



Sharx Base 6.3

Руководство администратора

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. Введение	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
4.1. Обязательные шаги	17
4.2. Дополнительные шаги	18
4.2.1. Оптимизация и организация	18
4.2.2. Детализация прав доступа	18
4.2.3. Настройки безопасности	18
4.2.4. Обслуживание кластера	18
5. Аутентификация	19
5.1. Использование TLS-соединения для взаимодействия с Sharx Base	20
6. ВЦОД управления	21

Sharx Base 6.3. Руководство администратора

7. Информация об оборудовании кластера	22
8. Метки	23
8.1. Принципы выбора объекта для размещения ресурсов по меткам	23
8.2. Принцип размещения пользовательских ресурсов на узлы, содержащие метку-ограничение	24
8.3. Управлять метками в кластере	25
8.3.1. Ограничения работы с метками в кластере	25
8.3.2. Операции с метками кластера	26
8.4. Управлять метками во ВЦОД	30
8.4.1. Ограничения работы с метками во ВЦОД	30
8.4.2. Операции с метками во ВЦОД	31
9. Параметры IPAM и сетей	33
9.1. IPAM	33
9.2. Определить сети	35
9.3. VXLAN	37
9.3.1. Настроить базовые параметры плагина для работы с VXLAN	37
9.3.2. VXLAN-сети	37
9.3.3. Автоматическое управление сетевыми интерфейсами	39
9.4. Взаимодействие с OFC-контроллером	40
9.5. Удаление сетей и сетевых параметров	40
10. Управление ресурсами кластера	41
10.1. Резервирование ресурсов	41
10.1.1. Резервировать ресурсы для высокой доступности	41
10.1.2. Резервировать ресурсы под системные нужды	43
10.2. Переподписка ресурсов	45
10.2.1. Переподписка ЦПУ	45
10.2.2. Переподписка ОЗУ	48
10.3. Просмотр ресурсов и управление узлами кластера	53
10.3.1. Просмотр доступных ресурсов	53

Sharx Base 6.3. Руководство администратора

10.3.2. Управление узлами кластера	53
11. ВЦОД	53
11.1. Создать ВЦОД	54
11.2. Удалить ВЦОД	55
12. Хранилище	56
12.1. Типы хранилищ	56
12.1.1. Краткое описание типов хранилищ	56
12.1.2. Этапы работы с хранилищами	57
12.2. Libvirt. Пулы	58
12.2.1. Создать пул	58
12.2.2. Работа с пулами	58
12.2.3. Дальнейшая работа	60
12.3. РСХД. Настроить лимиты использования ресурсов	60
12.3.1. Резервировать ресурсы	61
12.3.2. Создать шаблон	62
12.3.3. Команды для работы с шаблонами	63
12.4. NFS. Настроить хранилище	64
13. Структура ВЦОД	68
14. Уведомления	69
14.1. Создать почтовый сервер	70
14.2. Сформировать список адресов	70
14.3. Задать шаблоны сообщений	71
14.4. Настроить маршрут отправки уведомлений	71
14.5. Настроить уведомления с помощью файла YAML	72
14.6. Контроль отправленных сообщений	72
15. Права доступа и роли	73
15.1. Права доступа	73

Sharx Base 6.3. Руководство администратора

15.2. Роли	74
15.2.1. Роли по умолчанию	74
15.2.2. Действия с ролями	77
16. Пользователи ВЦОД управления	78
16.1. Создать пользователя	79
16.2. Управлять пользователями	80
16.3. Ограничить время активности пользователя	81
16.4. Сессии пользователя	81
17. Время активности пользователя	82
18. Политика безопасности учетных записей	84
18.1. Команды управления базовыми параметрами механизмов безопасности	84
18.2. Параметры безопасности ВЦОД для обычного пользователя	84
18.3. Параметры ВЦОД для привилегированного пользователя	88
19. Валидация объектов ВЦОД	90
19.1. Управление ключами ВЦОД управления	91
19.2. Отправка сертификатов пользователям и включение двухфакторной аутентификации	91
19.3. Настройка клиентской машины для входа во ВЦОД с использованием сертификата	93
19.4. Настройка валидации объектов ВЦОД	94
19.5. Оповещение пользователя о событиях КЦ	98
19.6. Рекомендации по действиям Администратора кластера при нарушении контроля целостности	99
20. Резервное копирование данных кластера	99
20.1. Настроить резервное копирование кластера	99
20.2. Сбор логов по операции	102
21. Журнал безопасности	102
21.1. Просмотреть события в журнале безопасности	103

Sharx Base 6.3. Руководство администратора

21.2. Настроить параметры записи событий	104
21.2.1. Настроить параметры регистрации событий вручную	104
21.2.2. Загрузить файл с параметрами регистрации событий	105
21.3. Архивировать журнал безопасности	106
21.3.1. Настроить регулярное архивирование журнала безопасности	106
21.3.2. Принудительно архивировать и очистить журнал безопасности	109
22. Логирование	109
22.1. Создать конфигурацию подключения к rsyslog-серверу	110
22.2. Управлять конфигурацией	111
22.3. Пример настройки конечной точки подключения по TCP с TLS	112
23. Мониторинг	113
23.1. Общие сведения	113
23.2. Внутренний мониторинг Sharx Base	113
23.2.1. Экспортеры	113
23.2.2. Настройка мониторинга	114
23.2.3. Проверка работоспособности	118
23.3. Работа с шаблонами	119
23.4. Внешний мониторинг. Sharx ProView	120
24. Сервисный режим узлов	121
24.1. Перевести узлы в сервисный режим	121
24.2. Отключить сервисный режим	121
25. Обновить компоненты кластера	122
25.1. Автоматически обновить плагины	122
25.2. Обновить плагины вручную	123
25.3. Проверить обновления	123
26. Система автоматизации процедур checker	124
26.1. Checker	124
26.1.1. Ключевые понятия	124

