`.
- **Функциональные** – определяют право доступа к определенным действиям в команде. Например, `login_namespace` – позволяет выполнять авторизацию через атрибуты `cluster_to`, `ns_to`.

Права доступа имеют следующие параметры:

- `name` – имя права доступа;
- `uuid` – уникальный идентификатор в системе;
- `privileged` – флаг **привилегированности**. 1 – привилегированное, 0 – обычное;
- `descr` – описание права доступа.

Команды управления правами доступа:

### 1. Просмотр всех прав доступа во ВЦОД

```
aaa permissions list
```

### 2. Просмотр подробной информации о конкретном праве доступа.

```
aaa permissions show --name <NAME>
```

где `name` – имя права доступа.

## 9.2. Роли

**Роль** – набор прав доступа, которые необходимы пользователю или группе пользователей для выполнения определенных задач. Права доступа объединяются в роли. Затем роли присваиваются пользователю или группе пользователей. Каждый пользователь может быть связан от 1 до N ролей.

Разные роли создаются в системе на определенных уровнях объектов виртуализации. Роли и пользователей по умолчанию удалить нельзя. Новые роли и пользователи могут быть созданы администраторами в пределах их ВЦОД. Права доступа выбираются из стандартного набора без права изменения и удаления.

Роли уровня ВЦОД автоматически появляются в каждом новом создаваемом ВЦОД из стандартных конфигураций:

- администратор ВЦОД.
- администратор безопасности ВЦОД.

- администратор VM.
- разработчик VM.

Ниже рассмотрены подробные права каждой роли, созданной по умолчанию.

### 9.2.1. Роли по умолчанию

#### АДМИНИСТРАТОР ВЦОД

Функции **Администратора ВЦОД** ограничены рамками ВЦОД и включают следующее:

- настройка структуры каталогов ВЦОД;
- настройка параметров безопасности ВЦОД и интеграции ВЦОД с каталогом LDAP;
- просмотр прав доступа, создание, удаление и настройка ролей ВЦОД;
- управление сессиями доступа ВЦОД;
- создание, удаление, настройка параметров учетных записей пользователей ВЦОД;
- просмотр журналов и конфигурационных параметров журналирования событий ВЦОД;
- просмотр выделенных для ВЦОД ресурсов виртуальных сетей, настройка параметров и операции с сетевым стеком, виртуальными сетями и ACL ВЦОД;
- конфигурационные настройки и операции с сетевым стеком и ACL VM ВЦОД;
- настройка параметров механизма уведомлений и рассылок ВЦОД;
- настройка параметров и выполнение операций с виртуальными кластерами, ресурсными ограничениями виртуальных кластеров, пользовательскими ресурсами в виртуальных кластерах ВЦОД;
- создание, удаление, настройка параметров и выполнение операций по управлению состоянием VM (включение, выключение, перезагрузка);
- выполнение операций со снимками VM ВЦОД;
- выполнение операций с узлами виртуальных кластеров во ВЦОД;
- выполнение операций с метками виртуальных кластеров во ВЦОД;
- настройка параметров и выполнение операций с ключами и сертификатами ВЦОД;
- настройка параметров и выполнение операций с СХД (пулы хранения, тома для VM, снимки VM, копии дисков);
- выполнение операций мониторинга ВЦОД;
- доступ к VNC-консоли VM.

#### АДМИНИСТРАТОР БЕЗОПАСНОСТИ ВЦОД

Функции **Администратора безопасности ВЦОД**:

- просмотр всех конфигурационных параметров ВЦОД;
- настройка параметров и выполнение операций контроля целостности ВЦОД;
- настройка параметров журналирования событий ВЦОД и просмотр событий ВЦОД;
- выполнение операций мониторинга во ВЦОД.

### АДМИНИСТРАТОР ВМ

#### Функции Администратора ВМ:

- отображение списка пользовательских ресурсов виртуальных кластеров ВЦОД;
- выполнение операций по управлению состоянием ВМ (включение, выключение, перезагрузка);
- доступ к VNC-консоли ВМ.

### РАЗРАБОТЧИК ВМ

#### Функции Разработчика ВМ:

- просмотр журналов собственных событий ВЦОД;
- просмотр сведений и параметров виртуальных сетей ВЦОД;
- настройка и выполнение операций с сетевым стеком ВМ ВЦОД;
- просмотр сведений о пользовательских ресурсах и ресурсных ограничениях виртуальных кластеров ВЦОД;
- создание, удаление, настройка параметров и выполнение операций по управлению состоянием ВМ (включение, выключение, перезагрузка);
- выполнение операций по управлению снимками ВМ;
- просмотр сведений о виртуальных кластерах ВЦОД, метках виртуальных кластеров ВЦОД;
- просмотр сведений о пулах хранения СХД, томах для ВМ, снимках ВМ, шаблонах дисков ВМ;
- выполнение операций по созданию и удалению резервной копии ВМ;
- доступ к VNC-консоли ВМ.

---

### 9.2.2. Действия с ролями

**Администратор ВЦОД** может управлять ролями в рамках своего ВЦОД.

Роль имеет следующие параметры:

- `name` — имя роли;
- `uuid` — уникальный идентификатор роли;

## Sharx Base 6.3. Руководство пользователя. Командная строка

- `builtin` – флаг встроенной (`true`) или пользовательской (`false`) роли;
- `privileged` – флаг **привилегированности** роли. Зависит от наличия в роли привилегированных прав доступа. Равен `1`, если роль содержит привилегированные права, и равен `0`, если роль не содержит привилегированные права;
- `permissions` – список прав доступа, входящих в роль;
- `descr` – описание роли.

Команды для управления ролями:

1. Чтобы добавить роли (создать пользовательскую роль), в командной строке введите

```
aaa role add --role <ROLE>
             --permissions <PERMISSIONS>
             --descr <DESCR>
```

где

- `role` – имя роли;
- `permissions` – имя или список имен прав доступа без разделительных знаков;
- `descr` – описание роли.

Параметры `builtin` и `privileged` проставляются автоматически.

`Privileged` зависит от наличия привилегированных прав доступа в списке.

2. Просмотреть список всех ролей в пределах ВЦОД

```
aaa role list
```

3. Просмотреть информацию о роли

```
aaa role show --role <ROLE>
```

4. Изменить пользовательскую роль, добавить или удалить право доступа

```
aaa role update --role <ROLE>
                --permissions <PERMISSIONS>
```

### **Примечание**

В параметре `permissions` перечислите список прав доступа, исключив удаляемое право. Или выведите список действующих прав доступа, затем добавьте к нему новые права

5. Удалить пользовательскую роль.

Перед удалением роли проверьте, что она не присвоена ни одному из пользователей.

После проверки для удаления роли введите команду

```
aaa role del --role <ROLE>
```

### Примечание

Если значение параметра `role` равно `*`, будут удалены все пользовательские роли в пределах ВЦОД.

Если роль привязана к пользователю, то она не будет удалена.

Роли по умолчанию удалить нельзя

## 10. Пользователи

### 10.1. Создать пользователя

#### Примечание

В статье описано ручное создание пользователя.

Также Sharx Base поддерживает [создание пользователей с использованием LDAP](#)

1. Чтобы создать пользователя в командной строке **Sharx Base**, введите

```
aaa user add --login <LOGIN>
             [--passwd <PASSWD>]
             [--path <PATH> ]
             [--email <EMAIL> ]
             [--roles <ROLES>]
             [--whitelist_networks <WHITELIST_NETWORKS>]
```

где

- `login` — логин пользователя;
- `passwd` — пароль пользователя. Если пароль не был указан при создании, то он сформируется автоматически по следующим правилам: длина пароля — 8 символов, обязательно должны присутствовать цифры, заглавные и строчные буквы латиницы и специальные символы `!@#$$%^&*;`;
- `path` — расположение пользователя в пределах ВЦОД. Администратор ВЦОД находится в корне `/`. Пользователи в вышестоящей директории при наличии необходимых ролей могут просматривать и изменять параметры пользователей, находящихся уровнями ниже. Пользователи, находящиеся на одном уровне, но в разных поддиректориях, не могут видеть друг друга.
- `email` — электронная почта пользователя;
- `roles` — имя или список имен ролей;

- `whitelist_networks` – список белых адресов, с которых можно подключаться пользователю.

### Пример path

```
folder1: --path /folder1
```

После создания пользователя у него появляется служебный атрибут `passwd_is_generated`, доступный только для чтения. Атрибут показывает происхождение пароля пользователя:

- `true` – пароль установлен администратором;
- `false` – пароль установлен самим пользователем.

Этот флаг в связке с глобальной политикой ВЦОД `require_generated_password_change` определяет, может ли пользователь с административно заданным паролем работать в системе до его обязательной смены.

2. Чтобы посмотреть список пользователей в рамках ВЦОД, введите

```
aaa user list
```

3. Просмотр данных о пользователе

```
aaa user show --login <LOGIN>
```

## 10.2. Управлять пользователями

1. **Администратор** может изменить список ролей, пароль, почту и расположение пользователя. Для этого введите логин пользователя и изменяемые параметры

```
aaa user update --login <LOGIN>
                [--passwd <PASSWD>]
                [--prev_passwd <PREV_PASSWD>]
                [--path <PATH>]
                [--email <EMAIL> ]
                [--roles <ROLES>]
```

2. **Пользователь** может изменить только свой пароль или почту

```
aaa user param update --email <EMAIL>
                    [--passwd <PASSWD>]
                    [--prev_passwd <PREV_PASSWD>]
```

3. Изменить статус пользователя

## Sharx Base 6.3. Руководство пользователя. Командная строка

```
aaa user status update --login <LOGIN>
                        --status <STATUS>
```

где `status` — статус пользователя. Возможные значения `ACTIVE` — активный, `BLOCKED` — заблокирован.

### 4. Удалить собственный OTP-токен

```
aaa user otp del
```

### 5. Удалить токен пользователя в пределах ВЦОД

```
aaa user otp user del --login <LOGIN>
```

### 6. Изменить расположение пользователя, параметр `path`

```
aaa user path update --login <LOGIN>
                    --path <PATH>
```

### 7. Удалить пользователя

```
aaa user del --login <LOGIN>
```

После данного действия статус пользователя изменится на `REMOVED`.

Удалите пользователя из перечня вводом команды

```
aaa user clear --login <LOGIN>
```

#### **Внимание**

Невозможно изменить или удалить системных пользователей

## 10.3. Ограничить время активности пользователя

В Sharx Base реализовано ограничение по времени активности пользователя (см. соответствующий [раздел документации](#)).

## 10.4. Сессии пользователя

Возможность создания нескольких параллельных сессий для одного пользователя с разных IP-адресов регламентируется параметром настройки ВЦОД `sessions_multi_origin` (см. соответствующий [раздел документации](#)). По умолчанию такая возможность для пользователей имеется.

**Примечание**

В Sharx Base реализован механизм периодического опроса сессий пользователей. Период опроса – 30 секунд. Если сессия заканчивается, то консоль пользователя очищается и выдается сообщение: *Session has expired, please log in again*

## 11. Время активности пользователя

**Внимание**

Механизм ограничения времени активности пользователя работает только тогда, когда учетная запись пользователя не ограничена другими механизмами, например, не заблокирована.

Если для пользователя не установлено время активности, то учетная запись пользователя активна всегда

В **Sharx Base Администратор ВЦОД** может установить для пользователя время активности учетной записи, определить временной интервал и график его активности.

1. Чтобы задать время активности пользователя, в командной строке Sharx Base введите

```
aaa user active add --login <LOGIN>
                        [--date_from <DATE_FROM>]
                        [--date_before <DATE_BEFORE>]
                        [--active_weekdays <ACTIVE_WEEKDAYS>]
                        [--time_from <TIME_FROM>]
                        [--time_before <TIME_BEFORE>]
```

где

- `login` – логин пользователя, обязательный параметр;
- `date_from` – дата начала активности пользователя. Значение по умолчанию – дата создания пользователя. Формат записи – "YYYY-MM-DD", где YYYY – год, MM – месяц, DD – день;
- `date_before` – дата конца активности пользователя. Значение по умолчанию – 9999-12-31. Формат записи – "YYYY-MM-DD", где YYYY – год, MM – месяц, DD – день;
- `active_weekdays` – дни недели активности пользователя. Значение по умолчанию – `mon,tue,wed,thu,fri,sat,sun`. Формат записи – список дней недели, указанных через запятую из следующего массива: `mon,tue,wed,thu,fri,sat,sun`;
- `time_from` – время начала активности, задается в часовом поясе кластера. Значение по умолчанию – `00:00`. Формат записи – "hh:mm", где hh – часы, mm – минуты;

## Sharx Base 6.3. Руководство пользователя. Командная строка

- `time_before` — время конца активности, задается в часовом поясе кластера. Значение по умолчанию — 23:59. Формат записи — "hh:mm", где `hh` — часы, `mm` — минуты.

### Пример

```
aaa user active add --login user1 --active_from "2026-01-01" --active_before "2026-12-31" --active_weekdays mon,tue,wed,thu,fri --time_from "09:00" --time_before "19:00"
```

Учетная запись пользователя `user1` будет активна с 1 января 2026 года по 31 декабря 2026 года в будние дни с 09:00 до 19.00

Система каждые **15 секунд** проверяет активность пользовательских аккаунтов. Для каждого пользователя с текущим статусом `ACTIVE` проверяется соответствие времени его рабочему графику. В случае несоответствия статус пользователя изменяется на `INACTIVE`.

Пользователь может перейти в состояние `INACTIVE` только из состояния `ACTIVE`

2. Чтобы просмотреть параметры активности пользователя, в командной строке Sharx Base введите

```
aaa user active show --login <LOGIN>
```

где `login` — логин пользователя, обязательный параметр.

3. Чтобы изменить настройки времени активности пользователя, в командной строке Sharx Base введите

```
aaa user active update --login <LOGIN>
    [--date_from <DATE_FROM>]
    [--date_before <DATE_BEFORE>]
    [--active_weekdays <ACTIVE_WEEKDAYS>]
    [--time_from <TIME_FROM>]
    [--time_before <TIME_BEFORE>]
```

где

- `login` — логин пользователя, обязательный параметр;
- `date_from` — дата начала активности пользователя. Значение по умолчанию — дата создания пользователя. Формат записи — "YYYY-MM-DD", где `YYYY` — год, `MM` — месяц, `DD` — день;
- `date_before` — дата конца активности пользователя. Значение по умолчанию — 9999-12-31. Формат записи — "YYYY-MM-DD", где `YYYY` — год, `MM` — месяц, `DD` — день;
- `active_weekdays` — дни недели активности пользователя. Значение по умолчанию — `mon,tue,wed,thu,fri,sat,sun`. Формат записи — список дней недели, указанных через запятую из следующего массива: `mon,tue,wed,thu,fri,sat,sun`;
- `time_from` — время начала активности, задается в часовом поясе кластера. Значение по умолчанию — 00:00. Формат записи — "hh:mm", где `hh` — часы, `mm` — минуты;

- `time_before` — время конца активности, задается в часовом поясе кластера. Значение по умолчанию — 23:59. Формат записи — "hh:mm", где `hh` — часы, `mm` — минуты.

4. Чтобы удалить настройки времени активности пользователя, в командной строке Sharx Base введите

```
aaa user active del --login <LOGIN>
```

где `login` — логин пользователя, обязательный параметр.

После применения этой команды учетная запись пользователя будет активна всегда.

## 12. Загрузить пользователей с использованием LDAP

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

**Sharx Base** позволяет аутентифицировать пользователей с помощью внешнего LDAP-каталога, автоматически создавая для них учетные записи во ВЦОД и назначая роли.

Платформа совместима с тремя типами LDAP:

1. **OpenLDAP.**
2. **FreeIPA.**
3. **ActiveDirectory.**

### 12.1. Общий принцип работы

1. [Добавьте во ВЦОД конфигурацию подключения к LDAP-каталогу.](#)
2. [Активируйте конфигурацию и включите синхронизацию с ней.](#)
3. [Проверьте активацию LDAP-каталога.](#)

### Внимание

Одновременно во ВЦОД может быть активирована связь только с одним LDAP.

Если были синхронизированы несколько LDAP, то активными во ВЦОД будут только те пользователи, чья конфигурация LDAP синхронизируется в данный момент.

Если синхронизация выключена, то пользователь из LDAP не сможет аутентифицироваться во ВЦОД

## 12.2. Управлять конфигурациями LDAP

### 12.2.1. Добавить конфигурацию

Конфигурацию можно добавить с помощью команды в CLI или путем загрузки YAML-файла.

ДОБАВИТЬ КОНФИГУРАЦИЮ КОМАНДОЙ В CLI

Чтобы добавить конфигурацию, введите команду

```
aaa param ldap add --name <NAME>
                        [--url <URL>]
                        [--ca_data <CA_DATA>]
                        [--ca_file <CA_FILE>]
                        [--user <USER>]
                        [--passwd <PASSWD>]
                        [--base_dn <BASE_DN>]
                        [--query_group <QUERY_GROUP>]
                        [--query_user <QUERY_USER>]
                        [--query_active_users <QUERY_ACTIVE_USERS>]
                        [--user_map <USER_MAP>]
                        [--group_map <GROUP_MAP>]
                        [--default_path <DEFAULT_PATH>]
                        [--notif_route <NOTIF_ROUTE>]
                        [--error_limit <ERROR_LIMIT>]
```

где

- `name` — имя конфигурации LDAP в рамках текущего ВЦОД. Обязательный параметр;
- `url` — адрес сервера LDAP, к которому осуществляется подключение. Укажите IP-адрес или доменное имя сервера;
- `ca_data` — данные сертификата. Не может быть использован одновременно с `ca_file`;
- `ca_file` — путь к файлу сертификата. Не может быть использован одновременно с `ca_data`;
- `user` — пользователь на сервере LDAP, от имени которого будет осуществляться подключение;
- `passwd` — пароль пользователя на сервере LDAP, от имени которого будет осуществляться подключение;
- `base_dn` — корневой узел дерева LDAP, относительно которого система будет искать объекты;
- `query_group` — условия поиска, по которым система ищет групповые объекты в каталоге LDAP;
- `query_user` — определяет критерии поиска пользователей в каталоге LDAP;
- `query_active_users` — проверка статуса пользователя в каталоге LDAP;

## Sharx Base 6.3. Руководство пользователя. Командная строка

- `user_map` — присвоение атрибутам пользователя ВЦОД характеристик из LDAP. Например, для `login` LDAP устанавливается соответствие `uid`, для `email` LDAP устанавливается соответствие `email`, для `status` LDAP устанавливается соответствие `accountStatus`. Указывается в формате JSON;
- `group_map` — присвоение группе в каталоге LDAP роли пользователя ВЦОД. Указывается в формате JSON;
- `default_path` — путь к директории в рамках ВЦОД, куда будут размещены пользователи из LDAP;
- `notif_route` — имя заранее созданного маршрута для уведомлений об ошибках;
- `error_limit` — количество повторений одинаковой ошибки. При превышении указанного количества Sharx Base отправляет администратору сообщение о неполадках с LDAP. По умолчанию равно 3.