Sharx Base 6.3. Руководство администратора

26.1.2. Как работает Checker?	124
26.1.3. Статьи по работе с системой Checker	124
26.2. Управление процедурами	125
26.2.1. Определить процедуру	125
26.2.2. Обновить процедуру	126
26.2.3. Просмотреть определение процедуры	127
26.2.4. Удалить процедуру	127
26.2.5. Делегировать процедуру	127
26.2.6. Планирование выполнения процедуры	128
26.2.7. Ручной запуск процедур	129
26.2.8. Принудительная остановка процедур	130
26.2.9. Статусы запущенных процедур	130
26.2.10. Практические советы	131
26.2.11. Дополнительная информация	132
26.3. YAML-структура процедур	132
26.4. Примеры процедур	144
26.4.1. Отслеживание и восстановление состояния VM	144
26.4.2. Фенсинг узлов кластера	147

1. Введение

Руководство администратора описывает последовательность действий пользователя с ролью **Администратор кластера** в Sharx Base для управления виртуальной инфраструктурой:

- ВЦОД управления;
- ВЦОД;
- пользователями и ролями;
- IPAM, сетями, хранилищами и другими настройками.

Некоторые операции доступны пользователю с ролью **Администратор безопасности кластера** в пределах его прав доступа.

Функции Администратора кластера и Администратора безопасности кластера описаны в статье [Права доступа и роли](#).

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.

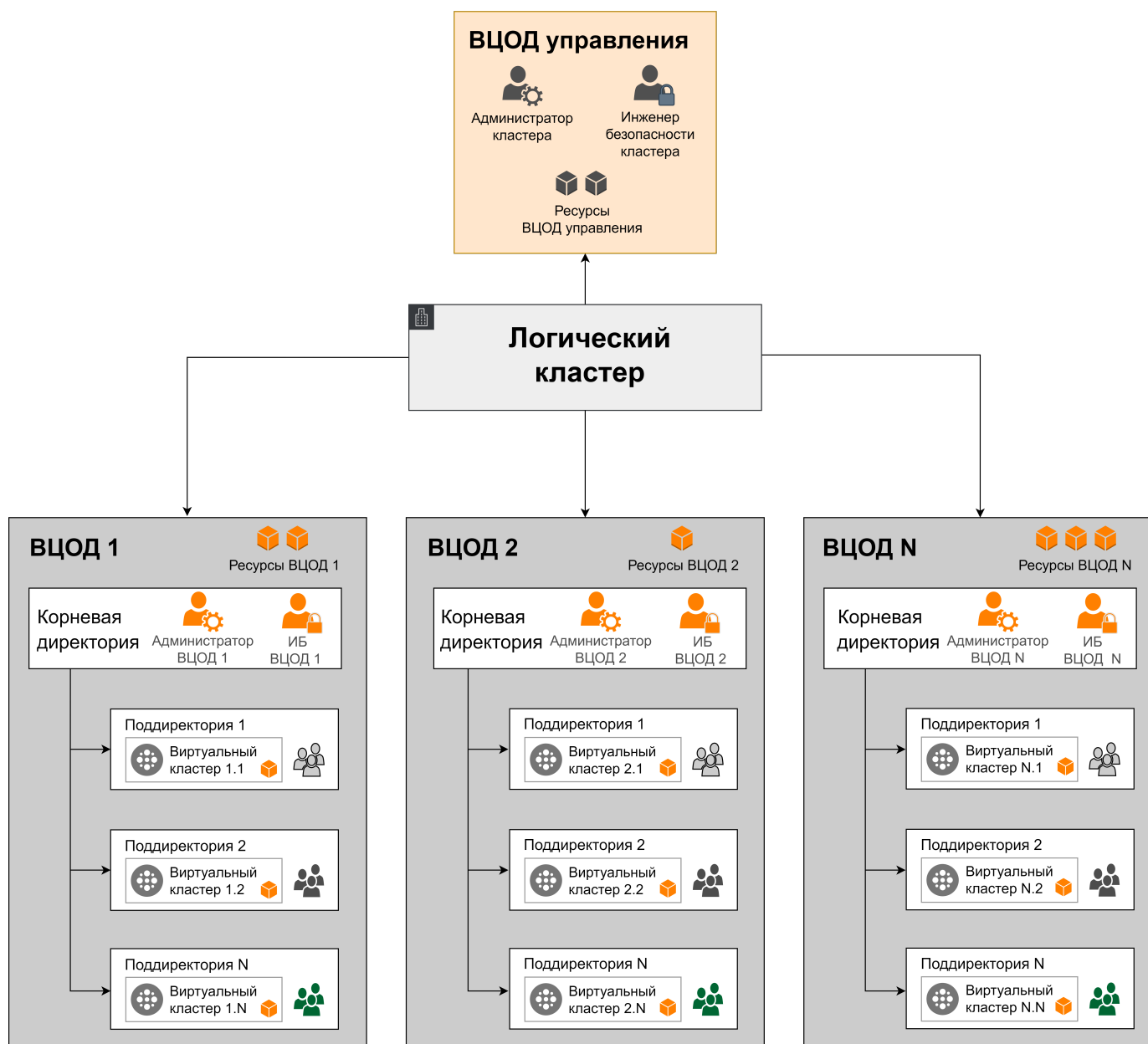
2.2. Что такое логический кластер?

Sharx Base 6.3. Руководство администратора

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.



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.

Sharx Base 6.3. Руководство администратора

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

Sharx Base 6.3. Руководство администратора

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 администратора

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.

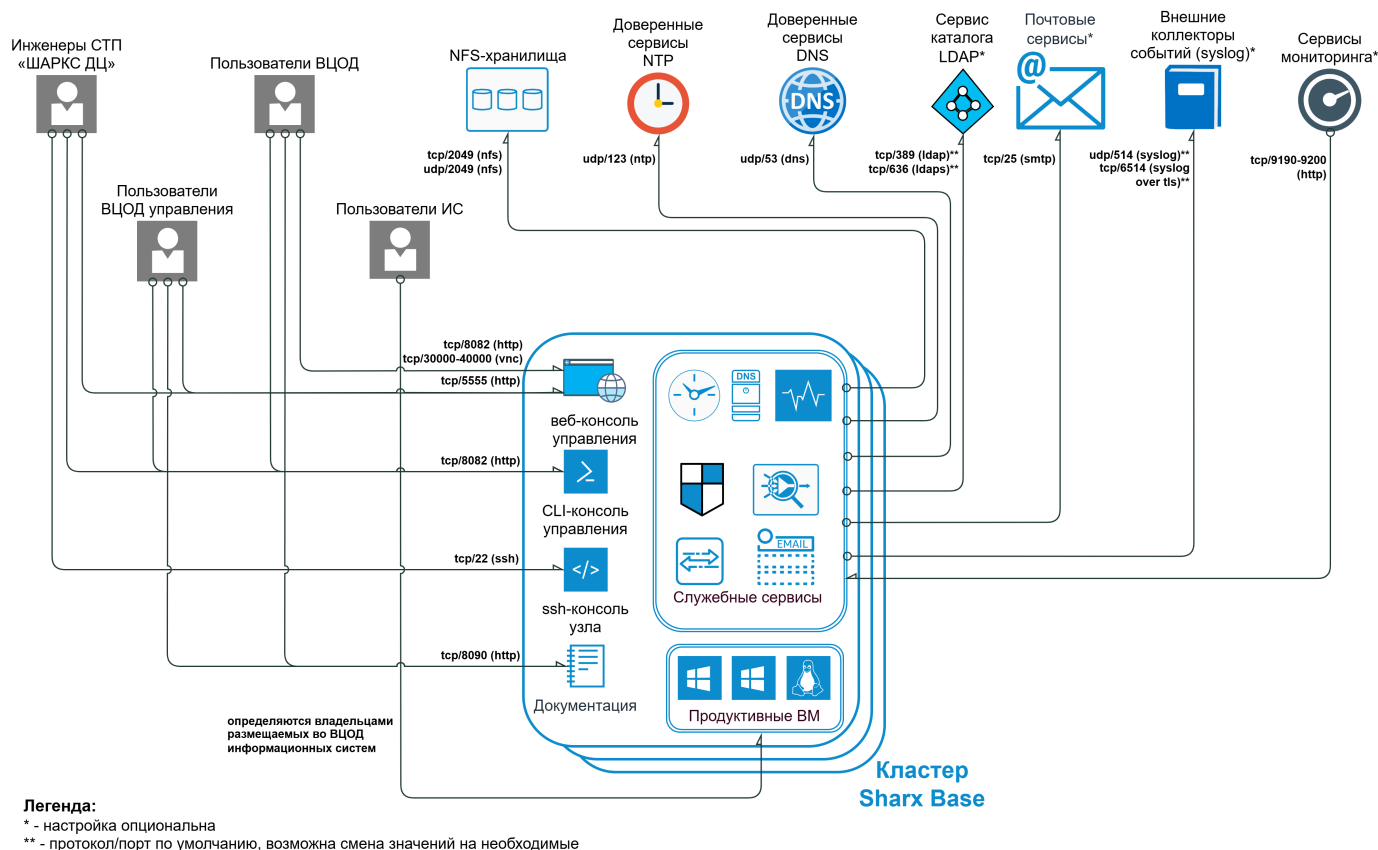


Таблица – Требования к сетевой доступности Платформы.

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 администратора

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 администратора

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.

Sharx Base 6.3. Руководство администратора

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
АРМ	
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)
Мобильное устройство	
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

Процесс настройки кластера разделен на обязательные шаги, без которых он не будет функционировать, и дополнительные, которые помогают оптимизировать и расширить функциональность кластера.

4.1. Обязательные шаги

1. [Аутентифицируйтесь в кластере.](#)
2. [Просмотрите информацию об оборудовании кластера.](#)
3. [Настройте метки кластера.](#)
Данный шаг не является обязательным, но оптимально выполнить его на данном этапе для базовых меток.
4. [Создайте параметры IPAM.](#)
5. [Настройте виртуальные сети кластера.](#)
6. [Настройте параметры переподписки.](#)
7. [Создайте ВЦОД.](#)
8. [Настройте необходимые хранилища:](#)
 - [Libvirt: пулы.](#)
 - [РСХД: группы хранения и шаблоны хранения.](#)

- NFS: настроить подключение.
-

4.2. Дополнительные шаги

Эти шаги можно выполнить после прохождения обязательных шагов настройки кластера.

4.2.1. Оптимизация и организация

1. Настройте ресурсы кластера.
2. Настройте структуру директорий.
3. Настройте уведомления для отслеживания состояния кластера и ВЦОД управления.

4.2.2. Детализация прав доступа

1. Настройте права доступа.
2. Настройте роли.
3. Настройте пользователей ВЦОД управления.

4.2.3. Настройки безопасности

1. Проверьте политику безопасности учетных записей.
2. Настройте валидацию объектов ВЦОД.
3. Выполните резервное копирование.
4. Выгрузите события журнала безопасности.
5. Настройте логирование.
6. Изучите мониторинг кластера.

4.2.4. Обслуживание кластера

1. Управляйте сервисным режимом узлов кластера.
2. Обновите компоненты до актуальных версий.
3. Расширяйте кластер или изменяйте состав узлов.
4. Настройте автоматический запуск процедур.

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. Далее введите

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

где

- `login` — логин администратора кластера. Значение по умолчанию — `admin`;
- `cluster` — имя логического кластера;
- `ns` — имя, которое пользователь присваивает создаваемому ВЦОД управления.

3. В случае успешного входа отобразится перечень плагинов, доступных в рамках созданного ВЦОД управления.