### Пример

```
aaa param ldap add --name <config_name>
                    --url "ldap:<ldap_addr>"
                    --user "cn=<admin_login>,dc=<dc>,dc=<dc>"
                    --passwd <ldap_admin_user_passwd>
                    --base_dn "dc=<dc>,dc=<dc>"
                    --query_group "(objectClass=<ldap_query_group>)"
                    --query_user "(objectClass=<ldap_query_user>)"
                    --query_active_users "<ldap_query_active_users>"
                    --user_map "{\"login\": \"uid\", \"email\": \"email\", \"status\":
                    \"accountStatus\"}"
                    --group_map "{\"<ldap_group>\": \"<sdс_core_role>\"}"
                    --default_path "<path_on_sdc_core_cluster>"
                    --notif_route <route_for_notification>
                    --error_limit <count_error>
```

### ДОБАВИТЬ КОНФИГУРАЦИЮ YAML-ФАЙЛОМ

1. Создайте файл конфигурации в формате YAML. Общий шаблон конфигурации приведен ниже

```
apiVersion: v1
kind: api
metadata:
  plugin:
    aaa:
      param:
        ldap: add
spec:
  name: "<config_name>"
  url: "ldap:<ldap_addr>"
  user: "cn=<admin_login>,dc=<dc>,dc=<dc>"
  passwd: "<passwd_to_LDAP>"
  base_dn: "dc=<dc>,dc=<dc>"
  query_group: "(objectClass=<ldap_query_group>)"
```

```
query_user: "(objectClass=<ldap_query_user>)"
query_active_users: "<ldap_query_active_users>"
user_map: {"login": "uid", "email": "email", "status": "accountStatus"}
group_map: {"<ldap_group>": "<sdс_core_role>"}
default_path: "/"
notif_route: "<route_name>"
error_limit: "<count_error>"
```

Описание полей приведено [выше](#).

### 2. Загрузите файл конфигурации на Платформу.

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` — расположение YAML-файла.

При локальном расположении на АРМ пользователя введите путь до файла.

При расположении в стороннем репозитории укажите ссылку на данный файл.

После загрузки рекомендуется проверить загрузку файла конфигурации и статус соединения согласно инструкции [ниже](#).

## 12.2.2. Просмотреть и проверить статус конфигурации LDAP-каталога

### 1. Просмотреть список всех конфигураций

```
aaa param ldap list
```

Обратите внимание, что указанное имя файла конфигурации, добавленное [выше](#), должно отобразиться в общем списке LDAP-конфигураций ВЦОД.

### 2. Посмотрите информацию о подгруженной конфигурации

```
aaa param ldap show --name <NAME>
```

где `name` — имя конфигурации LDAP в рамках текущего ВЦОД.

В информации отобразятся два дополнительных параметра:

- `status` — статус соединения. Возможные значения:
  - `CONNECTED` — соединение установлено,
  - `UNCONNECTED` — соединение отключено,
  - `ERROR` — ошибка соединения;
- `last_send_ldap_error_text` — запись о последней ошибке.

## 12.2.3. Обновить конфигурацию

Чтобы изменить параметры существующей конфигурации, введите

```
aaa param ldap update --name <NAME>
                        [--url <URL>]
                        [--user <USER>]
                        [--passwd <PASSWD>]
                        [--base_dn <BASE_DN>]
                        [--query_group <QUERY_GROUP>]
                        [--query_user <QUERY_USER>]
                        [--query_active_users <QUERY_ACTIVE_USERS>]
                        [--user_map <USER_MAP>]
                        [--group_map <GROUP_MAP>]
                        [--default_path <DEFAULT_PATH>]
                        [--notif_route <NOTIF_ROUTE>]
                        [--error_limit <ERROR_LIMIT>]
```

Описание полей приведено [выше](#).

## 12.2.4. Удалить конфигурацию

Чтобы удалить конфигурацию LDAP, введите

```
aaa param ldap del --name <NAME>
```

где `name` — имя конфигурации LDAP в рамках текущего ВЦОД.

## 12.3. Активировать синхронизацию с LDAP

Чтобы активировать подключение с определенным LDAP-каталогом, выполните следующие действия:

1. Укажите, какая конфигурация будет использоваться

```
aaa param update --ldap <LDAP>
```

где `ldap` — имя LDAP-каталога.

2. Включите синхронизацию с LDAP

```
aaa param update --ldap_sync yes
```

где `ldap_sync` — синхронизация с LDAP. Возможные значения:

- `yes` — синхронизация включена;
- `no` — синхронизация выключена.

### 12.4. Проверить активацию LDAP

После синхронизации пользователи из LDAP должны появиться в списке пользователей ВЦОД с закрепленными ролями согласно конфигурации.

Для проверки просмотрите пользователей ВЦОД

```
aaa user list
```

Если пользователи отображаются в списке, настройка проведена корректно.

Роли пользователям назначаются через параметр `group_map` при настройке конфигурации.

Если `group_map` не был указан, пользователи будут созданы во ВЦОД без назначенных ролей.

Пользователи могут аутентифицироваться во ВЦОД, используя свой логин и пароль из LDAP. Дополнительные настроек не требуется.

#### Примечание

Если синхронизация **выключена**, пользователь из LDAP не сможет аутентифицироваться во ВЦОД даже при наличии его учетной записи в списке

## 13. Политика безопасности учетных записей

ВЦОД по умолчанию обладает базовыми параметрами механизмов безопасности платформы. Настройки безопасности разделены между обычными и привилегированными пользователями.

#### Примечание

Параметры политики безопасности учетных записей устанавливаются в рамках текущего ВЦОД

### 13.1. Команды управления базовыми параметрами механизмов безопасности

1. Чтобы посмотреть текущие параметры политики безопасности учетных записей ВЦОД, введите в командной строке

```
aaa param show
```

## Sharx Base 6.3. Руководство пользователя. Командная строка

Описание параметров приведено в разделах [Параметры безопасности ВЦОД для обычного пользователя](#) и [Параметры безопасности ВЦОД для привилегированного пользователя](#).

2. Чтобы изменить параметры, введите

```
aaa param update --<param_name1> <value> --<param_name2> <value>
```

### 13.2. Параметры безопасности ВЦОД для обычного пользователя

Таблица – Описание параметров ВЦОД для обычного пользователя.

Параметр ВЦОД	Описание
<code>acc_block_try_cnt</code>	Количество попыток ввода некорректного логина и пароля. По умолчанию – 4. Допустимые значения – целочисленные. Диапазон от 0 до 30, где 0 – отключение проверки. Установка параметра = 0 отключает проверку счетчика попыток аутентификации, а также отключает проверки параметров <code>acc_block_try_suspend_sec</code> и <code>acc_block_try_timeout_sec</code>
<code>acc_block_try_suspend_sec</code>	Время блокировки аккаунта при превышении лимита неверных попыток ввода логина и пароля. По умолчанию – 3600 (1 ч). Допустимые значения – целочисленные. Диапазон 0, далее от 60 с (1 мин) до 86 400 с (1 сутки)
<code>acc_block_try_timeout_sec</code>	Интервал времени, в течение которого учитываются неудачные попытки входа для блокировки аккаунта. По умолчанию – 300 (5 мин)*. Допустимые значения – целочисленные
<code>acc_block_unused_days</code>	Количество неактивных дней после которого пользователь будет заблокирован. По умолчанию – 45 дней.

Параметр ВЦОД	Описание
<code>acc_delete_days</code>	Допустимые значения – целочисленные. Диапазон 0, далее от 10 до 365, где 0 – отключение проверки
<code>auth_type</code>	Тип аутентификации. Возможные значения: <ul style="list-style-type: none"> <li>• <i>BASIC</i> – логин и пароль (значение по умолчанию),</li> <li>• <i>TFA</i> – двухфакторная аутентификация</li> </ul>
<code>cert</code>	Использование сертификата при аутентификации. Возможные значения: <ul style="list-style-type: none"> <li>• <i>no</i> – не использовать (значение по умолчанию),</li> <li>• <i>yes</i> – использовать</li> </ul>
<code>ldap</code>	Имя конфигурации <b>LDAP</b> , подключаемого к Sharx Base
<code>ldap_sync</code>	Включение синхронизации с <b>LDAP</b> -сервером. Возможные значения: <ul style="list-style-type: none"> <li>• <i>no</i> – синхронизация выключена (значение по умолчанию),</li> <li>• <i>yes</i> – синхронизация включена</li> </ul>
<code>nesting_path_limit</code>	Лимит вложенности ВЦОД. По умолчанию – 3
<code>notif_route</code>	Установленный маршрут для уведомлений. По умолчанию – <code>null</code> . См. <a href="#">Маршрут отправки уведомлений</a>
<code>ns_owner_access</code>	Разрешение входа во ВЦОД Администратору кластера. Возможные значения: <ul style="list-style-type: none"> <li>• <i>yes</i> – разрешен (значение по умолчанию),</li> <li>• <i>no</i> – не разрешен</li> </ul>

Параметр ВЦОД	Описание
<code>otp_code_live_period_years</code>	Срок действия ОТР-ключа, выраженный в годах. Возможные значения – любое положительное целое число. По умолчанию – <i>3 года</i>
<code>password_diff_cnt</code>	Минимальное количество различных (неповторяющихся) символов, которое должно содержаться в пароле пользователя. По умолчанию – <i>4</i> . Допустимые значения – целочисленные. Диапазон от 0 до 20, где 0 – отключение проверки. При установке значения 0 также отключается проверка <code>password_min_exp_days</code>
<code>password_exp_days</code>	Срок действия пароля. По умолчанию – <i>60 дней</i> . 0 – отключение проверки
<code>password_min_change</code>	Минимальное количество символов, которое надо изменить при замене пароля. По умолчанию – <i>4</i>
<code>password_min_exp_days</code>	Минимальный срок, который должен пройти после установки пароля, прежде чем его можно будет сменить. По умолчанию – <i>1 день</i>
<code>password_pattern</code>	Сложность пароля Минимум 8 символов, включающих прописные буквы, строчные буквы, цифры, специальные знаки
<code>require_generated_password_change</code>	Определяет, требуется ли пользователю сменить автоматически сгенерированный пароль при первом входе во ВЦОД. Возможные значения: • <i>no</i> – не требуется (значение по умолчанию), • <i>yes</i> – требуется сменить пароль
<code>sessions_max_cnt</code>	Максимальное количество сессий пользователя. По умолчанию – <i>2</i> .

Параметр ВЦОД	Описание
	Допустимые значения – целочисленные. Диапазон от 0 до 10, где 0 – отключение проверки
<code>sessions_timeout_sec</code>	<p>Время отключения сессии при бездействии. По умолчанию – 300 с (5 мин).</p> <p>Допустимые значения – целочисленные. Диапазон 0, далее от 180 с (3 мин) до 43 200 с (12 ч), где 0 – отключение проверки</p>
<code>sessions_multi_origin</code>	<p>Определяет, разрешено ли создание нескольких параллельных сессий для одного пользователя с разных IP-адресов. Возможные значения:</p> <ul style="list-style-type: none"> <li>• <i>yes</i> – разрешено (значение по умолчанию),</li> <li>• <i>no</i> – запрещено.</li> </ul> <p><b>Примечание:</b> При смене значения с <i>yes</i> на <i>no</i> уже существующие сессии с разными IP-адресами не будут автоматически разорваны</p>
<code>sessions_timeout_sec</code>	<p>Время отключения сессии при бездействии. По умолчанию – 300 с (5 мин).</p> <p>Допустимые значения – целочисленные. Диапазон 0, далее от 300 с (5 мин) до 43 200 с (12 ч), где 0 – отключение проверки</p>
<code>tfa_client</code>	<p>Клиент для двухфакторной аутентификации. По умолчанию – <i>OTP</i></p>
<code>tfa_wait_sec</code>	<p>Таймаут для двухфакторной аутентификации. По умолчанию – 60 с</p>
<code>validation_ip</code>	<p>Проверка IP-адреса пользователя. Возможные значения:</p> <ul style="list-style-type: none"> <li>• <i>no</i> – выключена (значение по умолчанию),</li> <li>• <i>yes</i> – включена</li> </ul>
<code>whitelist_networks</code>	<p>Список белых адресов, с которых можно подключаться к Sharx Base</p>

## 13.3. Параметры ВЦОД для привилегированного пользователя

Чтобы разделить настройки безопасности между обычными и привилегированными пользователями, в параметры ВЦОД входят аналогичные поля с постфиксом `_privileged`.

Таблица – Описание параметров ВЦОД для привилегированного пользователя.

Параметр ВЦОД	Описание
<code>acc_block_try_cnt_privileged</code>	<p>Количество попыток ввода некорректного логина и пароля. По умолчанию – 4. Допустимые значения – целочисленные. Диапазон от 0 до 30, где 0 – отключение проверки. Установка параметра = 0 отключает проверку счетчика попыток аутентификации, а также отключает проверки параметров <code>acc_block_try_suspend_sec_privileged</code> и <code>acc_block_try_timeout_sec_privileged</code></p>
<code>acc_block_try_suspend_sec_privileged</code>	<p>Время блокировки аккаунта при превышении лимита неверных попыток ввода логина и пароля. По умолчанию – 3600 (1 ч). Допустимые значения – целочисленные. Диапазон 0, далее от 60 с (1 мин) до 86 400 с (1 сутки)</p>
<code>acc_block_try_timeout_sec_privileged</code>	<p>Интервал времени, в течение которого учитываются неудачные попытки входа для блокировки аккаунта. По умолчанию – 300 (5 мин)*. Допустимые значения – целочисленные</p>
<code>acc_block_unused_days_privileged</code>	<p>Количество неактивных дней после которого пользователь будет заблокирован. По умолчанию – 45 дней. Допустимые значения – целочисленные. Диапазон 0, далее от 10 до 365, где 0 – отключение проверки</p>

Параметр ВЦОД	Описание
<code>auth_type_privileged</code>	<p>Тип аутентификации. Возможные значения:</p> <ul style="list-style-type: none"> <li>• <i>BASIC</i> – логин и пароль (значение по умолчанию),</li> <li>• <i>TFA</i> – двухфакторная аутентификация</li> </ul>
<code>cert_privileged</code>	<p>Использование сертификата при аутентификации. Возможные значения:</p> <ul style="list-style-type: none"> <li>• <i>no</i> – не использовать (значение по умолчанию),</li> <li>• <i>yes</i> – использовать</li> </ul>
<code>password_diff_cnt_privileged</code>	<p>Минимальное количество различных (неповторяющихся) символов, которое должно содержаться в пароле пользователя. По умолчанию – 4. Допустимые значения – целочисленные. Диапазон от 0 до 20, где 0 – отключение проверки. При установке значения 0 также отключается проверка <code>password_min_exp_days_privileged</code></p>
<code>password_exp_days_privileged</code>	<p>Срок действия пароля. По умолчанию – 60 дней. 0 – отключение проверки</p>
<code>password_min_change_privileged</code>	<p>Минимальное количество символов, которое надо изменить при замене пароля. По умолчанию – 4</p>
<code>password_min_exp_days_privileged</code>	<p>Минимальный срок, который должен пройти после установки пароля, прежде чем его можно будет сменить. По умолчанию – 1 день</p>
<code>password_pattern_privileged</code>	<p>Сложность пароля Минимум 8 символов, включающих прописные буквы, строчные буквы, цифры, специальные знаки</p>

Параметр ВЦОД	Описание
<code>sessions_max_cnt_privileged</code>	Максимальное количество сессий пользователя. По умолчанию – 2. Допустимые значения – целочисленные. Диапазон от 0 до 10, где 0 – отключение проверки
<code>sessions_timeout_sec_privileged</code>	Время отключения сессии при бездействии. По умолчанию – 300 с (5 мин). Допустимые значения – целочисленные. Диапазон 0, далее от 180 с (3 мин) до 43 200 с (12 ч), где 0 – отключение проверки

## 14. Двухфакторная аутентификация

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

Двухфакторная аутентификация – опциональная функция.

При необходимости ее включения выполните настройки из данной статьи.

Шаги для включения двухфакторной аутентификации:

1. Посмотрите текущие настройки механизмов безопасности платформы

```
aaa param show
```

**Sharx Base** выведет текущие настройки.

Обратите внимание на параметр `auth_type`. Если его значение не равно *TFA*, то выполните действия ниже.

2. Чтобы включить двухфакторную аутентификацию введите команду

```
aaa param update --auth_type TFA
```

3. Проверьте включение двухфакторной аутентификации в настройках механизмов безопасности платформы

```
aaa param show
```

Параметр `auth_type` должен быть равен `TFA`.

### **Внимание**


Для использования двухфакторной аутентификации у пользователей на мобильном устройстве должно быть установлено приложение **SharxOTP**

## 14.1. Проверка настройки


1. Выйдите из ВЦОД.
2. [Аутентифицируйтесь во ВЦОД](#).
3. После включения двухфакторной аутентификации при первом входе пользователя генерируется OTP-токен, на экране Sharx Base отобразится QR-код. Запустите на мобильном устройстве приложение SharxOTP, введите пароль для входа в приложение.

### **Примечание**

Дистрибутив мобильного приложения SharxOTP в виде файла с расширением `.apk` недоступен для скачивания и предоставляется [технической поддержкой](#) ООО «Шаркс ДЦ». С инструкцией по установке можно ознакомиться в статье [Установить SharxOTP](#)

4. Нажмите в приложении кнопку .
5. Нажмите кнопку считывания QR-кода.
6. При необходимости в настройках мобильного устройства разрешите приложению SharxOTP снимать фото и видео.
7. Считайте сгенерированный QR-код.
8. После считывания в мобильном приложении появится шестизначный код. В течение 15 секунд после его появления введите код в командную строку Sharx Base

```
Enter OTP code: *****
```

9. Если QR-код не считывается, в приложении нажмите кнопку , затем кнопку, похожую на карандаш.

В открывшемся окне в полях введите:

- в поле **`jdoue@example.com`** введите свою *электронную почту* или *логин пользователя* в Sharx Base;
- в поле **`Example Inc`** – *имя кластера* или *адрес выделенного узла*;
- в поле **`Secret`** – значение *OTP secret* из консоли Sharx Base;

- в поле **Type** выберите флаг *TOTP*;
- в поле **Digits** – 6;
- в поле **Algorithm** – *SHA1*;
- в поле **Interval** – 15.

### 10. Нажмите кнопку **ADD TOKEN**.

Появится шестизначный код.

В течение 15 секунд после его появления введите код в командную строку Sharx Base

```
Enter OTP code: *****
```

#### **Внимание**

Код аутентификации действует 15 с

### 11. Если Вы не успели ввести код в течение 15 с:

- Оставайтесь в приложении SharxOTP.
- Нажмите на экран мобильного устройства.
- Сгенерируется новый код.
- Введите его в Sharx Base в течение 15 с.