4. При необходимости **Администратор кластера** может авторизоваться в любом ВЦОД с помощью команды

```
login <LOGIN> --cluster <CLUSTER> --ns <NS> [--cluster_to <CLUSTER_TO>] [--ns_to <NS_TO>]
```

где

- `login` — логин администратора кластера. Значение по умолчанию — `admin`;

Sharx Base 6.3. Руководство администратора

- `cluster` — имя логического кластера;
- `ns` — имя ВЦОД управления;
- `cluster_to` — имя логического кластера, где находится ВЦОД, к которому нужно подключиться;
- `ns_to` — имя ВЦОД, к которому нужно подключиться.

5.1. Использование TLS-соединения для взаимодействия с Sharx Base

TLS-соединение защищает передачу данных между узлами клиентской и серверной машины за счет криптографического шифрования информации.

Примечание

Возможность использования TLS для взаимодействия с **Sharx Base** включает поставщик при сборке кластера.
После включения клиенту предоставляется файл `cert_name.crt` для дальнейшей настройки

Чтобы настроить TLS для взаимодействия с Sharx Base, выполните следующие действия:

1. Поместите файл `cert_name.crt`, предоставленный поставщиком, в хранилище доверенных сертификатов

ОС Debian

Директория `/usr/local/share/ca-certificates/`

ОС AlmaLinux

Директория `/etc/pki/ca-trust/source/anchors/`

2. Обновите сертификаты

ОС Debian

Выполните команду

```
update-ca-certificates
```

ОС AlmaLinux

Выполните команду

```
1 update-ca-trust
2 openssl x509 -in /etc/pki/ca-trust/extracted/openssl/ca-bundle.trust.crt -text | less
```

Sharx Base 6.3. Руководство администратора

3. После выполненных действий вход в Sharx Base должен осуществляться с использованием флагов `--tls_verify` и `--tls`

```
sdc-cli --tls_verify <TLS_VERIFY> --tls <yes|no>
```

где

- `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`.

6. ВЦОД управления

После аутентификации **Администратор кластера** находится во **ВЦОД управления**.

ВЦОД управления – пространство для управления доступными ресурсами (лимитами) логического кластера, используется только для настройки других пространств (ВЦОД) и не имеет собственной виртуализации (виртуальных кластеров и VM).

ВЦОД управления создается технической поддержкой при подготовке кластера.

Во ВЦОД управления по умолчанию созданы две роли:

- **Администратор кластера;**
- **Администратор безопасности кластера.**

Затем администратор кластера с помощью конфигурационного файла создает ВЦОД.

Созданный 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.

Действия по управлению ВЦОД и его виртуальными объектами выполняет пользователь с ролью Администратор ВЦОД. Описание действий приведены в отдельных разделах портала документации: в [Руководстве пользователя в командной строке](#) и в [Руководстве пользователя в веб-интерфейсе](#).

Администратор кластера создает шаблоны для использования ресурсов хранилища в рамках определенного ВЦОД. Подробно с терминологией и процессом создания шаблонов можно ознакомиться в статье [РСХД. Настроить лимиты использования ресурсов](#).

Чтобы посмотреть информацию и статус ВЦОД управления, введите

```
aaa namespace show --cluster <CLUSTER> --ns <NS>
```

где

- `cluster` — имя логического кластера;
- `ns` — имя ВЦОД управления.

Примечание

Действия для создания ВЦОД описаны в статье [Создать ВЦОД](#)

7. Информация об оборудовании кластера

Администратор кластера может получить актуальную информацию о физических ресурсах кластера. Обновление данных происходит регулярно. Полученная информация собирается и фильтруется для каждого отдельного логического кластера.

Чтобы просмотреть текущие данные о ресурсах конкретного узла, выполните команду

```
hardware event list --event_id <EVENT_ID>  
                    --server_id <SERVER_ID>  
                    [--limit <LIMIT>]
```

где

- `event_id` — тип сущности или атрибута, который необходимо отобразить. Возможные значения:
 - `baseboardlibvirt` — характеристики платы,
 - `cpulibvirt` — характеристики процессора,
 - `cpulibvirtinfo` — доступные ресурсы узла,
 - `cpudmi` — частота шины,
 - `ramlibvirt` — информация об объеме ОЗУ,
 - `ramutil` — использование ОЗУ,
 - `network` — подключенные сетевые интерфейсы,
 - `mnt` — точка монтирования;
- `server_id` — идентификатор узла, по которому выполняется запрос;
- `limit` — максимальное количество типов сущностей или атрибутов `event` для отображения.

8. Метки

Метки (labels) – пары ключ-значение формата `key=value`, которые описывают атрибуты объектов кластера. Они играют ключевую роль в механизме планирования размещения пользовательских ресурсов (VM) на узлах. Метки могут быть назначены в кластере, ВЦОД и виртуальном кластере.

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.
- **Делегирование метки** – предоставление возможности использовать метку объектам: ВЦОД или виртуальному кластеру.
- **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.
- **Ограничение `taint`** – свойство узла, не позволяющее планировать на него ресурсы, у которых нет допуска (toleration) к узлу с данным ограничением `taint`.
- **Метка-ограничение** – метка со свойством `taint`. На узлы с такой меткой ресурсы планируются по определенным правилам.
- **Допуски `tolerations`** – это параметры виртуальных машин, которые позволяют им размещаться на узлах с ограничениями. Допуск должен соответствовать введенному ограничению `taint` на узле `key=value:effect`.
- **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 [Сервисное обслуживание](#)
- **Криптографическая метка** – метка, применяемая к VM для обозначения ее принадлежности к контуру криптографической защиты (СКЗИ). Имеет ключ-значение `vm=crypto`. Метка определяет политику управления VM, блокируя функции миграции, создания снимков и шаблонов на основе VM для исключения риска утечки защищаемых данных. Криптографическую метку может назначить или снять только определенная роль специальной командой. Подробная информация описана в [Руководстве пользователя в командной строке](#).