## 15. Создать виртуальный кластер

#### **Примечание**

Действия выполняются пользователем с ролью **Администратор ВЦОД**

### 15.1. Создать виртуальный кластер

#### **Примечание**

При создании виртуального кластера узлам, входящим в него, автоматически назначаются метки, отданные во ВЦОД

#### 1. Проверьте количество доступных ресурсов

```
scheduler vcluster resource show
```

#### 2. Если ресурсов достаточно, создайте виртуальный кластер

```
scheduler vcluster add --name <NAME>
                        --labels <LABELS>
                        [--descr <DESCR>]
                        [--rf <RF>]
                        [--drain_strategy <stop|nothing|migrate>]
                        [--ulimits <yes|no>]
                        [--cpu <CPU>]
                        [--ram <RAM>]
                        [--vcpu_scaling <false|true>]
                        [--vram_scaling <false|true>]
                        [--path_to <PATH_TO>]
```

где

- `name` — имя виртуального кластера;
- `labels` — список меток для добавления в виртуальный кластер. Метки задаются в формате `key=value`, разделяются пробелами;
- `descr` — описание виртуального кластера;
- `rf` — количество узлов, возможных к выходу из строя;
- `drain_strategy` — логика планирования ресурсов в случае выведения узла из эксплуатации. Значение по умолчанию — `migrate`.

Возможные значения:

- `migrate` — перенести ресурсы на другие доступные узлы;
- `nothing` — не производить никаких действий;
- `stop` — остановить все ресурсы на текущем узле;
- `ulimits` — использование пользовательских лимитов на уровне виртуального кластера. Значение по умолчанию — `no`;

Возможные значения:

- `yes` — лимиты включены;
- `no` — лимиты выключены;
- `cpu` — лимит ЦПУ. При отсутствии параметра система вычислит его автоматически с учетом доступных на данный момент ресурсов;
- `ram` — лимит ОЗУ. При отсутствии параметра система вычислит его автоматически с учетом доступных на данный момент ресурсов. Например, единственному виртуальному кластеру в системе будут отданы все свободные ресурсы;
- `vcpu_scaling` — возможность увеличивать ВЦПУ ВМ, когда она включена. Возможные значения:
  - `false` — возможность выключена. Значение по умолчанию;
  - `true` — возможность включена. Максимальное увеличение может в 2 раза превышать начальное значение ВЦПУ при включении ВМ;

- `vram_scaling` – возможность увеличивать ОЗУ ВМ, когда она включена. Возможные значения:
  - `false` – возможность выключена. Значение по умолчанию;
  - `true` – возможность включена. Максимальное увеличение может в 2 раза превышать начальное значение ОЗУ при включении ВМ;
- `path_to` – размещение виртуального кластера в иерархии директорий.

### ✘ Внимание

Параметры `vram_scaling` и `vcpu_scaling` нельзя изменить после создания виртуального кластера

## 15.2. Дополнительные команды

1. Просмотреть список всех виртуальных кластеров в пределах ВЦОД

```
scheduler vcluster list
```

2. Просмотреть информацию о конкретном виртуальном кластере

```
scheduler vcluster show --name <NAME>
```

где `name` – имя виртуального кластера.

3. Просмотреть логины и роли пользователей, имеющих доступ к виртуальному кластеру

```
scheduler vcluster rights --name <NAME>
```

## 16. Управлять виртуальным кластером

### ✎ Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

Администратор ВЦОД может настраивать различные параметры виртуального кластера после его создания. Возможно изменить ресурсы, стратегии поведения, состав узлов и лимиты.

### 16.1. Изменить параметры виртуального кластера

Обновить параметры кластера в части ресурсов, стратегий или пользовательских лимитов

```
scheduler vcluster update --name <NAME>
                        [--descr <DESCR>]
                        [--rf <RF>]
                        [--cpu <CPU>]
                        [--ram <RAM>]
                        [--drain_strategy <stop|nothing|migrate>]
                        [--ulimits <yes|no>]
```

где

- `name` — имя виртуального кластера;
- `descr` — описание виртуального кластера;
- `rf` — количество узлов, возможных к выходу из строя;
- `cpu` — лимит ЦПУ;
- `ram` — объем оперативной памяти;
- `drain_strategy` — логика планирования ресурсов в случае выведения узла из эксплуатации. Значение по умолчанию — `migrate`.

Возможные значения:

- `migrate` — перенести ресурсы на другие доступные узлы;
- `nothing` — не производить никаких действий;
- `stop` — остановить все ресурсы на текущем узле;
- `ulimits` — использование пользовательских лимитов на уровне виртуального кластера. Значение по умолчанию — `no`.

Возможные значения:

- `yes` — лимиты включены;
- `no` — лимиты выключены.

Подробная информация о настройке описана в статье [Лимиты пользователей](#).

## 16.2. Управлять узлами виртуального кластера

### 1. Добавить узлы в виртуальный кластер

```
scheduler vcluster nodes add --name <NAME>
                             --uuids <UUIDS>
```

где

- `name` — имя изменяемого виртуального кластера;

- `uuids` — список идентификаторов добавляемых узлов.

### 2. Удалить узлы из виртуального кластера

```
scheduler vcluster nodes del --name <NAME>
                               --uuids <UUIDS>
```

где

- `name` — имя изменяемого виртуального кластера;
- `uuids` — список идентификаторов удаляемых узлов.

### 3. Просмотреть текущие узлы в составе виртуального кластера

```
scheduler vcluster nodes list
```

## 16.3. Удалить виртуальный кластер

### 1. Удалить виртуальный кластер

```
scheduler vcluster del --name <NAME>
```

где `name` — имя виртуального кластера.

### 2. Очистить список виртуальных кластеров в БД

```
scheduler vcluster clear --name <NAME>
```

## 17. Лимиты пользователей

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД**

**Лимиты пользователей** используются для контроля объема доступных ресурсов, выделенных каждому пользователю. Чтобы настроить систему пользовательских ограничений, необходимо активировать и правильно настроить соответствующие лимиты в конфигурации виртуального кластера.

Эта функция позволяет:

- Установить максимальные пределы использования ресурсов пользователями.
- Контролировать потребление вычислительных мощностей.
- Предотвратить чрезмерное использование ресурсов пользователем.

- Обеспечить распределение ресурсов между всеми пользователями кластера.

### 17.1. Включить режим лимитов для пользователей

#### Важно

Ограничьте доступ к командам изменения лимитов в правах пользователей.  
Настройка должна производиться только администраторами

Чтобы включить режим лимита пользователей, введите команду

```
scheduler vcluster update --ulimits yes
```

### 17.2. Настроить лимиты пользователя

Укажите объемы ресурсов и параметры для каждого пользователя

```
scheduler user param add --vcluster <VCLUSTER>  
                        --login <LOGIN>  
                        [--rf <RF>]  
                        [--isolation_strategy <ISOLATION_STRATEGY>]  
                        [--drain_strategy <DRAIN_STRATEGY>]  
                        [--cpu <CPU>]  
                        [--ram <RAM>]
```

где

- `vcluster` — имя виртуального кластера;
- `login` — логин пользователя;
- `rf` — фактор репликации;
- `isolation_strategy` — логика планирования ресурсов в случае изоляции узла. По умолчанию — *NOTHING*. Возможные значения:
  - `NOTHING` — не производить никаких действий,
  - `STOP` — остановить все ресурсы на текущем узле,
  - `RECREATE` — остановить все ресурсы на изолированном узле, пересоздать на доступных;
- `drain_strategy` — логика планирования ресурсов в случае выведения узла из эксплуатации. По умолчанию — *NOTHING*. Возможные значения:
  - `MIGRATE` — перенести ресурсы на другие доступные узлы,
  - `NOTHING` — не производить никаких действий,

- `STOP` — остановить все ресурсы на текущем узле;
- `cpu` — лимит ЦПУ;
- `ram` — лимит ОЗУ.

### 17.3. Обновить лимиты пользователя

```
scheduler user param update --vcluster <VCLUSTER>
                             --login <LOGIN>
                             [--rf <RF>]
                             [--isolation_strategy <ISOLATION_STRATEGY>]
                             [--drain_strategy <DRAIN_STRATEGY>]
                             [--cpu <CPU>]
                             [--ram <RAM>]
```

где

- `vcluster` — имя виртуального кластера;
- `login` — логин пользователя;
- `rf` — фактор репликации;
- `isolation_strategy` — логика планирования ресурсов в случае изоляции узла. По умолчанию — *NOTHING*. Возможные значения:
  - `NOTHING` — не производить никаких действий,
  - `STOP` — остановить все ресурсы на текущем узле,
  - `RECREATE` — остановить все ресурсы на изолированном узле, пересоздать на доступных;
- `drain_strategy` — логика планирования ресурсов в случае выведения узла из эксплуатации. По умолчанию — *NOTHING*. Возможные значения:
  - `MIGRATE` — перенести ресурсы на другие доступные узлы,
  - `NOTHING` — не производить никаких действий,
  - `STOP` — остановить все ресурсы на текущем узле;
- `cpu` — лимит ЦПУ;
- `ram` — лимит ОЗУ.

### 17.4. Просмотреть лимиты пользователей

1. Чтобы посмотреть список всех пользователей, у которых установлены пользовательские лимиты в виртуальном кластере, введите

```
scheduler user param list --vcluster <VCLUSTER>
```

где `vcluster` — имя виртуального кластера.

### 2. Просмотр информации о лимитах конкретного пользователя

```
scheduler user param admin --vcluster <VCLUSTER>  
--login <LOGIN>
```

где

- `vcluster` — имя виртуального кластера;
- `login` — логин пользователя.

### 3. Пользователь может посмотреть свои лимиты в пределах виртуального кластера командой

```
scheduler user param show --vcluster <VCLUSTER>
```

## 17.5. Удалить лимиты пользователей

Чтобы удалить лимиты пользователей, введите

```
scheduler user param del --vcluster <VCLUSTER>  
--login <LOGIN>
```

где

- `vcluster` — имя виртуального кластера;
- `login` — логин пользователя.

## 18. Настроить метки виртуального кластера

**Метка (label)** — пара ключ-значение формата `key=value`, которая описывает атрибуты объектов кластера. Играет ключевую роль в механизме планирования размещения пользовательских ресурсов (ВМ) на узлах. Метки могут быть назначены в кластере, ВЦОД, виртуальном кластере и ВМ.

В Sharx Base используются следующие типы меток: обычная, делегированная, игнорируемая, метка-ограничение, системная и криптографическая метка.

Основные понятия:

- **Обычная метка** – метка, создаваемая и назначаемая пользователями вручную. Не имеет специальных системных свойств, таких как `taint` или игнорирование.
- **Системная метка** – используется системой, недоступна для создания и назначения вручную. Например, метка активации сервисного обслуживания `system_drain_node=yes`. Подробная информация о данной метке описана в статье [CLI.Руководство администратора/Сервисное обслуживание](#).
- **Игнорируемая метка** – метка, существующая в системе, но не учитываемая при планировании. Метка может игнорироваться в контексте ВЦОД и виртуального кластера. Планировщик не учитывает игнорируемые метки во время поиска узлов по ним.
- **Делегирование метки** – предоставление возможности использовать метку объектам: ВЦОД или виртуальному кластеру.
- **Криптографическая метка** – метка, применяемая к ВМ для обозначения ее принадлежности к контуру криптографической защиты (СКЗИ). Имеет ключ-значение `vm=crypto`. Метка определяет политику управления ВМ, блокируя функции миграции и создания снимков для исключения риска утечки защищаемых данных. Криптографическую метку может назначить или снять только определенная роль специальной [командой](#).

Метки могут быть назначены следующим объектам виртуальной инфраструктуры:

- кластеру (см. [Руководство администратора](#));
- ВЦОД (см. [Руководство администратора](#));
- виртуальному кластеру.
- виртуальной машине.

**Администратор ВЦОД** управляет метками в виртуальном кластере.

Чтобы просмотреть метки, отданные во ВЦОД, введите команду

```
scheduler labels ns show [--filter <FILTER>]
```

где `filter` – фильтрация меток. Возможные значения:

- `a` – all, все метки;
- `i` – ignored, игнорируемые метки;
- `u` – unignored, все метки, кроме игнорируемых.

### ✘ Внимание

Пользователи могут видеть метки только ВЦОД, в котором они находятся.  
При попытке запроса информации о других ВЦОД будет выдаваться ошибка.  
Исключение – ВЦОД управления

## 18.1. Ограничения работы с метками в виртуальном кластере

- При создании виртуального кластера в него автоматически добавляется метка сервисного обслуживания `system_drain_node=yes` при ее наличии в родительском ВЦОД.
- В виртуальный кластер можно добавить только метки данного ВЦОД.
- В виртуальный кластер нельзя добавить метки, которые игнорируются в данном ВЦОД.
- Нельзя удалить **все** метки из виртуального кластера.
- При удалении виртуального кластера все записи о делегированных ему метках удаляются.
- Нельзя открепить метки, если по ним размещены ресурсы.
- При указании в `nodeSelector` меток, игнорируемых в кластере, ВЦОД или виртуальном кластере, будет выведена ОШИБКА.
- Системная метка `system_drain_node=yes` запрещает размещение ресурсов на соответствующих узлах.
- При обновлении ресурса изменение `nodeSelector` невозможно.

## 18.2. Операции с метками в виртуальном кластере

### ✎ Примечание

Далее в тексте понятие **Текущий ВЦОД** – это ВЦОД, в котором выполняется команда

### 1. Добавить метки в заданные виртуальные кластеры текущего ВЦОД

```
scheduler labels vcluster add --vcluster <VCLUSTER>  
                               --labels <LABELS>
```

где

- `vcluster` – список имен виртуальных кластеров. При значении `*` метка устанавливается на все виртуальные кластеры текущего ВЦОД;

- `labels` — список меток для добавления в виртуальные кластеры. Метки задаются в формате `key=value`, разделяются пробелами.

### 2. Просмотреть метки виртуальных кластеров

```
scheduler labels vcluster show [--vcluster <VCLUSTER>]
                               [--filter <FILTER>]
```

где

- `vcluster` — список имен виртуальных кластеров для просмотра их меток. При значении `*` отобразятся все метки на всех виртуальных кластерах;
- `filter` — фильтрация меток. Возможные значения:
  - `a` — all, все метки;
  - `i` — ignored, игнорируемые метки;
  - `u` — unignored, все метки, кроме игнорируемых.

### 3. Просмотреть метки со списками виртуальных кластеров, в которые они делегированы

```
scheduler labels vcluster list [--vcluster <VCLUSTER>]
                               [--filter <FILTER>]
```

где

- `vcluster` — список имен виртуальных кластеров для просмотра делегированных им меток. При значении `*` отобразятся все делегированные метки на всех виртуальных кластерах;
- `filter` — фильтрация меток. Возможные значения:
  - `a` — all, все метки;
  - `i` — ignored, игнорируемые метки;
  - `u` — unignored, все метки, кроме игнорируемых.

### 4. Просмотреть группы совместимых меток в виртуальном кластере

```
scheduler labels vcluster sharing --vcluster <VCLUSTER>
```

В результате выполнения команды будут показаны группы меток, которые можно использовать вместе для успешного планирования пользовательских ресурсов на узлы кластера.

### 5. Удалить метки из виртуальных кластеров текущего ВЦОД

### **Внимание**

- Из виртуального кластера можно удалить только метки, которые делегированы в текущий ВЦОД.
- Нельзя удалить метки, если их используют VM

```
scheduler labels vcluster del --vcluster <VCLUSTER>
                                --labels <LABELS>
```

где

- `vcluster` — список имен виртуальных кластеров для просмотра делегированных им меток. При значении `*` отобразятся все делегированные метки на всех виртуальных кластерах;
- `labels` — список меток для удаления. Метки задаются в формате `key=value`, разделяются пробелами.

## 19. Требования к хранилищу

Для работы с хранилищем в Sharx Base необходимо, чтобы Администратор кластера предварительно настроил [соответствующие ресурсы](#).

Администратор ВЦОД и Разработчик VM работают с уже подготовленным хранилищем: создают и используют тома виртуальных машин.

Sharx Base поддерживает три типа хранилища:

1. **Libvirt** — локальное хранилище узлов;
2. **PCXD** — распределённое хранилище;
3. **NFS** — сетевое файловое хранилище.

### **Примечание**

Выбор типа хранилища влияет на доступные возможности работы с томами и виртуальными машинами. например, поддержку снимков и миграции

### 19.1. Libvirt

Libvirt используется для хранения томов виртуальных машин в пулах на локальных дисках узлов.

## Sharx Base 6.3. Руководство пользователя. Командная строка

Для работы с Libvirt Администратор кластера должен [создать пул](#), прикрепив его к конкретному ВЦОД.

После этого Администратор ВЦОД и Разработчик VM могут создавать тома и использовать их в качестве дисков виртуальных машин.

Создание томов описано в статье [Libvirt. Создать том](#).

Работа с томами – в статье [Работа с томами](#).

---

### 19.2. РСХД

РСХД – распределенное хранилище, обеспечивающее отказоустойчивость и масштабируемость.

Для работы с РСХД Администратор кластера должен заранее [настроить группы размещения и шаблоны](#).

После этого Администратор ВЦОД и Разработчик VM могут создавать тома на основе шаблонов и использовать тома в виртуальных машинах.

Создание томов описано в статье [РСХД. Создать том](#).

Работа с томами – в статье [Работа с томами](#).

---

### 19.3. NFS

NFS – сетевое файловое хранилище.

Для работы с NFS Администратор кластера должен настроить [подключение к NFS-хранилищу](#).

После этого Администратор ВЦОД и Разработчик VM могут создавать тома и использовать их в виртуальных машинах.

Создание томов описано доступно только в командной строке и описано в статье [NFS. Создать том](#).

Работа с томами в командной строке – в статье [Работа с томами](#).

---

## 20. Журнал безопасности

### Примечание

Действия выполняются пользователем с ролью **Администратор безопасности ВЦОД**

**Журнал безопасности** – хронологическая запись действий пользователей в рамках ВЦОД. Параметры событий определяются политикой безопасности Клиента и иными руководящими документами.

### ✘ Внимание

Настройка и отслеживание событий происходит в рамках ВЦОД

В рамках журнала безопасности можно выполнить следующие действия:

1. [Просмотреть зарегистрированные события.](#)
2. Настроить параметры записи событий:
  - [вводом параметров регистрации событий через командную строку;](#)
  - [загрузкой файла с параметрами регистрации событий.](#)

### 20.1. Просмотреть события в журнале безопасности

1. Чтобы отобразить отслеживаемые события, введите в командной строке

```
aaaevents event list
```

2. Чтобы настроить фильтр отображаемых событий, используйте параметры, указанные в Таблице ниже.

Таблица – Параметры настройки фильтра отображаемых событий.

Параметр	Функции параметра
<code>begin, end</code>	Временной диапазон отображаемых событий
<code>full</code>	Подробная информация о событии. Включает результат завершения операции: успех или ошибку
<code>group &lt;event_group&gt;</code>	Отображение событий в пределах указанной группы
<code>level &lt;INFO, WARNING, CRITICAL&gt;</code>	Уровень событий
<code>limit &lt;number_events&gt;</code>	Отображение заданного количества событий

Параметр	Функции параметра
<code>login &lt;user_login&gt;</code>	Список всех действий пользователя за последнее время
<code>obj_uuid</code>	Отобразить события, связанные с объектом, UUID которого указан: виртуальным кластером, VM, пулом или томом Libvirt, шаблоном или томом РСХД, подключением или томом NFS
<code>operation</code>	Тип событий

3. Чтобы отобразить конкретное событие, введите

```
aaaevents event show --uuid <event_uuid>
```

где `uuid` — идентификатор события.

4. Просмотр всех типов событий в пределах ВЦОД

```
aaaevents event types
```

## 20.2. Настроить параметры записи событий

### 20.2.1. Настроить параметры регистрации событий вручную

1. Настроить уровень регистрируемых событий

```
aaaevents event loglevel --setlevel {INFO, WARNING, CRITICAL}
```

2. Задать параметры хранения событий

```
aaaevents param add --events_params <group or level> --ttl_days <days_number>
```

где

- `events_params` — указывает группы или уровни события;
- `ttl_days` — период хранения событий в днях.

3. Обновить параметры

```
aaaevents param update --events_params <group or level> --ttl_days <days_number>
```

4. Удалить параметры

```
aaaevents param del
```

## 20.2.2. Загрузить файл с параметрами регистрации событий

Параметры записи событий можно загрузить с помощью файла YAML.

1. Создайте и сохраните файл с параметрами записи событий.

Пример файла

```
apiVersion: v1
kind: api
metadata:
  plugin:
    aaaevents:
      param: add

spec:
  ttl_days: 14
  events_params:
    "User logout":
      group: "Authentication and authorization"
      level: "WARN"
      notif_route: test
    "Auth to cluster":
      group: "Authentication and authorization"
      level: "CRITICAL"
      notif_route: test
    "Add new user":
      group: "User account"
      level: "INFO"
      notif_route: test
```

где

- "User logout", "Auth to cluster", "Add new user" — типы событий;
- group — группа, к которой относится данный тип события;
- level — уровень критичности события. Доступны 3 уровня критичности: *INFO*, *WARNING*, *CRITICAL*;
- notif\_route — имя маршрута для отправки уведомлений о событии.

### Примечание

Заранее настройте [маршрут отправки уведомлений о событиях](#).  
Для каждого типа событий можно использовать отдельные маршруты записи

2. Загрузите файл на Платформу.

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` — расположение YAML-файла.

При локальном расположении на АРМ пользователя введите путь до файла.

При расположении в стороннем репозитории укажите ссылку на данный файл.

### 3. Проверьте загрузку файла

```
aaaevents param show
```

Параметры событий должны отобразиться в общем списке.

## 21. Виртуальные машины

Создание VM состоит из следующих шагов:

1. [Аутентифицируйтесь во ВЦОД](#).
2. [Создайте том](#).
3. [Создайте виртуальную машину](#).
4. [Запустите виртуальные машины](#).

### Примечание

Далее действия выполняются пользователем с ролью **Разработчик VM**.  
**Администратор ВЦОД** также может выполнять данные действия

### 21.1. Аутентифицироваться во ВЦОД

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД** или **Разработчик VM**

Действия для аутентификации:

1. Введите в командной строке АРМ пользователя или со стороннего аппаратного средства (в случае удаленного подключения)

```
sdc-cli --host <HOST> [--tls_verify <TLS_VERIFY>] [--tls <yes|no>]
```

где

- `host` — управляющий IP-адрес кластера. IP-адрес имеет вид `a.b.c.d`;
- `tls-verify` — путь к сертификату TLS. Обязателен при включенном TLS-соединении;

- `tls` – флаг использования TLS. Значение по умолчанию – `no`. При включенном TLS-соединении обязательно использовать значения `yes`.

### 2. Аутентифицируйтесь во ВЦОД

```
login <LOGIN> --cluster <CLUSTER> --ns <NS>
```

где

- `login` – логин пользователя;
- `cluster` – имя логического кластера;
- `ns` – имя ВЦОД.

## 21.2. Создать том

Том – объект хранилища, используемый виртуальной машиной как диск для хранения данных.

Тип хранилища определяет особенности создания тома и доступные параметры.

Дополнительная информация описана в статье [Требования к хранилищу](#).

### 21.2.1. Libvirt

В Libvirt том – единица деления пула.

Тома Libvirt создаются в пуле, закрепленном за текущим ВЦОД. Пул и узел указывать не требуется – том создается автоматически в доступном пуле ВЦОД

#### Важно

**Администратор кластера** заранее должен создать [пулы Libvirt](#)

Том можно создать двумя способами:

#### 1. Создать пустой том.

Используется, если требуется новый том для установки ОС, хранения данных, подключения к существующей ВМ.

#### 2. Создать том из образа.

Используется для быстрого развертывания ВМ из подготовленного образа.

Перед созданием тома проверьте доступные ресурсы командой

```
storage libvirt volume resources show
```

### СОЗДАТЬ ТОМ

Создайте том командой

```
storage libvirt volume add --name <NAME>
                             --capacity <CAPACITY>
                             --type <cdrom|system|datablock>
                             --persistent <yes|no>
                             [--frmt <FRMT>]
                             [--base_on <BASE_ON>]
                             [--descr <DESCR>]
```

где

- `name` — имя тома;
- `capacity` — объем тома в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `frmt` — формат тома. Значение по умолчанию — `qcow2`. Возможные значения: `raw`, `qcow2`;
- `base_on` — имя `base_on` тома, используемого при создании нового тома. Новый том будет скопирован с существующего тома-источника со всеми существующими данными;
- `descr` — описание тома.

### СОЗДАТЬ ТОМ ИЗ ОБРАЗА

Чтобы загрузить образ и использовать его в качестве тома Libvirt для виртуальных машин, используйте команду

```
storage libvirt image upload --name <NAME>
                              --path <PATH>
                              --type <cdrom|system|datablock>
                              --persistent <yes|no>
                              [--path_to <PATH_TO>]
                              [--local_path <LOCAL_PATH>]
                              [--description <DESCRIPTION>]
```

где

- `name` — имя тома;
- `path` — путь загрузки тома: локальный путь или URL;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `path_to` — путь размещения тома в структуре ВЦОД;

- `local_path` — путь на узле до тома при локальном расположении на узле кластера;
- `description` — описание тома.

### Примечание

Команды для работы с томами Libvirt описаны в статье [Работа с томом](#)

### 21.2.2. РСХД

#### Важно

**Администратор кластера** заранее должен создать [группы размещения и шаблоны РСХД](#)

В РСХД том — единица деления шаблона.

Чтобы создать том в хранилище РСХД:

1. Просмотрите список созданных шаблонов

```
storage sp template list
```

2. Просмотрите подробную информацию о шаблоне

```
storage sp template show --name <NAME>
```

где `name` — имя шаблона.

3. Просмотрите лимиты шаблона

```
storage sp template limit show --name <NAME>
```

где `name` — имя шаблона.

4. Чтобы создать том, введите в командной строке

```
storage sp volume add --name <NAME>
                      --size <SIZE>
                      --template <TEMPLATE>
                      --type <cdrom|system|datablock>
                      --persistent <yes|no>
                      [--bw <BW>]
                      [--iops <IOPS>]
                      [--descr <DESCR>]
                      [--parent <PARENT>]
                      [--reuse_server <REUSE_SERVER>]
                      [--base_on <BASE_ON>]
                      [--path_to <PATH_TO>]
                      [--bus <virtio|sata|ide|scsi>]
```

где

- `name` — имя тома;
- `size` — размер тома в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`. Не требуется при указании параметра `base_on`;
- `template` — имя шаблона;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `bw` — ограничение пропускной способности в килобайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `1000KB`, `2000Kb`, `3000kb`;
- `iops` — ограничение операций ввода-вывода;
- `descr` — описание тома;
- `parent` — имя снимка, на основе которого можно создать том;
- `reuse_server` — разрешение на размещение реплик на одном сервере;
- `base_on` — имя `base_on` тома, на основе которого создается новый том. Новый том будет скопирован с существующего тома-источника со всеми существующими данными.



### Примечание

Использование параметра `base_on` увеличивает время выполнения команды `add`

- `path_to` — размещение тома в иерархии директорий;
- `bus` — шина для подключения тома к ВМ. Возможные значения: `virtio`, `sata`, `ide`, `scsi`.

Также создание тома РСХД может осуществляться следующими способами:

### 1. Импортировать том в указанный шаблон

```
storage sp volume import --name <NAME>
                        --type <cdrom|system|datablock>
                        --template <TEMPLATE>
                        --persistent <yes|no>
                        --path <PATH>
                        [--descr <DESCR>]
                        [--path_to <PATH_TO>]
                        [--md5hash <MD5HASH>]
                        [--sha256hash <SHA256HASH>]
                        [--bus <virtio|sata|ide|scsi>]
```

где

## Sharx Base 6.3. Руководство пользователя. Командная строка

- `name` — имя тома;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `template` — имя шаблона;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `path` — путь загрузки тома: локальный путь или URL;
- `descr` — описание тома;
- `path_to` — размещение тома в иерархии директорий;
- `md5hash` — контрольная сумма (MD5) тома, позволяет провести проверку перед импортом тома в шаблон. Если проверка не проходит успешно, то импорт не осуществляется и пользователю возвращается ошибка `Verify hash failed. Upload aborted`;
- `sha256hash` — контрольная сумма (SHA-256) тома, позволяет провести проверку перед импортом тома в шаблон. Если проверка не проходит успешно, то импорт не осуществляется и пользователю возвращается ошибка `Verify hash failed. Upload aborted`;
- `bus` — шина для подключения тома к VM. Возможные значения: `virtio`, `sata`, `ide`, `scsi`.

### 2. Создать том из снимка

```
storage sp volume fromparent --name <NAME>
                               --snapshot_uuid <SNAPSHOT_UUID>
                               --type <cdrom|system|datablock>
                               --persistent <yes|no>
```

где

- `name` — имя тома;
- `snapshot_uuid` — идентификатор снимка;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`.

### 3. Экспортировать том в локальное хранилище

```
storage sp volume export --uuid <UUID>
                          --export_name <EXPORT_NAME>
                          [--export_format <qcow2|iso>]
```

где

- `uuid` — идентификатор тома;
- `export_name` — имя тома, получившегося в результате экспорта;
- `export_format` — формат экспорта. Возможные значения: `qcow2`, `iso`.

## Примечание

Команды для работы с томами РСХД описаны в статье [Работа с томом](#)

## 21.2.3. NFS

### Важно

Администратор кластера заранее должен [Настроить NFS-хранилище](#)

Тома – образы дисков, которые назначаются виртуальным машинам для хранения и использования информации.

### Внимание

Операции с VM, которые используют NFS-том, имеют ряд ограничений:

- нельзя делать снимки VM;
- невозможно изменить размер подключенного тома к VM;
- недоступна корректировка характеристик созданного тома;
- не работает миграция VM с подключенным томом NFS.

Создание тома NFS возможно двумя способами:

### 1. Создать том

```
storage nfs volume add --name <NAME>
                        --mount_uuid <MOUNT_UUID>
                        --type <cdrom|system|datablock>
                        --persistent <yes|no>
                        [--size <SIZE>]
                        [--base_on <BASE_ON>]
                        [--descr <DESCR>]
                        [--vm_uuid <VM_UUID>]
                        [--path_to <PATH_TO>]
                        [--bus <virtio|sata|ide|scsi>]
```

где

- `name` – имя создаваемого тома в виде `test.qcow2`. Формат тома `.qcow2` определяется через заданное имя. В настоящее время поддерживаются 2 формата: `.qcow2` и `.iso`;
- `mount_uuid` – идентификатор подключения. По нему определяется путь создания тома;

- `size` — размер создаваемого тома в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`. Не требуется при указании параметра `base_on`;
- `type` — тип хранилища. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `base_on` — имя `base_on` тома, на основе которого создается новый том. Новый том будет скопирован с существующего тома-источника со всеми существующими данными;
- `descr` — описание тома;
- `vm_uuid` — идентификатор VM;
- `path_to` — размещение тома в иерархии директорий;
- `bus` — шина для подключения тома к VM. Возможные значения: `virtio`, `sata`, `ide`, `scsi`.

### 2. Импорт NFS-тома

```
storage nfs volume import --name <NAME>
                           --mount_uuid <MOUNT_UUID>
                           --type <cdrom|system|datablock>
                           --persistent <yes|no>
                           --path <PATH>
                           [--descr <DESCR>]
                           [--path_to <PATH_TO>]
                           [--bus <virtio|sata|ide|scsi>]
```

где

- `name` — имя импортируемого тома в виде `test.qcow2`. Формат тома `.qcow2` определяется через заданное имя. В настоящее время поддерживаются 2 формата: `.qcow2` и `.iso`;
- `mount_uuid` — идентификатор подключения. По нему определяется путь импорта тома;
- `type` — тип хранилища. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `path` — путь загрузки тома: локальный путь или URL;
- `descr` — описание тома;
- `path_to` — размещение тома в иерархии директорий;
- `bus` — шина для подключения тома к VM. Возможные значения: `virtio`, `sata`, `ide`, `scsi`.

#### Примечание

Команды для работы с томами NFS описаны в статье [Работа с томом](#)

## 21.3. Работа с томом

В статье описана работа с томами Libvirt, PCXD и NFS.

Создание томов описано в статье [Создать том](#).

### 21.3.1. Libvirt

Тома Libvirt автоматически создаются в пуле, закрепленном за текущим ВЦОД.

Проверьте доступные ресурсы командой

```
storage libvirt volume resources show
```

Действия, доступные с томами Libvirt:

#### 1. Просмотреть список всех томов ВЦОД

```
storage libvirt volume list
```

#### 2. Просмотреть подробную информацию о конкретном томе

```
storage libvirt volume show [--name <NAME>]
                             [--uuid <UUID>]
                             [--get_volume_full_info <yes|no>]
```

где

- `name` — имя тома;
- `uuid` — идентификатор тома;
- `get_volume_full_info` — полное имя тома. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

#### 3. Обновить параметры тома

```
storage libvirt volume update --uuid <UUID>
                               [--capacity <CAPACITY>]
                               [--type <cdrom|system|datablock>]
                               [--persistent <yes|no>]
                               [--descr <DESCR>]
```

где

- `uuid` — идентификатор тома;
- `capacity` — объем тома в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `descr` — описание тома.

### 4. Просмотреть права доступа к тому

```
storage libvirt volume rights [--name <NAME>]
                               [--uuid <UUID>]
```

где

- `name` — имя тома;
- `uuid` — идентификатор тома.

### 5. Удалить том

```
storage libvirt volume del [--name <NAME>]
                           [--uuid <UUID>]
                           [--get_volume_full_info <yes|no>]
```

где

- `name` — имя тома;
- `uuid` — идентификатор тома;
- `get_volume_full_info` — предоставление полной информации о томе. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

### 6. Том может зависнуть в статусе *Pending* (Ожидание).

#### **Внимание**

Данная команда доступна только Администратору ВЦОД

В таких случаях удалите его из базы данных командой

```
storage libvirt volume clear [--name <NAME>]
                              [--uuid <UUID>]
                              [--get_volume_full_info <yes|no>]
```

где

- `name` — имя тома;
- `uuid` — идентификатор тома;
- `get_volume_full_info` — предоставление полной информации о томе. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

После этого создайте том заново, указав корректные параметры.

#### **Примечание**

Команда `clear` используется для удаления из списка объектов, зависших в процессе создания и фактически несозданных.

Команда `del` применяется для удаления уже созданных объектов

## 21.3.2. РСХД

В выводе информации о томе после выполнения команд `storage sp volume list` и `storage sp volume show` добавлено поле `template_type`, в котором указано к какому разделу относится данный том.

Возможные значения:

- `img` – том находится в разделе шаблона РСХД `img`, в котором хранятся все образы после создания или загрузки. Если образ персистентный, то он подключается к ВМ напрямую из этого раздела;
- `sys` – том находится в разделе шаблона РСХД `sys`, в котором хранятся `base on` образы ВМ, созданные на основе неперсистентных образов из раздела `img`

### РАБОТА С ТОМАМИ

#### 1. Просмотреть список всех томов в рамках указанного шаблона

```
storage sp volume list [--all <yes|no>]
                      [--template <TEMPLATE>]
```

где

- `all` – список всех томов.

Возможные значения:

- по умолчанию – `yes`, список всех томов;
- `no` – список томов, принадлежащих определенному шаблону `template`;
- `template` – имя шаблона при флаге `all no`.

#### 2. Просмотр подробной информации о конкретном томе

```
storage sp volume show --uuid <UUID>
```

где `uuid` – идентификатор тома.

#### 3. Обновить параметры тома

```
storage sp volume update --uuid <UUID>
                        [--name <NAME>]
                        [--descr <DESCR>]
                        [--bw <BW>]
                        [--iops <IOPS>]
                        [--reuse_server <REUSE_SERVER>]
                        [--size <SIZE>]
                        [--type <cdrom|system|datablock>]
                        [--persistent <yes|no>]
                        [--bus <virtio|sata|ide|scsi>]
```

где

- `uuid` — идентификатор тома;
- `name` — имя тома;
- `descr` — описание тома;
- `bw` — ограничение пропускной способности в килобайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `1000KB`, `2000Kb`, `3000kb`;
- `iops` — ограничение операций ввода-вывода;
- `reuse_server` — разрешение на размещение реплик на одном сервере;
- `size` — размер тома в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `type` — тип тома. Возможные значения: `cdrom`, `system`, `datablock`;
- `persistent` — возможность изменения тома. Возможные значения: `yes`, `no`;
- `bus` — шина для подключения тома к ВМ. Возможные значения: `virtio`, `sata`, `ide`, `scsi`.

### Примечание

Как увеличить размер тома, подключенного к ВМ, описано в подразделе [ниже](#)

4. Просмотреть, сколько пространства занимает том РСХД, подключенный к ВМ

```
storage sp volume status --uuid <UUID>
```

где `uuid` — идентификатор тома.

5. Конвертировать том в снимок.

### Важно

Том после команды удаляется

```
storage sp volume freeze --uuid <UUID>
```

где `uuid` — идентификатор тома.

6. Том может зависнуть в статусе *Pending (Ожидание)*.

### Внимание

Данная команда доступна только Администратору ВЦОД

В таких случаях удалите его из базы данных командой

```
storage sp volume clear --uuid <UUID>
```

где `uuid` — идентификатор тома.

После этого создайте том заново, указав корректные параметры.

### 7. Удалить том

```
storage sp volume del --uuid <UUID>
```

где `uuid` — идентификатор тома.

#### Примечание

Команда `clear` используется для удаления из списка объектов, зависших в процессе создания и фактически несозданных.

Команда `del` применяется для удаления уже созданных объектов

### УВЕЛИЧИТЬ РАЗМЕР ТОМА, ПОДКЛЮЧЕННОГО К ВИРТУАЛЬНОЙ МАШИНЕ

#### Важно

Если том подключен к ВМ, то его размер можно только увеличить

Чтобы увеличить размер тома, подключенного к виртуальной машине:

#### 1. Просмотрите список всех томов

```
storage sp volume list --all yes
```

2. Выберите из списка `base_on` том, размер которого нужно увеличить. Запишите его идентификатор.

#### 3. Введите команду

```
storage sp volume update --uuid <UUID>  
                        --size <SIZE>
```

где

- `uuid` — идентификатор `base_on` тома;
- `size` — новый размер тома в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`. Может быть только больше предыдущего размера.