8.1. Принципы выбора объекта для размещения ресурсов по меткам

✘ Внимание

- 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

Планировщик размещает ресурсы только на узлах, содержащих **все** метки, указанные в запросе на создание ресурса в элементе `nodeSelector`. Игнорируемые метки **не участвуют в фильтрации узлов**.

Например, узлам А, В и С присвоены метки

node	labels
A	key=base, disk=ssd, graphics=fast, foo=bar
B	key=base
C	key=base, disk=ssd

Получен запрос на создание пользовательского ресурса

`nodeSelector` из запроса на создание ресурса

```
nodeSelector:  
  key: base  
  disk: ssd  
  graphics: fast
```

Только узел А будет выбран планировщиком доступным для размещения ресурса, так как он единственный содержит **все** запрошенные метки.

8.2. Принцип размещения пользовательских ресурсов на узлы, содержащие метку-ограничение

При подборе узлов, где существует метка-ограничение, планировщик следует определенному алгоритму:

Sharx Base 6.3. Руководство администратора

1. Если метка-ограничение не была указана пользователем в параметре `nodeSelector`, ресурс не будет запланирован на узлы, имеющие эту метку.
2. Если метка-ограничение была указана пользователем в параметре `nodeSelector`, ресурс автоматически получит допуск к размещению на узле, содержащем данную метку-ограничение.

Например, узлам A, B и C присвоены метки

node	labels
A	key=base, foo=bar:{taint}
B	key=base
C	key=base

где метка `foo=bar:{taint}` — метка-ограничение.

Получен запрос на создание пользовательского ресурса

nodeSelector из запроса на создание ресурса

```
nodeSelector:  
  key: base
```

Планировщик выберет только узлы B и C, так как они не имеют метку-ограничение.

Если получен запрос с параметром `nodeSelector` вида

nodeSelector из запроса на создание ресурса

```
nodeSelector:  
  foo: bar
```

Планировщик выберет только узел A, так как он единственный имеет метку-ограничение `foo=bar:{taint}`.

8.3. Управлять метками в кластере

8.3.1. Ограничения работы с метками в кластере

- При создании кластера каждому узлу по умолчанию присваивается метка `key=base`.

Sharx Base 6.3. Руководство администратора

- Максимальное количество меток на узле – 150.
- Узлу нельзя назначить несколько меток с одинаковым ключом.
- С узла нельзя открепить все метки.
- Нельзя удалить метки, делегированные во ВЦОД и в виртуальные кластеры.
- Метки можно создавать и редактировать на узлах до создания первого ВЦОД (не считая ВЦОД управления). За исключением ситуаций, когда в кластер добавляется новый узел, и на нем еще нет меток.
- Из обычной метки можно сделать метку-ограничение. Также из метки-ограничения можно сделать обычную метку, удалив у нее свойство `taint`. Добавлять или удалять свойство `taint` меткам можно только до создания первого ВЦОД (не считая ВЦОД управления).
- Общие ограничения описаны [выше](#).

8.3.2. Операции с метками кластера

1. Создать и назначить метки узлам кластера

✖ Внимание

Игнорировать можно те метки, которые были предварительно делегированы во ВЦОД

```
scheduler labels add --labels <LABELS>
                    --nodes <NODES>
                    [--taints <TAINTS>]
                    [--descr <DESCR>]
```

где

- `labels` – список меток узла. Метки задаются в формате `key=value`, разделяются пробелами;
- `nodes` – список идентификаторов узлов, на которые устанавливается метка. При значении `*` метка устанавливается на все узлы;
- `taints` – свойство ограничения для метки. Возможные значения:
 - `n` – метка останется обычной. Значение по умолчанию;
 - `y` – метка с ограничением `NoSchedule`;

При указании `--taints y` метка будет помечена как метка-ограничение. VM будут размещаться только при явном указании этой метки в `nodeSelector`. Описание метки `descr` будет присвоено всем создаваемым меткам в данном запросе.

- `descr` – описание метки.

Пример

```
scheduler labels add --labels node=security
                    --nodes be492e22-d18d-46e6-9afc-b4c839ab584c bd2fcdfc-f4d3-4233-8a6b-
                    f4ac4cbbf739
                    --taints y
                    --descr secure nodes for critical applications
```

2. Назначить существующие метки узлам кластера

```
scheduler labels assign --labels <LABELS>
                       --nodes <NODES>
```

где

- `labels` — список меток узла. Метки задаются в формате `key=value`, разделяются пробелами;
- `nodes` — список идентификаторов узлов, на которые устанавливается метка. При значении `*` метка устанавливается на все узлы.

Пример

```
scheduler labels assign --labels graphics=fast disk=ssd
                       --nodes be492e22-d18d-46e6-9afc-b4c839ab584c bd2fcdfc-
                       f4d3-4233-8a6b-f4ac4cbbf739
```

3. Просмотреть метки кластера со списками узлов

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

где

- `filter` — фильтрация меток. Возможные значения:
 - `a` — all, все метки;
 - `i` — ignored, игнорируемые метки;
 - `u` — unignored, все метки, кроме игнорируемых.

4. Просмотреть узлы кластера со списками их меток

```
scheduler labels list [--nodes <NODES>]
                    [--filter <FILTER>]
```

где

- `nodes` — список идентификаторов узлов. При значении `*` выводятся метки на всех узлах;
- `filter` — фильтрация меток. Возможные значения:

Sharx Base 6.3. Руководство администратора

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

Пример

```
scheduler labels list --nodes be492e22-d18d-46e6-9afc-b4c839ab584c
```

5. Запросить список узлов кластера по определенным меткам

Внимание

Игнорируемые метки не будут учтены при поиске

```
scheduler labels nodes list --labels <LABELS>
                             [--strict <STRICT>]
                             [--silent <SILENT>]
```

где

- `labels` – список меток. Метки задаются в формате `key=value`, разделяются пробелами;
- `strict` – отображать узлы в соответствии с набором меток. Возможные значения:
 - при `strict y` отображать узлы, которые имеют **все** запрошенные метки;
 - при `strict n` отображать узлы, которые имеют **хотя бы одну** из запрошенных меток;
- `silent` – отображать ошибки, если какие-либо метки не существуют. Возможные значения:
 - `y` – отображать ошибки;
 - `n` – не отображать ошибки.