4. Если виртуальная машина, к которой подключен увеличенный том, была включена, выключите ВМ командой

```
scheduler request state --name <NAME>
                        --vcluster <VCLUSTER>
                        --state stop
                        [--force <yes|no>]
```

где

- `name` — имя пользовательского ресурса, содержащего VM;
- `vcluster` — имя виртуального кластера;
- `state` — статус VM. Возможные значения:
  - `START` — запуск VM;
  - `STOP` — остановка VM;
  - `REBOOT` — перезагрузка VM;
  - `SUSPEND` — приостановка работы VM;
  - `RESUME` — возобновление работы VM;
- `force` — опция, разрешающая принудительную небезопасную перезагрузку или остановку виртуальной машины. Значение по умолчанию — `no`. Возможные значения:
  - `yes` — принудительная перезагрузка или остановка VM разрешена. При этом несохраненные данные могут быть потеряны;
  - `no` — выполняется только штатная перезагрузка или остановка.

### **Внимание**

Опция `force` действует исключительно для состояний `STOP` и `REBOOT`

## 5. Включите виртуальную машину

```
scheduler request state --name <NAME>
                        --vcluster <VCLUSTER>
                        --state start
```

где

- `name` — имя VM;
- `vcluster` — имя виртуального кластера;
- `state` — статус VM. Возможные значения:
  - `START` — запуск VM;
  - `STOP` — остановка VM;
  - `REBOOT` — перезагрузка VM;
  - `SUSPEND` — приостановка работы VM;
  - `RESUME` — возобновление работы VM.

### 21.3.3. NFS

#### Примечание

Ограничения работы с NFS-томами описаны в статье [NFS. Создать том](#)

#### 1. Просмотреть информацию о томе

```
storage nfs volume show --uuid <UUID>
                        [--name <NAME>]
                        [--mount_uuid <MOUNT_UUID>]
                        [--get_full_name <yes|no>]
```

где

- `uuid` — идентификатор тома;
- `name` — имя тома;
- `mount_uuid` — идентификатор подключения;
- `get_full_name` — полное имя тома. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`;

#### 2. Просмотреть все существующие тома

```
storage nfs volume list [--mount_uuid <MOUNT_UUID>]
                        [--get_full_name <yes|no>]
                        [--all <yes|no>]
```

где

- `mount_uuid` — идентификатор подключения;
- `get_full_name` — полное имя тома. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`;

## Sharx Base 6.3. Руководство пользователя. Командная строка

- `all` — показать `base_on` том. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

### 3. Удалить том

```
storage nfs volume del [--name <NAME>]
                        [--uuid <UUID>]
                        [--mount_uuid <MOUNT_UUID>]
                        [--get_full_name <yes|no>]
```

где

- `name` — имя тома;
- `mount_uuid` — идентификатор подключения;
- `get_full_name` — полное имя тома. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

4. Том может зависнуть в статусе *Pending (Ожидание)*. В таких случаях удалите его из базы данных командой

```
storage nfs volume clear --uuid <UUID>
```

где `uuid` — идентификатор тома.

После этого создайте том заново, указав корректные параметры.

## 21.4. Сетевые интерфейсы и правила доступа

Статья описывает операции с сетевыми ресурсами, доступные **Администратору ВЦОД** после настройки базовой инфраструктуры [Администратором кластера](#).

### Основные операции управления сетями:

1. [VXLAN-подсети](#). Создание и управление IP-сегментами в overlay-сетях.
2. [Сетевые интерфейсы](#). Просмотр и изменение параметров интерфейсов VM.
3. [Статические IP-адреса](#). Резервирование фиксированных адресов.
4. [Правила доступа \(ACL\)](#). Настройка политик безопасности для сетей и портов VM.

#### Примечание

Базовая настройка сетевой инфраструктуры должна быть выполнена Администратором кластера в соответствии с [Руководством администратора](#)

### 21.4.1. VXLAN-подсети

VXLAN-подсеть – это IP-сегмент внутри VXLAN-сети, который предоставляет услуги маршрутизации для виртуальных машин.

Особенности VXLAN-подсетей:

- каждая подсеть создается внутри родительской VXLAN-сети и наследует ее ВЦОД;
- при создании подсети автоматически назначается VID из диапазона `vxlan_vid_range`;
- не поддерживают применение правил списков контроля доступа.

Действия с VXLAN-подсетями:

1. Чтобы создать VXLAN-подсеть, введите в командной строке

```
ipam vxlan subnet add --name <NAME>
                        --vxlan_uuid <VXLAN_UUID>
                        --network <NETWORK>
                        --ip_first <IP_FIRST>
                        --ip_last <IP_LAST>
                        [--gateway <GATEWAY>]
                        [--dns_1 <DNS_1>]
                        [--dns_2 <DNS_2>]
                        [--desc <DESC>]
```

где

- `name` – имя подсети;
- `vxlan_uuid` – идентификатор родительской VXLAN-сети;
- `network` – IP-адрес подсети в формате CIDR;
- `ip_first` – начало диапазона DHCP IP-адресов;
- `ip_last` – конец диапазона DHCP IP-адресов;
- `gateway` – адрес шлюза по умолчанию;
- `dns_1` – адрес первичного DNS-сервера;
- `dns_2` – адрес вторичного DNS-сервера;
- `desc` – описание VXLAN-подсети.

### Пример

```
ipam vxlan subnet add --name vx-sub-1
                        --vxlan_uuid <UUID parent VXLAN>
                        --network 10.1.2.0/24
                        --gateway 10.1.2.1
                        --ip_first 10.1.2.2
                        --ip_last 10.1.2.254
```

2. Просмотреть список подсетей

```
ipam vxlan subnet list [--vxlan_uuid <VXLAN_UUID>]
```

где `vxlan_uuid` — идентификатор родительской VXLAN-сети.

Из текущего ВЦОД видны только подсети, находящиеся в нем. При указании `vxlan_uuid` отобразятся подсети конкретной VXLAN-сети.

### 3. Просмотреть подробную информацию о VXLAN-подсети

```
ipam vxlan subnet show --subnet_uuid <SUBNET_UUID>
```

где `subnet_uuid` — идентификатор VXLAN-подсети.

### 4. Просмотреть интерфейсы VM в VXLAN-подсети

```
ipam iface list --net_uuid <SUBNET_UUID>
```

где `net_uuid` — идентификатор VXLAN-подсети.

### 5. Удалить VXLAN-подсеть

#### **Внимание**

Нельзя удалить VXLAN-подсеть, если в ней есть интерфейсы VM

```
ipam vxlan subnet del --subnet_uuid <SUBNET_UUID>
```

где `subnet_uuid` — идентификатор VXLAN-подсети.

## 21.4.2. Сетевые интерфейсы

Sharx Base автоматически управляет сетевыми интерфейсами:

- При создании VM в VXLAN-подсети система автоматически создает сетевой интерфейс.
- Для каждого интерфейса генерируются уникальные MAC и IP-адреса.
- При удалении VM все связанные сетевые интерфейсы освобождаются автоматически.
- Освобожденные IP-адреса могут быть назначены новым VM.

Также сетевыми интерфейсами можно управлять вручную.

### 1. Посмотреть список сетевых интерфейсов

```
ipam iface list [--net_uuid <NET_UUID>]
                [--vxlan_uuid <VXLAN_UUID>]
                [--show_acl <yes|no>]
```

где

- `net_uuid` — идентификатор сети. Обязателен;

- `vxlan_uuid` — идентификатор VXLAN. Обязателен;

### **Примечание**

Для выполнения команды необходимо указать один из параметров: `net_uuid` или `vxlan_uuid`

- `show_acl` — показать правила списков контроля доступа для каждого интерфейса. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

### 2. Обновить IP-адрес сетевого интерфейса виртуальной машины

```
ipam iface update --iface_id <IFACE_ID>
                  --ip <IP>
```

где

- `iface_id` — идентификатор сетевого интерфейса, который нужно изменить;
- `ip` — новый IP-адрес для указанного сетевого интерфейса.

## 21.4.3. Статические IP-адреса

### 1. Чтобы создать статический IP-адрес, введите

```
ipam addr virtual add --net_uuid <NET_UUID>
                      --ip <IP>
```

где

- `net_uuid` — идентификатор сети;
- `ip` — новый виртуальный IP-адрес для указанной сети.

### 2. Посмотреть список созданных адресов

```
ipam addr virtual list --net_uuid <NET_UUID>
```

где `net_uuid` — идентификатор сети;

### 3. Удалить статический IP-адрес

```
ipam addr virtual del --net_uuid <NET_UUID>
                      --ip <IP>
```

где

- `net_uuid` — идентификатор сети;
- `ip` — виртуальный IP-адрес для указанной сети.

## 21.4.4. Правила доступа

Списки контроля доступа ACL можно применять как к конкретному порту VM, так и на всю виртуальную локальную сеть.

### Примечание

Добавляйте ACL с помощью файла YAML, так как часто возникают ошибки при внесении данных

### ПРАВИЛА ДОСТУПА ДЛЯ ПОРТА VM

### Важно

Сначала загружайте ACL с разрешениями, затем с запретом. Неправильный порядок загрузки ACL приводит к блокировке трафика VM.  
ACL работают только для VLAN

1. Пример для настройки ACL для VM [acl\\_vm.yaml](#).

Файл описывает ACL `allow-rules`. Перечень списков контроля доступа включает:

- разрешение доступа к репозиторию 10.2.2.11;
- TCP-трафик до `ya.ru`;
- TCP-трафик до конкретного адреса или с конкретного адреса.

2. ACL для блокировки остального вида трафика содержатся в файле [deny\\_all.yaml](#).

3. Загрузите файл на Платформу.

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` — расположение YAML-файла.

При локальном расположении на АРМ пользователя введите путь до файла.

При расположении в сторонней репозитории укажите ссылку на данный файл.

4. Чтобы посмотреть список всех созданных ACL, введите

```
ipam acl list
```

5. Посмотреть подробную информацию о конкретном списке контроля доступа

```
ipam acl show --name <NAME>
```

где `name` — имя списка контроля доступа.

6. Обновить список контроля доступа

```
ipam acl update --name <NAME>
                --data <DATA>
```

где

- `name` — имя списка контроля доступа;
- `data` — обновленный перечень правил в формате JSON, входящих в файл `acl`.

### 7. Добавить списки контроля доступа для сетевого интерфейса

```
ipam iface acls add --net_uuid <NET_UUID>
                   --iface_uuid <IFACE_UUID>
                   --acls <ACLS>
```

где

- `net_uuid` — идентификатор сети;
- `iface_uuid` — идентификатор сетевого интерфейса;
- `acls` — перечень имен списков контроля доступа для указанной сети.

### 8. Удалить список контроля доступа с порта VM можно командой

```
ipam iface acls del --net_uuid <NET_UUID>
                   --iface_uuid <IFACE_UUID>
                   --acls <ACLS>
```

где

- `net_uuid` — идентификатор сети;
- `iface_uuid` — идентификатор сетевого интерфейса;
- `acls` — перечень имен списков контроля доступа для указанной сети.

## СПИСКИ КОНТРОЛЯ ДОСТУПА ACL ДЛЯ ВИРТУАЛЬНОЙ ЛОКАЛЬНОЙ СЕТИ

### Важно

Сначала загружайте ACL с разрешениями, затем с запретом. Неправильный порядок загрузки ACL приводит к блокировке трафика VM.  
ACL работают только для VLAN

### 1. Пример для настройки ACL для виртуальной сети [aclvlan.yaml](#).

Файл описывает ACL `allow-rules`. Они совпадают с ACL для порта VM, но в [aclvlan.yaml](#) заданы списки контроля доступа для черных списков, а для порта VM — только для белых.

В перечень ACL входит:

## Sharx Base 6.3. Руководство пользователя. Командная строка

- разрешение доступа к репозиторию 10.2.2.11;
- TCP-трафик до ya.ru;
- TCP-трафик до конкретного адреса или с конкретного адреса;
- ACL для черных списков.

2. Списки контроля доступа для блокировки остального вида трафика содержатся в файле [denyall.yaml](#).

3. Загрузите файл на Платформу.

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` — расположение YAML-файла.

При локальном расположении на APM пользователя введите путь до файла.

При расположении в стороннем репозитории укажите ссылку на данный файл.

4. Чтобы добавить ACL к виртуальной сети, введите

```
ipam network acls add --net_uuid <NET_UUID>  
                    --acls <ACLS>
```

где

- `net_uuid` — идентификатор сети;
- `acls` — перечень имен списков контроля доступа для указанной сети.

5. Удалить списки контроля доступа у виртуальной сети

```
ipam network acls del --net_uuid <NET_UUID>  
                    --acls <ACLS>
```

где

- `net_uuid` — идентификатор сети;
- `acls` — перечень имен списков контроля доступа для указанной сети.

6. Удалить списки контроля доступа с кластера **Sharx Base**

```
ipam acl del --name <NAME>
```

где `name` — имя списка контроля доступа.

---

### ПЕРЕЧЕНЬ ДОПОЛНИТЕЛЬНЫХ СПИСКОВ КОНТРОЛЯ ДОСТУПА ACL

1. ACL, позволяющий использовать ICMP

```
allow-ping:
- rule:
  # Разрешить ICMP (ip protocol number 1)
  eth_type: 0x800
  ip_proto: 1
  actions:
    allow: True
- rule:
  # Блокировать весь остальной трафик
  eth_type: 0x800
  actions:
    allow: False
```

### 2. ACL с блокировкой ICMP по критерию запрос/ответ

```
block-icmp-by-type:
- rule:
  # Блокировать ICMP Echo Reply
  eth_type: 0x800
  ip_proto: 1
  icmpv4_type: 0
  actions:
    allow: False
- rule:
  # Блокировать ICMP Echo Request
  eth_type: 0x800
  ip_proto: 1
  icmpv4_type: 8
  actions:
    allow: False
```

### 3. ACL для защиты от спуфинга (antispoofing)

```
antispoof_acl:
- rule:
  # Разрешить ARP с IP-адресом VM (IP antispoofing)
  dl_type: 0x806
  arp_spa: $VM_interface_IP
  actions:
    allow: True
- rule:
  # Разрешить ARP с MAC-адресом VM (MAC antispoofing)
  dl_type: 0x806
  arp_sha: $VM_interface_MAC
  actions:
    allow: True
- rule:
  # Блокировать все остальные ARP-пакеты
  dl_type: 0x806
  actions:
    allow: False
```

### 4. ACL для разрешения всего трафика

```
permit_all_acl:
- rule:
```

```
# Разрешить весь трафик
actions:
  allow: True
```

### 21.5. Правила размещения виртуальных машин

**Sharx Base** поддерживает гибкие правила выбора узлов при создании VM. Основные способы:

- указание конкретного узла `nodeName`;
- использование меток узлов `nodeSelector`;
- совместное размещение `affinity`, раздельное совмещение `antiAffinity` и предпочтительное размещение `colocation`.

#### Важно

Создавайте VM с помощью файлов YAML, так как часто возникают ошибки при внесении данных

#### 21.5.1. Выбор узла по идентификатору `nodeName`

В YAML-файле создания VM в разделе `data` задайте параметр `nodeName` — идентификатор узла, на который нужно спланировать ресурс

```
nodeName: <uuid_node_where_place_vms>
```

VM будет размещена на узле с указанным идентификатором.

#### 21.5.2. Использование меток узлов `nodeSelector`

#### Примечание

Подробная информация о метках приведена в статье [Работа с метками](#)

Позволяет указать пары `ключ-значение` для выбора узлов. Количество данных пар может быть не ограничено, но чаще всего используется только одна пара. VM будет размещена на узле, удовлетворяющем **всем** условиям.

Ниже рассмотрены случаи применения меток по умолчанию и пользовательских меток.

## Sharx Base 6.3. Руководство пользователя. Командная строка

### МЕТКА ПО УМОЛЧАНИЮ

По умолчанию каждый узел уже имеет метку `base`, которую можно применить в разделе `nodeSelector`.

Пример определения `nodeSelector` на основе метки `base`

```
nodeSelector:  
  key: base
```

В данном случае VM будет размещена на менее загруженном узле виртуального кластера.

Пример файла с заданным `nodeSelector` с меткой по умолчанию:

- для **Libvirt** `scheduler_request_nodeselector_libvirt.yaml`.
- для **РСХД** `scheduler_request_nodeselector_sp.yaml`.

### МЕТКИ ПОЛЬЗОВАТЕЛЯ

При вводе пользовательских меток VM будет размещена на узле, содержащем **все** метки.

Чтобы ввести правило выбора узла на основе меток, выполните следующие действия:

1. Просмотрите список меток, отданных во ВЦОД

```
scheduler labels ns show [--filter <FILTER>]
```

где `filter` — фильтрация меток. Возможные значения:

- `a` — all, все метки;
- `i` — ignored, игнорируемые метки;
- `u` — unignored, все метки, кроме игнорируемых.

#### **✗ Внимание**

Пользователи могут видеть метки только ВЦОД, в котором они находятся.  
При попытке запроса информации о других ВЦОД будет выдаваться ошибка.  
Исключение — ВЦОД управления

2. Просмотрите группы совместимых меток в виртуальном кластере

```
scheduler labels vcluster sharing --vcluster <VCLUSTER>
```

В результате выполнения команды будут показаны группы меток, которые можно использовать вместе для успешного планирования пользовательских ресурсов на узлы кластера.

3. В YAML-файл создания VM в раздел `spec` добавьте параметр `nodeSelector`, содержащий комбинацию совместимых меток, например

```
nodeSelector:  
  disktype: ssd  
  vcluster: test
```

VM будет размещена на узле, содержащем метки `disktype=ssd` и `vcluster=test`.

### 21.5.3. `nodeAffinity`, `vmAffinity` и `colocation`

Планировщик **Scheduler** позволяет управлять размещением виртуальных машин в кластере с помощью правил **совместного** `affinity`, **раздельного** `antiAffinity` и предпочтительного `colocation` размещения. Эти правила задаются в YAML-файлах создания VM и включают:

- `nodeAffinity` – ограничение по меткам узлов.
- `vmAffinity`, `antiAffinity` – совместное или раздельное размещение VM
- `colocation` – совместное размещение связанных компонентов. Позволяет задавать более сложные условия выбора узлов с операторами и весами.

#### ОБЩИЕ ТИПЫ СТРАТЕГИЙ ПЛАНИРОВАНИЯ

Для `nodeAffinity` и `vmAffinity` применяются две стратегии:

Стратегия	Описание
<code>requiredDuringSchedulingIgnoredDuringExecution</code>	<b>Жесткое правило.</b> Если ни один узел не соответствует, VM не будет размещена
<code>preferredDuringSchedulingIgnoredDuringExecution</code>	<b>Мягкое правило.</b> Если узлы не найдены, VM будет размещена на более подходящих узлах

#### NODEAFFINITY

Правило `nodeAffinity` позволяет ограничить, на каких узлах может выполняться виртуальная машина. `nodeAffinity` – аналог `nodeSelector`, но более гибкий за счет применения стратегий.

При использовании `requiredDuringSchedulingIgnoredDuringExecution` планировщик будет выбирать только узлы, строго удовлетворяющие условиям. В режиме

## Sharx Base 6.3. Руководство пользователя. Командная строка

`preferredDuringSchedulingIgnoredDuringExecution` задается предпочтение: узлы с нужными метками получают больший вес при оценке, но при отсутствии таких узлов VM все равно создастся.

Для селекторов меток узлов `nodeAffinity` поддерживает операторы:

- `In` — значение присутствует;
- `NotIn` — значение отсутствует;
- `Exist` — ключ существует;
- `DoesNotExist` — ключ отсутствует.

Например, можно указать

```
...
matchExpressions:
- key: disktype
  operator: In
  values:
  - ssd
...
```

и выбор будет выполняться **только** среди узлов с меткой `disktype=ssd`.



## Пример описания nodeAffinity из YAML-файла

```
...
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: az-name
                operator: NotIn
                values:
                  - az1
            - key: stage
                operator: DoesNotExist

          - key: disktype
                operator: In
                values:
                  - ssd

          - matchExpressions:
              - key: vcluster
                operator: In
                values:
                  - test
                  - test2

        preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 100
            preference:
              matchExpressions:
                - key: az-name
                  operator: In
                  values:
                    - az2

                - key: vcluster
                  operator: In
                  values:
                    - test

                - key: vcluster
                  operator: Exist

          - weight: 70
            preference:
              matchExpressions:
                - key: security
                  operator: In
                  values:
                    - hi_secure

          - weight: 10
            preference:
              matchExpressions:
                - key: vcluster
                  operator: In
                  values:
```

```
... - test
```

где возможные значения `operator`:

- `In` — значение присутствует;
- `NotIn` — значение отсутствует;
- `Exist` — ключ существует;
- `DoesNotExist` — ключ отсутствует.

`vmAffinity`, `antiAffinity`, `colocation`

Правила `vmAffinity`, `antiAffinity` и `colocation` позволяют учитывать **расположение других ВМ** при планировании:

- совместное размещение ВМ `vmAffinity`;
- раздельное размещение ВМ `antiAffinity`;
- размещение ВМ с учетом уже размещенных ресурсов `colocation`.

Типы стратегий описаны в разделе [выше](#).

СОВМЕСТНОЕ РАЗМЕЩЕНИЕ `vmAffinity`

`vmAffinity` определяет, что ВМ должна размещаться на том же узле, что и другие ВМ с нужными метками.

Пример применения правила представлен в файле YAML. Выберите файл в зависимости от типа хранилища.

**Libvirt**

[scheduler\\_request\\_colocation\\_libvirt.yaml](#)

в файле возможные значения `operator`:

- `In` — значение присутствует;
- `NotIn` — значение отсутствует;
- `Exist` — ключ существует;
- `DoesNotExist` — ключ отсутствует.

**РСХД**

[scheduler\\_request\\_colocation\\_sp.yaml](#)

в файле возможные значения `operator`:

- `In` — значение присутствует;
- `NotIn` — значение отсутствует;
- `Exist` — ключ существует;
- `DoesNotExist` — ключ отсутствует.

Разберем, как работает планировщик на основе данного примера.

### 1. `nodeSelector`

```
nodeSelector:  
  key: base  
  disktype: ssd  
  vcluster: test2
```

Это правило жестко ограничивает выбор узлов. Узел должен иметь все перечисленные метки: `key=base`, `disktype=ssd` и `vcluster=test2`.

Если ни один узел в кластере не соответствует этим условиям, VM не будет размещена.

### 2. `vmAffinity`

```
vmAffinity:  
  requiredDuringSchedulingIgnoredDuringExecution:  
    - labelSelector:  
      matchExpressions:  
        - key: role  
          operator: NotIn  
          values:  
            - web-db2
```

Жесткое правило. VM не может быть размещена на том же узле, где уже есть другие VM с меткой `role=web-db2`.

Далее

```
preferredDuringSchedulingIgnoredDuringExecution:  
  - weight: 100  
    vmAffinityTerm:  
      labelSelector:  
        matchExpressions:  
          - key: authtor  
            operator: In  
            values:  
              - dev
```

Мягкое предпочтение. VM желательно размещать рядом с другими VM, у которых метка `authtor=dev`. Если таких VM нет, это правило не мешает размещению.

## 3. Вывод

Учитывая правила, VM будет размещена следующим образом:

- планировщик сначала жестко отфильтрует узлы по `nodeSelector`. Узел должен иметь метки `key=base` и `disktype=ssd`, и `vcluster=test2`;
- затем он исключит узлы, где есть VM с меткой `role=web-db2`;
- среди оставшихся предпочтет те, где уже размещены VM с `authtor=dev`;
- если ни один узел не соответствует всем жестким условиям, VM **не будет размещена**.

### РАЗДЕЛЬНОЕ РАЗМЕЩЕНИЕ VM `antiAffinity`

`vmAntiAffinity` запрещает размещать VM вместе с другими VM с указанной меткой.

В примере ниже VM не будет размещена на узле, где есть VM с метками `authtor=dev` или `role=web-db2`.

```
...
affinity:

  vmAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
            - key: authtor
              operator: In
              values:
                - dev
            - key: role
              operator: In
              values:
                - web-db2
      ...
```

### ПРЕДПОЧТИТЕЛЬНОЕ РАЗМЕЩЕНИЕ РЯДОМ `colocation`

Пример предпочтительного размещения

```
apiVersion: v1
kind: api
metadata:
  plugin:
    scheduler:
      request: add
spec:
  vcluster: "имя виртуального кластера"
  kind: vm
```

```
name: "уникальное имя виртуальной машины"
labels: "role=analytics,env=dev"
descr: "описание виртуальной машины"
data:

  affinity:

    vmAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 100
          vmAffinityTerm:
            labelSelector:
              matchExpressions:
                - key: env
                  operator: In
                  values:
                    - dev
                - key: role
                  operator: In
                  values:
                    - analytics

    nodeSelector:
      key: base
      disktype: ssd
      vcluster: test

  vms:
    - name: "уникальное имя виртуальной машины"
      descr: "Разместить рядом с другими analytics"
      vnc_passwd: "пароль для доступа по VNC"
      storage:
        sp:
          uuid: "идентификатор тома"
      networks: ["идентификатор сети, которая подключается к VM"]
      vram: "объем виртуальной памяти"
      vcpu: "количество выделяемых виртуальных процессоров"
```

Поведение планировщика:

- **VM желательно** разместить рядом с другими VM с `env=dev` и `role=analytics`.
- Если таких VM рядом нет, данная VM все равно будет размещена.

### 21.6. Оптимизация ресурсов VM

Глобальные параметры переподписки устанавливаются администратором кластера и по умолчанию применяются к каждой VM. Подробная информация описана в [Руководстве администратора в командной строке](#).

При необходимости можно переопределить некоторые параметры переподписки индивидуально для VM.

## 21.6.1. Настройка переподписки ЦПУ

Переподписка ЦПУ позволяет выделить виртуальной машине больше ВЦПУ, чем физически доступно в кластере за счет разделения процессорного времени между несколькими ВЦПУ.

При установленных настройках [Переподписки ЦПУ](#) в кластере назначен глобальный коэффициент переподписки ЦПУ `default_cpu_overcommit_ratio`. Этот коэффициент применяется ко всем VM по умолчанию, но может быть переопределен для каждой виртуальной машины индивидуально.

При создании VM можно указать коэффициент переподписки `cpu_overcommit_ratio` в диапазоне 1-4.

### Пример создания VM с указанным коэффициентом переподписки ЦПУ

```
apiVersion: v1
kind: api
metadata:
  plugin:
    scheduler:
      request: add
spec:
  data:
    vms:
      - name: "vm"
        vcpu: 3
        cpu_overcommit_ratio: 2
    ...
```

Чем выше коэффициент переподписки ЦПУ, тем меньше доля ЦПУ на каждый ВЦПУ, то есть можно создать больше ВЦПУ.

### Пример

VM с `vcpu: 3` и `cpu_overcommit_ratio: 2`. Каждый ВЦПУ использует  $1 / 2 = 0.5$  физического ЦПУ. Таким образом VM будет потреблять  $3 \times 0.5 = 1.5$  ЦПУ

## 21.6.2. Настройка переподписки ОЗУ

Переподписка ОЗУ позволяет выделить VM больше виртуальной памяти, чем физически доступно в кластере, при условии, что в кластере настроена и включена [переподписка ОЗУ](#).

При установленных настройках [Переподписки ОЗУ](#) в кластере назначено глобальное значение толерантности VM к переподписке `default_overcommit_tolerance`. Этот коэффициент применяется ко всем VM по умолчанию, но может быть переопределен для каждой виртуальной машины индивидуально.

При создании VM можно указать толерантность VM к переподписке `overcommit_tolerance` в диапазоне от 0-3.

Правила работы коэффициента переподписки ОЗУ:

- VM с `overcommit_tolerance` больше 0 становится «донором памяти»;
- при достижении порога срабатывания балунинга VM начинает отдавать часть своей памяти другим VM;
- чем выше `overcommit_tolerance`, тем больше памяти может отдать VM;
- при `overcommit_tolerance: 0` VM не отдает свою память.

### Пример создания VM с указанной толерантностью к переподписке ОЗУ

```
apiVersion: v1
kind: api
metadata:
  plugin:
    scheduler:
      request: add
spec:
  data:
    vms:
      - name: "vm"
        vcpu: 3
        overcommit_tolerance: 2
      ...
```

## 21.7. Операции с виртуальными машинами

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД** или **Разработчик VM**

**Пользовательский ресурс** – физическая и логическая совокупность ресурсов, выделяемая на виртуальном кластере и содержащая виртуальную машину. Один пользовательский ресурс содержит одну VM.

### 21.7.1. Создать VM

Создание VM возможно двумя способами: с помощью YAML-файлов или через командную строку. Первый вариант предпочтительнее, так как он минимизирует вероятность ошибок.

## Sharx Base 6.3. Руководство пользователя. Командная строка

### СОЗДАТЬ ВМ С ПОМОЩЬЮ YAML-ФАЙЛА

1. Создайте файл формата YAML.
2. Опишите в файле все необходимые параметры и правила для создания ВМ.  
Подробная информация описана в статье [Правила размещения виртуальных машин](#).
3. При необходимости укажите IP-адрес из пула доступных адресов:

- a. Чтобы просмотреть доступные адреса, введите команду

```
ipam addr free list --uuid <UUID>  
                    [--full_output <FULL_OUTPUT>]
```

где

- `uuid` — идентификатор сети или подсети;
- `full_output` — отобразить весь список адресов. Без указания `full_output` по умолчанию отобразятся только три доступных адреса.

- b. Чтобы просмотреть занятые адреса, введите

```
ipam addr allocated list --uuid <UUID>
```

где `uuid` — идентификатор сети или подсети.

- c. Добавьте в YAML

```
networks: ["идентификатор сети"]  
ip_addresses: [{"\идентификатор сети\": \"IP-адрес из пула доступных адресов\"}]
```

4. При необходимости ограничьте трафик для сетевых интерфейсов

#### Пример для ВМ с одним сетевым интерфейсом

```
networks: ["идентификатор сети"]  
inbounds: ["лимит входящего трафика"]  
outbounds: ["лимит исходящего трафика"]
```

#### Пример для ВМ с двумя сетевыми интерфейсами

```
networks: ["идентификатор сети 1", "идентификатор сети 2"]  
inbounds: ["-", "лимит входящего трафика сети 2"]  
outbounds: ["лимит исходящего трафика сети 1", "-"]
```

Для ВМ с двумя сетевыми интерфейсами ограничения указываются в том же порядке, что и интерфейсы в `networks`, значение `"-"` означает отсутствие ограничения на сетевой интерфейс.

5. При необходимости определите индивидуальные параметры [переподписки ЦПУ и ОЗУ](#).
6. Сохраните файл.

### 7. Загрузите файл на Платформу.

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` — расположение YAML-файла.

При локальном расположении на АРМ пользователя введите путь до файла.

При расположении в стороннем репозитории укажите ссылку на данный файл.

### 8. После загрузки проверьте список всех созданных VM

```
scheduler request list --vcluster <VCLUSTER>
```

где `vcluster` — имя виртуального кластера.

#### Пример

Примеры файлов создания VM:

- для хранилища **Libvirt** с `nodeSelector`, указанием IP-адреса и ограничением на единственный сетевой интерфейс `scheduler_request_libvirt.yaml`.
- для хранилища **РСХД** с `nodeSelector`, указанием IP-адресов и ограничением только на второй сетевой интерфейс `scheduler_request_sp.yaml`.

#### СОЗДАТЬ VM ЧЕРЕЗ КОМАНДНУЮ СТРОКУ

Чтобы создать VM через командную строку, введите

```
scheduler request add --vcluster <VCLUSTER>  
                      --name <NAME>  
                      [--kind <KIND>]  
                      [--descr <DESCR>]  
                      [--labels <LABELS>]  
                      [--data <DATA>]  
                      [--type <common|linked_clone>]  
                      [--template <TEMPLATE>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `kind` — тип запроса. По умолчанию `vm`;
- `descr` — описание виртуальной машины;
- `labels` — метки VM. Задаются в формате `key=value`, разделяются пробелами;

- `data` — определение спецификаций. Возможные значения можно посмотреть в примерах YAML-файлов: [создание VM с Libvirt-хранилищем](#), [создание VM с РСХД-хранилищем](#);
- `type` — тип VM. Значение по умолчанию: `common`. Возможные значения:
  - `common` — VM создается стандартным способом, дисковая система формируется как полная копия хранилища источника;
  - `linked_clone` — VM создается по технологии связанных клонов, использует общий родительский виртуальный диск в режиме «только чтение» и хранит только свои собственные изменения;
- `template` — идентификатор шаблона VM.

### ✘ Внимание

Опция `type=linked_clone` может быть использована только для создания VM с Libvirt-хранилищем.

Прежде чем использовать опцию `template` при создании VM, необходимо прочитать статью [Шаблоны виртуальных машин](#)

## 21.7.2. Просмотреть VM

### 1. Просмотреть список всех созданных VM

```
scheduler request list --vcluster <VCLUSTER>
```

где `vcluster` — имя виртуального кластера.

### 2. Просмотреть информацию о конкретной VM

```
scheduler request show --vcluster <VCLUSTER>
                        --name <NAME>
                        [--kind <KIND>]
                        [--show_uuid <SHOW_UUID>]
                        [--metrics <METRICS>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `kind` — тип запроса. По умолчанию `vm`;
- `show_uuid` — показать идентификатор объекта;
- `metrics` — получить метрики VM. Возможные значения:
  - `yes` — включить получение метрик,

- `no` — отключить получение метрик.

### 3. Просмотреть статус VM

```
scheduler request status --vcluster <VCLUSTER>
                        [--kind <KIND>]
                        [--name <NAME>]
```

где

- `vcluster` — имя виртуального кластера;
- `kind` — тип запроса. По умолчанию `vm`;
- `name` — имя пользовательского ресурса, содержащего VM.

### 4. Просмотреть логины и роли пользователей, имеющих доступ к VM

```
scheduler request rights --vcluster <VCLUSTER>
                        --name <NAME>
                        [--kind <KIND>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `kind` — тип запроса. По умолчанию `vm`.

---

### 21.7.3. Обновить VM

```
scheduler request update --vcluster <VCLUSTER>
                        --name <NAME>
                        --data <DATA>
                        [--kind <KIND>]
                        [--descr <DESCR>]
                        [--labels <LABELS>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `data` — определение спецификаций. Возможные значения можно посмотреть в примерах YAML-файлов: [создание VM с Libvirt-хранилищем](#), [создание VM с РСХД-хранилищем](#);
- `kind` — тип запроса. По умолчанию `vm`;
- `descr` — описание виртуальной машины;
- `labels` — метки VM. Задаются в формате `key=value`, разделяются пробелами.

## 21.7.4. Изменить статус VM

```
scheduler request state --vcluster <VCLUSTER>
                        --name <NAME>
                        --state <START|STOP|REBOOT|SUSPEND|RESUME>
                        [--kind <KIND>]
                        [--vms <VMS>]
                        [--force <yes|no>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM, у которой изменяется статус;
- `state` — статус VM. Возможные значения:
  - `START` — запуск VM;
  - `STOP` — остановка VM;
  - `REBOOT` — перезагрузка VM;
  - `SUSPEND` — приостановка работы VM;
  - `RESUME` — возобновление работы VM;
- `kind` — тип запроса. Значение по умолчанию — `vm`;
- `vms` — список имен VM. Значение по умолчанию — `*`;
- `force` — опция, разрешающая принудительную небезопасную перезагрузку или остановку виртуальной машины. Значение по умолчанию — `no`. Возможные значения:
  - `yes` — принудительная перезагрузка или остановка VM разрешена. При этом несохраненные данные могут быть потеряны;
  - `no` — выполняется только штатная перезагрузка или остановка.

### ✘ Внимание

Опция `force` действует исключительно для состояний `STOP` и `REBOOT`

## 21.7.5. Назначить метку СКЗИ виртуальной машине

**Метка СКЗИ** или **криптографическая метка** применяется к VM, чтобы обозначить ее принадлежность к контуру криптографической защиты. Метка имеет ключ-значение `vm=crypto`. Для VM с меткой блокируются функции миграции, создания снимков и шаблона на основе этой VM. Данные меры исключают риск утечки защищаемых данных.

### ✘ Внимание

Криптографическую метку может назначить или снять только определенная роль — Администратор СКЗИ ВЦОД. Эта роль находится в конфигурации ВЦОД `cb_virt` и будет описана в следующих версиях документации

1. Чтобы назначить метку СКЗИ виртуальной машине, введите команду

```
scheduler request cips label add --vcluster <VCLUSTER>
                                --name <NAME>
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM.

2. Чтобы удалить метку СКЗИ с виртуальной машины, введите команду

```
scheduler request cips label del --vcluster <VCLUSTER>
                                --name <NAME>
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM.

### 21.7.6. Мигрировать VM

#### ⚠ Важно

Чтобы мигрировать VM, у которых есть снимок:

1. Выключите VM `state=stop`.
2. Выполните миграцию с параметром `mode=offline`

```
scheduler request migrate --vcluster <VCLUSTER>
                          --name <NAME>
                          --node <NODE>
                          --mode <MODE>
                          [--kind <KIND>]
```

где

- `vcluster` — имя виртуального кластера;

- `name` — имя пользовательского ресурса, внутри которого находится мигрируемая VM;
- `node` — идентификатор целевого узла;
- `mode` — режим миграции. Возможные значения:
  - `live` — миграция без прекращения работы VM;
  - `offline` — миграция с отключением VM;
- `kind` — тип запроса. Значение по умолчанию — `vm`.

### **Внимание**

Миграция VM с меткой СКЗИ невозможна

### 21.7.7. Удалить VM

Чтобы удалить VM, введите

```
scheduler request del --vcluster <VCLUSTER>
                    --name <NAME>
                    [--kind <KIND>]
                    [--backup <yes|no>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `kind` — тип запроса. По умолчанию `vm`;
- `backup` — создать снимок удаляемой VM. Снимок удалится через 1 день. Возможные значения `backup` — `yes`, `no`.