Пример

Узлам А и В присвоены метки

node	labels
A	key=base, disk=ssd, foo=bar
B	key=base

Запрос 1 вернет список [A, B]

Запрос 1

```
scheduler labels nodes list --labels key=base disk=ssd --strict n
```

Запрос 2 вернет только А

Запрос 2

```
scheduler labels nodes list --labels key=base disk=ssd --strict y
```

6. Игнорировать метки в пределах всего кластера.

Если не указывать параметр `ns`, то метка заблокируется для всех новых и уже существующих ВЦОД: в новый ВЦОД метку добавить будет нельзя, а в уже существующих ВЦОД новые виртуальные кластеры будут создаваться без этой метки.

Внимание

Игнорировать можно те метки, которые были предварительно делегированы во ВЦОД

```
scheduler labels ignore --labels <LABELS>
                        [--ns <NS>]
```

где

- `labels` – список меток для игнорирования. Метки задаются в формате `key=value`, разделяются пробелами;
- `ns` – список ВЦОД. При значении `*` метки будут автоматически помечены как игнорируемые во всех ВЦОД кластера, куда они были делегированы.

Пример

```
scheduler labels ignore --labels disk=ssd key=base
```

7. Сделать обычную метку меткой-ограничением

```
scheduler labels taints add --labels <LABELS>
```

Меткам, указанным в `labels`, будет присвоено ограничение `NoSchedule`.

Пример

```
scheduler labels taints add --labels foo=bar
```

8. Сделать из метки-ограничения обычную метку

```
scheduler labels taints del --labels <LABELS>
```

Пример

```
scheduler labels taints del --labels foo=bar
```

9. Удалить метки кластера

Примечание

При удалении метки предварительно будет произведена попытка ее открепления со всех узлов кластера

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

где `labels` – список меток для удаления. Метки задаются в формате `key=value`, разделяются пробелами.

Пример

```
scheduler labels del --labels disk=ssd key=base
```

8.4. Управлять метками во ВЦОД

8.4.1. Ограничения работы с метками во ВЦОД

- При создании нового ВЦОД и при наличии в кластере базовой метки `key=base` и метки сервисного обслуживания `system_drain_node=yes` они будут автоматически делегированы в созданный ВЦОД.
- Во ВЦОД нельзя делегировать метки, игнорируемые в пределах кластера.

- При удалении ВЦОД все записи о делегированных ему метках удаляются.
- Общие ограничения описаны [выше](#).

8.4.2. Операции с метками во ВЦОД

Примечание

Действия выполняются пользователем с ролью **Администратор кластера** во ВЦОД управления

1. Делегировать метки в заданные ВЦОД

```
scheduler labels delegate ns --ns <NS>
                        --labels <LABELS>
```

где

- `ns` — список имен ВЦОД. При значении `*` метка будет делегирована на все ВЦОД;
- `labels` — список делегируемых меток. Метки задаются в формате `key=value`, разделяются пробелами.

Пример

```
scheduler labels delegate ns --ns vdc --labels disk=ssd key=base
```

2. Просмотреть метки во ВЦОД

Внимание

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

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

где

- `ns` — список имен ВЦОД для просмотра их меток.
Администраторы кластера `admins` могут просмотреть метки всех ВЦОД кластера, добавив флаг `ns *`.
Чтобы ограничить вывод меток для определенных ВЦОД, администратор кластера должен указать их имена в `ns`.
Пользователям с другими ролями доступен просмотр меток только в своем ВЦОД без указания флага `ns`.

Sharx Base 6.3. Руководство администратора

- `filter` – фильтрация меток. Возможные значения:
 - `a` – all, все метки;
 - `i` – ignored, игнорируемые метки;
 - `u` – unignored, все метки, кроме игнорируемых.

3. Просмотр меток со списками ВЦОД, в которые они делегированы

```
scheduler labels ns list [--ns <NS>]
                        [--filter <FILTER>]
```

где

- `ns` – список имен ВЦОД для просмотра делегированных меток. Администраторы кластера `admins` могут просмотреть делегированные метки всех ВЦОД кластера, добавив флаг `--ns *`. Чтобы ограничить вывод меток для определенных ВЦОД, администратор кластера должен указать их имена в `--ns`. Пользователям с другими ролями доступен просмотр делегированных меток только в своем ВЦОД без указания флага `--ns`.
- `filter` – фильтрация меток. Возможные значения:
 - `a` – all, все метки;
 - `i` – ignored, игнорируемые метки;
 - `u` – unignored, все метки, кроме игнорируемых.

4. Игнорировать метки в пределах заданных ВЦОД

Внимание

Игнорировать можно те метки, которые были предварительно делегированы во ВЦОД

```
scheduler labels ignore --labels <LABELS>
                        [--ns <NS>]
```

где

- `labels` – список меток для игнорирования. Метки задаются в формате `key=value`, разделяются пробелами;
- `ns` – список ВЦОД. При значении `*` метки будут автоматически помечены как игнорируемые во всех ВЦОД кластера, куда они были делегированы.

Пример

```
scheduler labels ignore --ns vdc --labels disk=ssd key=base
```

9. Параметры IPAM и сетей

Управление IP-адресами (IPAM) – это интегрированный набор средств для комплексного планирования, развертывания, управления и мониторинга инфраструктуры IP-адресов с широкими возможностями взаимодействия с пользователем. IPAM автоматически определяет серверы инфраструктуры IP-адресов и DNS-серверы в сети и позволяет управлять ими из центрального интерфейса.