### 21.8. Запустить виртуальные машины

#### **Примечание**

Действия выполняются пользователем с ролью **Администратор ВЦОД** или **Разработчик VM**

Чтобы запустить VM, выполните команду

```
scheduler request state --vcluster <VCLUSTER>
                    --name <NAME>
```

```
--state START  
[--kind <KIND>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM, у которой изменяется статус;
- `state` — статус VM. Возможные значения:
  - `START` — запуск VM;
  - `STOP` — остановка VM;
  - `REBOOT` — перезагрузка VM;
  - `SUSPEND` — приостановка работы VM;
  - `RESUME` — возобновление работы VM;
- `kind` — тип запроса. Значение по умолчанию — `vm`.

### 21.9. Снимки VM и ТОМОВ

#### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД** или **Разработчик VM**

**Снимок** — моментальный снимок тома или VM. Копия на уровне блоков физических или виртуальных систем, выполненная без остановки системных служб. Включает в себя структуру директорий, файлов и информацию о состоянии системы на фиксированный момент времени. Снимок не является резервной копией, применяется или используется как временный источник для создания согласованных резервных копий.

#### Важно

При удалении виртуальной машины снимки удаляются вместе с ней.

#### 21.9.1. Снимки VM

#### Внимание

У VM, развернутой с использованием Libvirt-тома, может быть не более одного снимка. Создание снимка VM с [меткой СКЗИ](#) невозможно

## 1. Чтобы создать снимок VM, введите

```
scheduler request snapshot add --vcluster <VCLUSTER>
                                --name <NAME>
                                --snapshot_name <SNAPSHOT_NAME>
                                [--kind <KIND>]
                                [--vms <VMS>]
                                [--descr <DESCR>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `snapshot_name` — имя снимка;
- `kind` — тип запроса. Значение по умолчанию — `vm`;
- `vms` — список имен виртуальных машин. Значение по умолчанию — `*`;
- `descr` — описание снимка.

## 2. Просмотреть список снимков VM

```
scheduler request snapshot list --vcluster <VCLUSTER>
                                --name <NAME>
                                [--kind <KIND>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `kind` — тип запроса. Значение по умолчанию — `vm`.

## 3. Просмотреть информацию о конкретном снимке

```
scheduler request snapshot show --name <NAME>
                                --vcluster <VCLUSTER>
                                --uuid <UUID>
                                [--kind <KIND>]
```

где

- `name` — имя пользовательского ресурса, содержащего VM;
- `vcluster` — имя виртуального кластера;
- `uuid` — идентификатор снимка;
- `kind` — тип запроса. Значение по умолчанию — `vm`.

## 4. Чтобы восстановить VM из снимка, введите

```
scheduler request snapshot revert --vcluster <VCLUSTER>
                                   --name <NAME>
                                   --uuid <UUID>
                                   [--kind <KIND>]
```

```
[--vms <VMS>]
[--force <yes|no>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `uuid` — идентификатор снимка;
- `kind` — тип запроса. Значение по умолчанию — `vm`;
- `vms` — список имен виртуальных машин. Значение по умолчанию — `*`;
- `force` — опция, разрешающая восстановление виртуальной машины из снимка NFS, который не является последним в цепочке.

Возможные значения:

- `yes` — восстановление из произвольного снимка разрешено. При этом все снимки, созданные позже выбранного, безвозвратно удаляются;
- `no` — значение по умолчанию. Восстановление возможно только из последнего снимка в цепочке. Если указан любой другой снимок, операция завершается ошибкой.

### Важно

Установка `force=yes` приводит к необратимому удалению всех снимков, созданных после выбранного

## 5. Удалить снимок VM

```
scheduler request snapshot del --name <NAME>
                                --vcluster <VCLUSTER>
                                --uuid <UUID>
                                [--kind <KIND>]
```

где

- `name` — имя пользовательского ресурса, содержащего VM;
- `vcluster` — имя виртуального кластера;
- `uuid` — идентификатор снимка;
- `kind` — тип запроса. Значение по умолчанию — `vm`.

## 21.9.2. Снимки томов Libvirt

1. Чтобы создать снимок тома Libvirt, введите команду

```
storage libvirt snapshot add --vm_name <VM_NAME>
                             --vcluster_name <VCLUSTER_NAME>
                             [--uuid <UUID>]
                             [--name <NAME>]
                             [--disks <DISKS>]
                             [--descr <DESCR>]
```

где

- `vm_name` — имя ВМ;
- `vcluster_name` — имя виртуального кластера;
- `uuid` — пользовательский идентификатор снимка. Он будет присвоен снимку вместо генерируемого системой;
- `name` — имя снимка;
- `disks` — список имен томов для снимка. При отсутствии параметра `disks` выполнится снимок всех томов, прикрепленных к данной ВМ;
- `descr` — описание снимка.

### 2. Просмотреть подробную информацию о конкретном снимке ВМ

```
storage libvirt snapshot show --vm_name <VM_NAME>
                              --vcluster_name <VCLUSTER_NAME>
                              [--name <NAME>]
```

где

- `vm_name` — имя ВМ;
- `vcluster_name` — имя виртуального кластера;
- `name` — имя снимка.

### 3. Чтобы удалить снимок, введите команду

```
storage libvirt snapshot del --vm_name <VM_NAME>
                              --vcluster_name <VCLUSTER_NAME>
                              [--name <NAME>]
```

где

- `vm_name` — имя ВМ;
- `vcluster_name` — имя виртуального кластера;
- `name` — имя снимка.

---

## 21.9.3. Снимки томов РСХД

### 1. Чтобы создать снимок тома, введите команду

```
storage sp snapshot add --name <NAME>
                        --volume_uuid <VOLUME_UUID>
                        [--descr <DESCR>]
```

где

- `name` — имя снимка;
- `volume_uuid` — идентификатор тома;
- `descr` — описание снимка.

### 2. Просмотреть список всех снимков в пределах тома

```
storage sp snapshot list [--volume_uuid <VOLUME_UUID>]
```

где `volume_uuid` — идентификатор тома.

### 3. Просмотреть подробную информацию о конкретном снимке

```
storage sp snapshot show --uuid <UUID>
```

где `uuid` — идентификатор снимка.

### 4. Обновить параметры снимка

```
storage sp snapshot update --uuid <UUID>
                           [--name <NAME>]
                           [--descr <DESCR>]
                           [--delete_after <DELETE_AFTER>]
```

где

- `uuid` — идентификатор снимка;
- `name` — имя снимка;
- `descr` — описание снимка;
- `delete_after` — время отсрочки удаления снимков, указывается в секундах относительно текущего времени. При отсутствии параметра снимок автоматически удаляться не будет.

### 5. Удалить снимок

```
storage sp snapshot del --uuid <UUID>
```

где `uuid` — идентификатор снимка.

---

## 21.9.4. Снимки томов NFS

## УПРАВЛЕНИЕ СНИМКАМИ ТОМОВ NFS

### 1. Чтобы создать снимок тома, введите команду

```
storage nfs snapshot add --name <NAME>
                        --vm_name <VM_NAME>
                        --vcluster_name <VCLUSTER_NAME>
                        [--uuid <UUID>]
                        [--descr <DESCR>]
                        [--disks <DISKS>]
```

где

- `name` — имя создаваемого снимка;
- `vm_name` — имя VM;
- `vcluster_name` — имя виртуального кластера;
- `uuid` — пользовательский идентификатор снимка. Он будет присвоен снимку вместо генерируемого системой;
- `descr` — описание снимка;
- `disks` — список имен томов для снимка. При отсутствии параметра `disks` выполнится снимок всех томов, прикрепленных к данной VM.

### 2. Просмотр списка всех снимков в пределах виртуального кластера

```
storage nfs snapshot list --vcluster_name <VCLUSTER_NAME>
                          [--request_name <REQUEST_NAME>]
                          [--vm_name <VM_NAME>]
```

где

- `vcluster_name` — имя виртуального кластера;
- `request_name` — имя пользовательского ресурса, содержащего VM. Значение по умолчанию — `*`;
- `vm_name` — имя VM. Без `request_name` не используется из-за вложенности VM в пользовательский ресурс.

Если не указать `request_name` и `vm_name`, то система отобразит снимки всех VM указанного виртуального кластера.

Если указать `request_name` с `vm_name` или без него, то система отобразит снимки одной конкретной VM внутри указанного пользовательского ресурса.

### 3. Просмотр подробной информации о конкретном снимке

```
storage nfs snapshot show --uuid <UUID>
```

где `uuid` — идентификатор снимка.

### 4. Удалить снимок

```
storage nfs snapshot del --uuid <UUID>
```

где `uuid` — идентификатор снимка.

### ВОССТАНОВЛЕНИЕ ВМ ИЗ СНИМКА

Чтобы восстановить ВМ с хранилищем NFS из снимка, введите

```
scheduler request snapshot revert --vcluster <VCLUSTER>  
                                --name <NAME>  
                                --uuid <UUID>  
                                [--kind <KIND>]  
                                [--vms <VMS>]  
                                [--force <yes|no>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя ВМ;
- `uuid` — идентификатор снимка;
- `kind` — тип запроса. Значение по умолчанию — `vm`;
- `vms` — список имен виртуальных машин. Значение по умолчанию — `*`;
- `force` — опция, разрешающая восстановление виртуальной машины из снимка NFS, который не является последним в цепочке.

Возможные значения:

- `yes` — восстановление из произвольного снимка разрешено. При этом все снимки, созданные позже выбранного, безвозвратно удаляются;
- `no` — значение по умолчанию. Восстановление возможно только из последнего снимка в цепочке. Если указан любой другой снимок, операция завершается ошибкой.

#### **Важно**

Установка `--force=yes` приводит к необратимому удалению всех снимков, созданных после выбранного

## 21.10. Шаблоны виртуальных машин

**Шаблон виртуальной машины (ВМ)** — конфигурационный файл, определяющий параметры виртуальной машины (ЦПУ, ОЗУ, хранилище, сетевые настройки и др.), необходимые для её быстрого развертывания. Шаблон служит заготовкой для создания идентичных виртуальных машин.

## СОЗДАТЬ ШАБЛОН ВМ

1. Создать шаблон ВМ можно аналогично процессу, описанному в статье [Операции с виртуальными машинами. Создать ВМ через командную строку](#).

```
scheduler vm template add --name <NAME>
                          --vcpu <VCPU>
                          --vram <VRAM>
                          --storage <STORAGE>
                          --nodeSelector <NODESELECTOR>
                          [--descr <DESCR>]
                          [--labels <LABELS>]
                          [--ip_addresses <IP_ADDRESSES>]
                          [--cpu_overcommit_ratio <CPU_OVERCOMMIT_RATIO>]
                          [--overcommit_tolerance <OVERCOMMIT_TOLERANCE>]
                          [--networks <NETWORKS>]
                          [--inbounds <INBOUNDS>]
                          [--outbounds <OUTBOUNDS>]
                          [--affinity <AFFINITY>]
                          [--type <common|linked_clone>]
                          [--path_to <PATH_TO>]
```

где

- `name` — имя шаблона;
- `vcpu` — количество ВЦПУ;
- `vram` — объем оперативной памяти в мегабайтах и гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `storage` — тип хранилища;
- `nodeSelector` — метка или список меток для размещения ВМ;
- `descr` — описание шаблона ВМ;
- `labels` — метка или список меток ВМ. Метки задаются в формате `key=value`, разделяются пробелами;
- `ip_addresses` — IP-адрес или список IP-адресов из пула доступных адресов;
- `cpu_overcommit_ratio` — коэффициент переподписки ЦПУ в кластере по умолчанию. Возможные значения от `1` до `4`;
- `overcommit_tolerance` — толерантность ВМ к переподписке. Возможные значения от `0` до `3`;
- `networks` — идентификатор сети или список идентификаторов;
- `inbounds` — лимит входящего трафика;
- `outbounds` — лимит исходящего трафика;
- `affinity` — правила размещения ВМ;
- `type` — тип ВМ. Значение по умолчанию: `common`. Возможные значения:

## Sharx Base 6.3. Руководство пользователя. Командная строка

- `common` — VM создается стандартным способом, дисковая система формируется как полная хранилища источника;
- `linked_clone` — VM создается по технологии связанных клонов, использует общий родительский виртуальный диск в режиме «только чтение» и хранит только свои собственные изменения;
- `path_to` — размещение шаблона VM в иерархии директорий.

### ✘ Внимание

Технология связанных клонов (`type=linked_clone`) может быть использована только при создании VM с Libvirt-хранилищем.

При указании в `nodeSelector` меток, игнорируемых в кластере, ВЦОД или виртуальном кластере, будет выведена ошибка. Подробности можно изучить в статье [Правила размещения виртуальных машин](#)

2. Создать шаблон VM из уже существующей виртуальной машины можно с помощью команды

```
scheduler vm template from --vm-uuid <VM_UUID>
```

где `vm-uuid` — идентификатор виртуальной машины.

Создать шаблон VM с помощью YAML-файла

1. Шаблон VM можно создать простым перечислением параметров, аналогично процессу, описанному в статье [Операции с виртуальными машинами. Создать VM с помощью YAML-файла](#).

## Пример

```
apiVersion: v1
kind: api
metadata:
  plugin:
    scheduler:
      vm:
        template: add
spec:
  name: test_vm_sp
  descr: "VM test"
  vram: "3GB"
  vcpu: 1
  storage:
    nfs:
      disks:
        - "1fed2a97-3c90-465b-93fd-0ff8ca38f2a1"
        - "f4bdbdf5-a586-4120-9b27-b5b9216ca501"
      buses:
        1fed2a97-3c90-465b-93fd-0ff8ca38f2a1: "sata"
  labels: "name=test"
  ip_addresses: "{ \"6f478ba9-9ecc-4435-b562-2a910bc0340b\": \"10.3.31.38\"}"
  cpu_overcommit_ratio: 2
  overcommit_tolerance: 2
  networks:
    - "6f478ba9-9ecc-4435-b562-2a910bc0340b"
  inbounds: ["14980"]
  outbounds: ["24048"]
  nodeSelector:
    key: base
```

2. Шаблон VM можно создать, скопировав параметры существующей виртуальной машины.

## Пример

```
apiVersion: v1
kind: api
metadata:
  plugin:
    scheduler:
      vm:
        template: from
spec:
  vm_uuid: 1f108dc6-a429-6fae-8d8e-5439abff52e6
```

### ПРОСМОТРЕТЬ ШАБЛОНЫ VM

1. Просмотреть список всех шаблонов VM

```
scheduler vm template list
```

2. Просмотреть информацию о конкретном шаблоне VM

```
scheduler vm template show --uuid <UUID>
```

где `uuid` — идентификатор шаблона VM.

### ИЗМЕНИТЬ ШАБЛОН VM

```
scheduler vm template update --uuid <UUID>
    [--name <NAME>]
    [--descr <DESCR>]
    [--vcpu <VCPU>]
    [--vram <VRAM>]
    [--storage <STORAGE>]
    [--labels <LABELS>]
    [--ip_addresses <IP_ADDRESSES>]
    [--cpu_overcommit_ratio <CPU_OVERCOMMIT_RATIO>]
    [--overcommit_tolerance <OVERCOMMIT_TOLERANCE>]
    [--networks <NETWORKS>]
    [--inbounds <INBOUNDS>]
    [--outbounds <OUTBOUNDS>]
    [--nodeSelector <NODESELECTOR>]
    [--affinity <AFFINITY>]
    [--type <common|linked_clone>]
```

где

- `uuid` — идентификатор шаблона VM;
- `name` — имя шаблона;
- `descr` — описание шаблона VM;
- `vcpu` — количество ВЦПУ;
- `vram` — объем оперативной памяти в мегабайтах и гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `storage` — тип хранилища;
- `labels` — метка или список меток VM. Метки задаются в формате `key=value`, разделяются пробелами;
- `ip_addresses` — IP-адрес или список IP-адресов из пула доступных адресов;
- `cpu_overcommit_ratio` — коэффициент переподписки ЦПУ в кластере по умолчанию. Возможные значения от `1` до `4`;
- `overcommit_tolerance` — толерантность VM к переподписке. Возможные значения от `0` до `3`;
- `networks` — идентификатор сети или список идентификаторов;
- `inbounds` — лимит входящего трафика;
- `outbounds` — лимит исходящего трафика;

- `nodeSelector` — метка или список меток для размещения VM;
- `affinity` — правила размещения VM;
- `type` — тип VM. Значение по умолчанию: `common`. Возможные значения:
  - `common` — VM создается стандартным способом, дисковая система формируется как полная хранилища источника;
  - `linked_clone` — VM создается по технологии связанных клонов, использует общий родительский виртуальный диск в режиме «только чтение» и хранит только свои собственные изменения.

### **Внимание**

Технология связанных клонов (`type=linked_clone`) может быть использована только при создании VM с Libvirt-хранилищем

### УДАЛИТЬ ШАБЛОН VM

```
scheduler vm template del --uuid <UUID>
```

где `uuid` — идентификатор шаблона VM.

### ВОССТАНОВИТЬ ВИРТУАЛЬНУЮ МАШИНУ ИЗ ШАБЛОНА VM

Процесс восстановления виртуальной машины из шаблона VM подробно описан в статье [Восстановить виртуальную машину](#).

## 21.11. Резервные копии виртуальных машин

### **Примечание**

Действия выполняются пользователем с ролью **Администратор ВЦОД** или **Разработчик VM**

Том с данными резервной копии VM находится в пуле, которому принадлежат исходные тома.

XML-данные резервной копии хранятся в БД и содержат полное описание конфигурации виртуальной машины.

Команды для резервного копирования:

1. Чтобы создать резервную копию диска, введите в командной строке

```
storage libvirt backup add --vm_name <VM_NAME>  
                        [--sc_node <SC_NODE>]
```

```
[--disks <DISKS>]
[--timeout <TIMEOUT>]
```

где

- `vm_name` — имя VM;
- `sc_node` — идентификатор узла. По умолчанию используется UUID текущего узла;
- `disks` — список имен томов для снимка. При отсутствии параметра `disks` выполнится снимок всех томов, прикрепленных к данной VM;
- `timeout` — таймаут для резервного копирования, указывается в секундах без единиц измерения. Значение по умолчанию — 300.

### 2. Просмотреть список всех **выполняемых** резервных копирований

#### **Внимание**

Данная команда доступна только Администратору ВЦОД

```
storage libvirt backup list [--sc_node <SC_NODE>]
```

где `sc_node` — идентификатор узла. По умолчанию используется UUID текущего узла.

После успешного выполнения резервного копирования информация об операции автоматически удаляется из перечня и не отображается в результате данной команды.

### 3. Просмотреть подробную информации о статусе выполняемой операции

```
storage libvirt backup show [--name <VM_NAME>]
                             [--sc_node <SC_NODE>]
```

где

- `name` — имя VM;
- `sc_node` — идентификатор узла. По умолчанию используется UUID текущего узла. Введите параметр `sc_node`, если запрос производится не с того узла, на котором находится запрашиваемая VM.

После успешного выполнения резервного копирования информация об операции автоматически удаляется из перечня и не отображается в результате данной команды.

### 4. В случае некорректной отработки операции резервного копирования можно удалить эту операцию из истории

```
storage libvirt backup del [--name <VM_NAME>]
                             [--sc_node <SC_NODE>]
```

где

- `name` — имя VM;

- `sc_node` — идентификатор узла. По умолчанию используется UUID текущего узла.

Затем повторите резервное копирование, указав корректные параметры.

### 5. Получить XML-описание резервной копии VM