В статье описаны:

1. Настройка базовой сетевой конфигурации:
 - глобальные параметры IPAM для всего кластера;
 - диапазоны VID и зарезервированные сети;
 - настройки мостов и MTU для виртуальных коммутаторов.
2. Настройка виртуальных сетей двух типов:
 - VLAN-сети. Классической сегментации трафика с поддержкой ACL.
 - VXLAN-сети. Для overlay-сетей. Без поддержки ACL.
3. Принцип работы со специализированной инфраструктурой:
 - OpenFlow-контроллер;
 - Автоматическое распределение IP-адресов.

Рекомендуемый порядок работы:

1. [Начальная настройка](#). Задание глобальных параметров IPAM.
2. Создание сетей. Определение [VLAN](#) или [VXLAN](#).
3. [Настройка сервиса взаимодействия с OFC-контроллером](#).
4. Оперативное управление сетями. Мониторинг и изменение параметров.

9.1. IPAM

Первоначально задайте параметры IPAM, которые определяют базовые настройки.

1. В **Sharx Base** в командной строке введите

```
ipam param add --mac_prefix <MAC_PREFIX>
               --allocated_vlans <ALLOCATED_VLANS>
               --reserved_vlans <RESERVED_VLANS>
               --bridge <BRIDGE>
               [--reserved_v4_nets <RESERVED_V4_NETS>]
               [--mtu <MTU>]
```

Sharx Base 6.3. Руководство администратора

```
[--vtep_if <VTEP_IF>]  
[--vxlan_vid_range <VXLAN_VID_RANGE>]
```

где

- `mac_prefix` — префикс MAC-адреса, который будет присваиваться виртуальным машинам;
- `allocated_vlans` — диапазон VID, используемых системой для создания виртуальных сетей;
- `reserved_vlans` — зарезервированные VID, недоступные для создания виртуальных сетей;
- `bridge` — имя бридж-моста для подключения виртуальных сетей;
- `reserved_v4_nets` — зарезервированные IPv4-подсети для системных нужд;
- `mtu` — максимальный размер пакета в байтах, который может быть отправлен одновременно без фрагментации. Указывается числом без единиц измерения. Рекомендуемое значение — 1500;
- `vtep_if` — имя VTEP-интерфейса;
- `vxlan_vid_range` — список выделенных ID для VXLAN-сетей.

2. Чтобы посмотреть созданные настройки, введите команду

```
ipam param show
```

3. Обновить параметры

```
ipam param update [--mac_prefix <MAC_PREFIX>]  
                  [--allocated_vlans <ALLOCATED_VLANS>]  
                  [--reserved_vlans <RESERVED_VLANS>]  
                  [--reserved_v4_nets <RESERVED_V4_NETS>]  
                  [--v4_gw <V4_GW>]  
                  [--mtu <MTU>]  
                  [--bridge <BRIDGE>]  
                  [--vtep_if <VTEP_IF>]  
                  [--vxlan_vid_range <VXLAN_VID_RANGE>]
```

где

- `mac_prefix` — префикс MAC-адреса, который будет присваиваться виртуальным машинам;
- `allocated_vlans` — диапазон VID, используемых системой для создания виртуальных сетей;
- `reserved_vlans` — зарезервированные VID, недоступные для создания виртуальных сетей;
- `reserved_v4_nets` — зарезервированные IPv4-подсети для системных нужд;
- `v4_gw` — адрес шлюза по умолчанию для протокола сети IPv4;

Sharx Base 6.3. Руководство администратора

- `mtu` — максимальный размер пакета в байтах, который может быть отправлен одновременно без фрагментации. Указывается числом без единиц измерения. Рекомендуемое значение — 1500;
- `bridge` — имя бридж-моста для подключения виртуальных сетей;
- `vtep_if` — имя VTEP-интерфейса;
- `vxlan_vid_range` — список выделенных ID для VXLAN-сетей.

9.2. Определить сети

1. Чтобы определить сеть, в командной строке введите

```
ipam network add --name <NAME>
                 --ipv <IPV>
                 --vlan <VLAN>
                 --v4_net <V4_NET>
                 [--ns <NS>]
                 [--mtu <MTU>]
                 [--v4_pref <V4_PREF>]
                 [--v4_mask <V4_MASK>]
                 [--v4_start <V4_START>]
                 [--v4_end <V4_END>]
                 [--v4_gw <V4_GW>]
                 [--v4_dns1 <V4_DNS1>]
                 [--v4_dns2 <V4_DNS2>]
                 [--descr <DESCR>]
```

где

- `name` — имя сети;
- `ipv` — версия протокола IP-сети. Возможное значение: 4 — IPv4;
- `vlan` — идентификатор виртуальной локальной сети;
- `v4_net` — адрес IPv4-сети. Например, X.X.X.X;
- `ns` — имя ВЦОД. Обязательно для `network`, необязательно для `vxlan`;
- `mtu` — максимальный размер пакета в байтах, который может быть отправлен одновременно без фрагментации. Указывается числом без единиц измерения. Рекомендуемое значение — 1500;
- `v4_pref` — префикс подсети IPv4 в формате CIDR;
- `v4_mask` — маска подсети IPv4 в формате CIDR;
- `v4_start` — первый адрес IPv4 для назначения VM;
- `v4_end` — последний адрес IPv4 для назначения VM;
- `v4_gw` — адрес шлюза по умолчанию для протокола сети IPv4;

Sharx Base 6.3. Руководство администратора

- `v4_dns1` – первичный DNS-сервер сети IPv4;
- `v4_dns2` – вторичный DNS-сервер сети IPv4
- `descr` – описание сети.

2. Посмотреть список созданных сетей

```
ipam network list [--show_acl <yes|no>]
```

где `show_acl` – показать правила списков контроля доступа для каждой сети. Значение по умолчанию – `no`. Возможные значения: `yes`, `no`.

3. Посмотреть подробную информацию о конкретной сети

```
ipam network show --net_uuid <NET_UUID>  
                [--show_acl <yes|no>]
```

где

- `net_uuid` – идентификатор сети;
- `show_acl` – показать правила списков контроля доступа для каждой сети. Значение по умолчанию – `no`. Возможные значения: `yes`, `no`.