```
storage libvirt backup lookup --name <NAME>
```

где `name` — имя виртуальной машины.

#### Пример

Пример XML-описания резервной копии VM [domain\\_backup.xml](#)

## 21.12. Восстановить виртуальную машину

### Примечание

Действия выполняются пользователем с ролью **Администратор ВЦОД** или **Разработчик VM**

### 21.12.1. Восстановить VM из резервной копии

Восстановление VM возможно при наличии резервной копии.

Команды резервного копирования подробно описаны в статье [Резервные копии VM](#).

Механизм восстановления VM из резервной копии состоит из шагов, описанных ниже.

**ПОДГОТОВИТЬ РЕЗЕРВНУЮ КОПИЮ**

1. Создайте резервную копию VM.

2. Если резервное копирование прошло успешно:

- Информация о совершенной операции автоматически удаляется из перечня и не отображается при выполнении команд `storage libvirt backup list` и `storage libvirt backup show`.
- В пуле хранилища появляются тома с данными резервной копии.
- Описание резервной копии в виде XML можно посмотреть командой

```
storage libvirt backup lookup --name <NAME>
```

где `name` — имя VM.

3. Если резервное копирование прошло неуспешно:

## Sharx Base 6.3. Руководство пользователя. Командная строка

- a. При выполнении команд `storage libvirt backup list` и `storage libvirt backup show` висит информация об ошибке.
- b. Удалите эту операцию из истории.
- c. Затем повторите резервное копирование, указав корректные параметры.

### ВОССТАНОВИТЬ ВМ

#### Примечание

Том с данными резервной копии ВМ находится в пуле, которому принадлежат исходные тома. XML-данные резервной копии хранятся в БД и содержат полное описание конфигурации виртуальной машины

Чтобы восстановить ВМ из резервной копии, необходимо:

1. Получить XML-описание резервной копии ВМ

```
storage libvirt backup lookup --name <NAME>
```

где `name` — имя виртуальной машины.

#### Пример

Пример XML-описания резервной копии ВМ [domain\\_backup.xml](#)

2. Создать YAML-файл для создания ВМ с параметрами из XML.

#### Пример

YAML-файл создания ВМ [request\\_libvirt\\_recovery.yaml](#)

3. Загрузить файл на Платформу

Чтобы загрузить файлы в Sharx Base, введите команду

```
resource --spec <SPEC>
```

где `spec` — расположение YAML-файла.

При локальном расположении на АРМ пользователя введите путь до файла.

При расположении в стороннем репозитории укажите ссылку на данный файл.

4. После загрузки проверить список всех созданных ВМ

```
scheduler request list --vcluster <VCLUSTER>
```

где `vcluster` — имя виртуального кластера.

### 5. Подробная информация о статусе конкретной VM

```
scheduler request show --name <NAME>
                        --vcluster <VCLUSTER>
```

где

- `name` — имя пользовательского ресурса, содержащего VM;
- `vcluster` — имя виртуального кластера.

В результате действий будет создана виртуальная машина, аналогичная по данным и конфигурации восстанавливаемой VM.

### 21.12.2. Восстановить VM из шаблона VM

Восстановление VM возможно при наличии шаблона VM.

Процесс создания шаблона VM подробно описан в статье [Шаблоны виртуальных машин](#).

Чтобы восстановить виртуальную машину из шаблона VM, необходимо:

```
scheduler request snapshot revert --vcluster <VCLUSTER>
                                   --name <NAME>
                                   --uuid <UUID>
                                   [--kind <KIND>]
                                   [--vms <VMS>]
                                   [--force <yes|no>]
```

где

- `vcluster` — имя виртуального кластера;
- `name` — имя пользовательского ресурса, содержащего VM;
- `uuid` — идентификатор снимка;
- `kind` — тип запроса. Значение по умолчанию — `vm`;
- `vms` — список имен виртуальных машин. Значение по умолчанию — `*`;
- `force` — опция, разрешающая восстановление виртуальной машины из снимка NFS, который не является последним в цепочке. Возможные значения:
  - `yes` — восстановление из произвольного снимка разрешено. При этом все снимки, созданные позже выбранного, безвозвратно удаляются;
  - `no` — значение по умолчанию. Восстановление возможно только из последнего снимка в цепочке. Если указан любой другой снимок, операция завершается ошибкой.

### Важно

Установка `--force=yes` приводит к необратимому удалению всех снимков, созданных после выбранного

## 22. Перевести узлы в сервисный режим

**Сервисное обслуживание** позволяет временно отключить отдельные узлы от процесса планирования ресурсов. Эта функция полезна при проведении технических работ с оборудованием.

При активации сервисного режима каждому узлу присваивается специальная метка `system_drain_node=yes`. После этого узел перестает быть доступным для размещения на нем новых ресурсов.

### 22.1. Перевести узлы в сервисный режим

Чтобы перевести узлы в сервисный режим, выполните команду

```
scheduler drain add --nodes <NODES>
```

где `nodes` — список идентификаторов узлов, которые требуется вывести из эксплуатации.

После выполнения команды:

- Узлам автоматически присваивается метка `system_drain_node=yes`.
- Узлы исключаются из размещения новых ресурсов.

### 22.2. Отключить сервисный режим

```
scheduler drain del --nodes <NODES>
```

После выполнения команда метка `system_drain_node` удаляется, и узлы возвращаются в общее планирование.

## 23. Мониторинг

### 23.1. Общие сведения

Кластер Sharx Base оснащен внутренней системой мониторинга, которая собирает метрики с узлов кластера и позволяет диагностировать состояние системы.

Внутренний мониторинг является штатной частью платформы и работает на основе **экспортеров** – сервисов, установленных на каждом узле.

Собранные метрики хранятся ограниченное время.

Для долговременного хранения и расширенной визуализации может использоваться дополнительное решение – [Sharx ProView](#).

### 23.2. Внутренний мониторинг Sharx Base

Внутренний мониторинг позволяет:

- получать метрики компонентов кластера;
- проверять состояние узлов и виртуальных машин;
- работать с шаблонами отображения метрик.

**Экспортеры** – сервисы, которые собирают метрики на узлах и предоставляют к ним доступ на определенном HTTP-порту.

В документации используются два типа имен экспортеров:

- имя сервиса на узле используется для проверки установленного сервиса или службы на уровне ОС;
- имя экспортера в Sharx Base используется в шаблонах и выводах команд в CLI.

Имена сервисов на узлах отличаются от имен экспортеров в Sharx Base.

Таблица – Соответствие имен экспортеров на узлах и в Sharx Base.

Имя сервиса на узле	Имя экспортера в Sharx Base	Порт	Назначение
<code>sdc-cgroup_exporter</code>	<code>cgroup_exporter</code>	9198	Метрики ограничений и использования ресурсов системными группами
<code>sdc-disk_exporter</code>	<code>disks_exporter_sharx</code>	9196	Метрики дисков
<code>sdc-domain-exporter</code>	<code>domain_exporter</code>	9190	Метрики состояния виртуальных машин

Имя сервиса на узле	Имя экспортера в Sharx Base	Порт	Назначение
<code>sdc-ipmi_exporter</code>	<code>ipmi_exporter_sharx</code>	9195	Метрики состояния аппаратного обеспечения
<code>sdc-node_exporter</code>	<code>node_exporter_sharx</code>	9191	Метрики общего состояния узлов кластера
<code>sdc-sp_exporter</code>	<code>sp_exporter_sharx</code>	9197	Метрики хранилища РСХД

Предустановки для мониторинга выполняются сотрудниками технической поддержки.

Настройка экспортеров выполняется **Администратором кластера**.

**Администратор ВЦОД** может работать с шаблонами, которые используются для извлечения информации о метриках из экспортера. Доступные действия **Администратора ВЦОД** описаны ниже:

#### 1. Просмотреть список созданных шаблонов

```
metrics template list
```

#### 2. Просмотреть подробную информацию о конкретном шаблоне

```
metrics template show --name <NAME>
```

где `name` — имя шаблона.

#### 3. Отобразить метрики по фильтрам, указанным в шаблоне

```
metrics template execute --name <NAME>
```

где `name` — имя шаблона.

### 23.3. Внешний мониторинг. Sharx ProView

**Sharx ProView** — это дополнительное решение для сбора метрик платформы виртуализации Sharx Base, которое позволяет:

- хранить метрики неограниченное время;
- анализировать производительность нескольких установок Sharx Base в едином интерфейсе;
- использовать преднастроенные дашборды для отображения информации о ресурсах.

Внешний мониторинг на базе Sharx ProView не входит в стандартную поставку платформы и настраивается отдельно по запросу заказчика. Для получения подробной информации и установки обратитесь в [техническую поддержку](#) ООО «Шаркс ДЦ».

### **Примечание**

Внутренний мониторинг при этом остается активным. Внешнее решение является дополнением для долговременного хранения и расширенной визуализации

## 24. Система автоматизации процедур checker

### 24.1. Checker

**Checker** – это плагин для автоматизации типовых операций в кластере и ВЦОД. Он позволяет выполнять последовательности действий (процедуры) как вручную, так и по расписанию, отслеживать их выполнение и обрабатывать ошибки.

Администратор ВЦОД не разрабатывает процедуры и работает только с готовыми, которые создал и делегировал Администратор кластера в данный ВЦОД.

#### 24.1.1. Ключевые понятия

**Процедура** – YAML-файл с набором шагов, созданный администратором кластера и доступный во ВЦОД после делегирования.

**Шаг** – отдельный этап процедуры, содержащий условия и действия. Например, перезагрузка VM, проверка статуса

**Делегирование** – назначение процедуры для использования в определенных ВЦОД.

**Планирование** – настройка автоматического выполнения процедуры по расписанию.

#### 24.1.2. Как работает Checker?

1. Администратор кластера создает процедуру и делегирует ее в определенный ВЦОД.
2. Администратор ВЦОД видит эту процедуру в списке доступных через команды планирования или запуска.
3. Администратор ВЦОД может:

- запустить процедуру вручную;
- запланировать ее автоматическое выполнение по расписанию;
- остановить выполняющуюся процедуру;
- просмотреть статус и логи выполнения.

4. После завершения процедуры статус хранится 90 дней, затем удаляется автоматически.

### 24.1.3. Доступные действия

Все команды выполняются в контексте определенного ВЦОД.

1. Просмотр доступных процедур.
2. Планирование процедуры.
3. Просмотр запланированных задач.
4. Удаление задачи из расписания.
5. Ручной запуск процедуры.
6. Остановка выполняющейся процедуры.
7. Просмотр статусов выполнения.

### 24.1.4. Дополнительная информация

1. Подробное описание команд и примеры их использования описаны в статье [Управление процедурами через CLI](#).
2. [Примеры процедур](#). Готовые сценарии для типовых задач.

#### Примечание

Администратор ВЦОД не может создавать, изменять или удалять процедуры, — эти действия выполняет только администратор кластера. Если вам нужна новая процедура, обратитесь к нему

## 24.2. Управление процедурами

Ключевые понятия описаны в статье [Checker](#).

В статье описано управление процедурами в рамках ВЦОД:

- [Просмотр определения процедуры](#).
- [Планирование выполнения по расписанию](#).
- [Ручной запуск процедуры](#).

- [Остановка выполняющейся процедуры.](#)
- [Отслеживание статуса выполнения.](#)

### 24.2.1. Просмотреть определение процедуры

Чтобы просмотреть определение процедуры, введите

```
checker procedure show --name <NAME>
                        [--json <no|yes>]
```

где

- `name` — имя обновляемой процедуры;
- `json` — формат возвращения ответа. Возможные значения:
  - `"no"` — ответ возвращается в человекочитаемом формате. Значение по умолчанию,
  - `"yes"` — ответ возвращается в формате JSON.

Команда поможет узнать, какие переменные `vars` требует процедура, и есть ли у них значения по умолчанию.

### 24.2.2. Планирование выполнения процедуры

Планирование процедур по расписанию доступно как для кластера во ВЦОД управления, так и для ВЦОД. Администратор ВЦОД может планировать только те процедуры, которые делегированы в его ВЦОД. Процедуры уровня кластера, например, фенсинг узлов, планируются администратором кластера и в данном руководстве не рассматриваются.

1. Чтобы процедура выполнялась автоматически по расписанию, введите команду

```
checker procedure schedule add --name <NAME>
                               [--vars '{"var": "value"}']
                               --cron <cron>
```

где

- `name` — имя процедуры;
- `vars` — переменная процедуры. Имеет значение `--vars '{"var": "value"}'`. Обязательно, если переменная `public` не имеет значения по умолчанию.
- `cron` — периодичность выполнения процедуры. Период задается в формате `"*****"`

где

- первая `*` — минута. От 0 до 59,

- вторая \* — час. От 0 до 23,
- третья \* — день месяца. От 1 до 31,
- четвертая \* — месяц. От 1 до 12,
- пятая \* — день недели. От 0 до 7. Воскресенье=0 или 7.

### Пример

```
"* * * * 7" — выполняется каждое воскресенье,  
"* 2 * * 7" — каждые два часа по воскресеньям,  
"*/1 * * * *" — каждую минуту.
```

Полное описание синтаксиса и примеров `cron` см. [GitLab Docs/Cron](#)

### 2. Изменить расписание для выполнения процедуры

```
checker procedure schedule update --name <NAME>  
                                [--vars '{"var": "value"}']  
                                --cron <cron>
```

### 3. Просмотр запланированных процедур

```
checker procedure schedule list
```

### 4. Чтобы удалить процедуру из расписания, сначала узнайте ее идентификатор командой

```
checker procedure schedule list
```

Затем удалите процедуру из расписания

#### Удалить процедуру из плана исполнения

```
checker procedure schedule del --uuid <UUID>
```

где `uuid` — идентификатор процедуры.

### Примечание

Удаление из расписания не останавливает уже запущенный экземпляр процедуры, но новые запуски по расписанию прекратятся

### 24.2.3. Ручной запуск процедур

Если нужно протестировать процедуру или выполнять ее вручную, введите команду

```
checker procedure execute --name <NAME>
                          [--vars '{"var": "value"}']
                          [--json <no|yes>]
```

где

- `name` — имя процедуры;
- `vars` — переменная процедуры. Имеет значение `--vars '{"var": "value"}'`. Обязательно, если переменная `public` не имеет значения по умолчанию;
- `json` — формат возвращения ответа. Возможные значения:
  - `"no"` — ответ возвращается в человекочитаемом формате. Значение по умолчанию,
  - `"yes"` — ответ возвращается в формате JSON.

### 24.2.4. Принудительная остановка процедур

Чтобы остановить процедуру вручную, введите

```
checker procedure interrupt --uuid <UUID>
```

где `uuid` — идентификатор запущенной процедуры.

Команда может быть применена к процедуре, запущенной по плану или вручную.

### 24.2.5. Статусы запущенных процедур

Все запуски процедур, как по расписанию, так и ручные, имеют статус выполнения. Запланированные процедуры хранят только последний статус. История статуса хранится 90 дней, затем автоматически удаляется.

1. Чтобы просмотреть список статусов всех процедур, введите команду

```
checker procedure status list
```

2. Чтобы просмотреть статус определенной процедуры, введите

```
checker procedure status show --uuid <UUID>
                              [--json <no|yes>]
```

где

- `uuid` — идентификатор статуса процедуры, полученный в результате команды `checker procedure status list`;

- `json` — формат возвращения ответа. Возможные значения:
  - `"no"` — ответ возвращается в человекочитаемом формате. Значение по умолчанию,
  - `"yes"` — ответ возвращается в формате JSON.

В результате команды по статусу вернется следующая информация:

- `PENDING` — процедура создана, но еще не начала выполняться;
- `IN_PROGRESS` — процедура выполняется;
- `WAITING` — процедура ждет завершения другой задачи;
- `SUCCESS` — успешное завершение;
- `TIMEOUT` — завершена по таймауту;
- `ERROR` — завершена с ошибкой.

### Пример получения списка статусов и детальной информации

```
checker procedure status list
[
  {
    "task_uuid": "0c6910ba-7f28-4ccb-9d6a-2a7d575cfadd",
    <...>
  }
]

checker procedure status show --uuid 0c6910ba-7f28-4ccb-9d6a-2a7d575cfadd
{
  "task_status": "IN_PROGRESS",
  <...>
}
```

## 24.2.6. Практические советы

### ТЕСТИРОВАНИЕ НОВОЙ ПРОЦЕДУРЫ

1. Запустите процедуру вручную для проверки выполнения

```
checker procedure execute --name "procedure_name" --vars '{} ' --json "no|yes"
```

2. Получите UUID задачи из ответа.
3. Отслеживайте выполнение

```
checker procedure status show --uuid <UUID>
```

4. Проверьте логи и результат. Если процедура отработала без ошибок, добавьте ее в выполнение по расписанию.

## АНАЛИЗ ПРОБЛЕМ

- Используйте параметр `--json yes` для машинночитаемого вывода ошибок.
- Проверяйте логи через команду `checker procedure status show`.
- Убедитесь, что процедура делегирована в ваш ВЦОД. Если ее нет в списке при планировании, обратитесь к администратору кластера.
- Проверьте корректность cron-выражения.

---

### 24.2.7. Дополнительная информация

1. [YAML-структуры процедур](#). Описание того, как устроены процедуры для общего понимания. Создание процедур выполняет администратор кластера.
2. [Примеры процедур](#). Готовые процедуры, которые вы можете использовать в своем ВЦОД после делегирования.

## 24.3. Примеры процедур

В разделе приведен пример готовой процедуры, которую вы можете использовать во ВЦОД после того, как администратор кластера ее делегирует. Пример включает:

- назначение процедуры;
- публичные переменные, которые нужно передавать при запуске;
- команды для ручного запуска и планирования по расписанию.

### 24.3.1. Отслеживание и восстановление состояния VM

#### ЗАДАЧА

Автоматически обнаруживать VM в некорректном состоянии и выполнять их перезагрузку в следующих случаях:

- VM «зависает» и не отвечает.
- Необходимо автоматическое восстановление сервисов.
- Мониторинг критичных VM.

#### ПУБЛИЧНЫЕ ПЕРЕМЕННЫЕ `vars`

1. `vcl (string, public)` — имя виртуального кластера.
2. `vmname (string, public)` — имя виртуальной машины.
3. `state (string, public)` — целевое состояние VM.

## КАК ИСПОЛЬЗОВАТЬ

Возможны следующие варианты использования:

### 1. Запуск процедуры вручную

```
checker procedure execute --name vm_monitoring
                        --vars '{
                            "vc1": "production-vdc",
                            "vmname": "web-server-01",
                            "state": "reboot"
                        }'
```

### 2. Планирование проверки каждые 5 минут

```
checker procedure schedule add --name vm_monitoring
                               --vars
                               '{"vc1": "production-vdc", "vmname": "web-server-01", "state": "reboot"}'
                               --cron "* /5 * * * *"
```

---

## 24.3.2. Дополнительная информация

1. [Введение в checker.](#)
2. [Управление процедурами через CLI.](#) Базовые команды.
3. Полное описание внутреннего устройства процедур: YAML-структура, шаги, условия, обработка ошибок, представлено в статье [YAML-структура процедур](#). Создание процедур выполняет администратор кластера.