4. Обновить параметры сети

```
ipam network update --net_uuid <NET_UUID>  
                   [--name <NAME>]  
                   [--ns <NS>]  
                   [--mtu <MTU>]  
                   [--v4_gw <V4_GW>]  
                   [--descr <DESCR>]  
                   [--v4_start <V4_START>]  
                   [--v4_end <V4_END>]  
                   [--v4_dns1 <V4_DNS1>]  
                   [--v4_dns2 <V4_DNS2>]
```

где

- `net_uuid` – идентификатор сети;
- `name` – имя сети;
- `ns` – имя ВЦОД. Обязательно для `network`, необязательно для `vxlان`;
- `mtu` – максимальный размер пакета в байтах, который может быть отправлен одновременно без фрагментации. Указывается числом без единиц измерения. Рекомендуемое значение – `1500`;
- `v4_gw` – адрес шлюза по умолчанию для протокола сети IPv4;
- `descr` – описание сети;
- `v4_start` – первый адрес IPv4 для назначения ВМ;
- `v4_end` – последний адрес IPv4 для назначения ВМ;

Sharx Base 6.3. Руководство администратора

- `v4_dns1` – первичный DNS-сервер сети IPv4;
- `v4_dns2` – вторичный DNS-сервер сети IPv4.

9.3. VXLAN

VXLAN – это технология виртуализации сетей, которая расширяет возможности обычных виртуальных локальных сетей. VXLAN позволяет создавать изолированные виртуальные сегменты поверх существующей сетевой инфраструктуры.

Важно

Перед созданием инфраструктуры VXLAN необходимо задать базовые параметры плагина IPAM

9.3.1. Настроить базовые параметры плагина для работы с VXLAN

Чтобы задать или изменить базовые параметры плагина, введите

```
ipam param update --vtep_if <VTEP_IF>
                  --vxlan_vid_range <VXLAN_VID_RANGE>
```

где

- `vtep_if` – имя VTEP-интерфейса;
- `vxlan_vid_range` – диапазон ID (VID), из которого будут выделяться идентификаторы подсети VXLAN.

Важно

Значение `vxlan_vid_range` используется всеми VXLAN-сетями кластера.

`vxlan_vid_range` не может пересекаться с диапазонами `allocated_vlans` и `reserved_vlans`

Пример

```
ipam param update --vtep_if <vtep_if_name> --vxlan_vid_range 1000-4094
```

Каждой VXLAN-подсети при создании автоматически будет назначаться уникальный VID из указанного диапазона

9.3.2. VXLAN-сети

Sharx Base 6.3. Руководство администратора

Особенности VXLAN-сетей:

- VXLAN-сеть выступает контейнером для изолированных подсетей с автоматической маршрутизацией между ними;
- для работы требуется предварительная настройка VTEP-интерфейса;
- каждому ВЦОД может быть назначена только одна VXLAN-сеть;
- нельзя изменить привязку к ВЦОД, если во VXLAN-сети уже созданы подсети;
- VXLAN-сеть нельзя удалить, если внутри нее созданы VXLAN-подсети.

Действия с VXLAN-сетями:

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

```
ipam vxlan network add --name <NAME>
                        [--ns <NS>]
                        [--descr <DESCR>]
```

где

- `name` — имя VXLAN-сети;
- `ns` — имя ВЦОД. ВЦОД может быть привязан только к одной VXLAN-сети;
- `descr` — описание VXLAN-сети.

Пример

```
ipam vxlan network add --name vx-1 --ns vcod-1 --descr "vxlan network"
```

2. Просмотреть список VXLAN-сетей кластера

```
ipam vxlan network list
```

3. Изменить ВЦОД для VXLAN-сети

Внимание

Нельзя изменить привязку к ВЦОД, если во VXLAN-сети уже созданы подсети

```
ipam vxlan network update --vxlan_uuid <VXLAN_UUID>
                           [--name <NAME>]
                           [--ns <NS>]
                           [--descr <DESCR>]
```

где

- `vxlan_uuid` — идентификатор VXLAN-сети;
- `name` — имя VXLAN-сети;

Sharx Base 6.3. Руководство администратора

- `ns` — имя ВЦОД;
- `descr` — описание VXLAN-сети.

Пример

```
ipam vxlan network update --vxlan_uuid 4ab2646c-1109-42ad-a3a3-506cb708c87c
--ns vcod-2
```

4. Удалить VXLAN-сеть

Внимание

Нельзя удалить VXLAN-сеть, если в ней есть подсети

```
ipam vxlan network del --vxlan_uuid <VXLAN_UUID>
```

где `vxlan_uuid` — идентификатор VXLAN-сети.

Пример

```
ipam vxlan network del --vxlan_uuid 4ab2646c-1109-42ad-a3a3-506cb708c87c
```

5. Просмотреть сетевые интерфейсы

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

где

- `net_uuid` — идентификатор сети;
- `vxlan_uuid` — идентификатор VXLAN-сети;
- `show_acl` — показать правила списков контроля доступа для каждой сети. Значение по умолчанию — `no`. Возможные значения: `yes`, `no`.

Пример

```
ipam iface list --vxlan_uuid 4ab2646c-1109-42ad-a3a3-506cb708c87c
```

Просмотреть сетевые интерфейсы конкретной подсети VXLAN

9.3.3. Автоматическое управление сетевыми интерфейсами

Sharx Base автоматически управляет сетевыми интерфейсами:

Sharx Base 6.3. Руководство администратора

- 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.

Дополнительные действия с сетевыми параметрами выполняет Администратор ВЦОД см. [Руководство пользователя](#).

9.4. Взаимодействие с OFC-контроллером

Для взаимодействия с точками конфигурации OpenFlow в Sharx Base используется сервис для запуска, перезапуска или остановки системных сервисов на узлах виртуализации кластера.

Чтобы включить контроль за сервисом, используйте команду

```
services global systemd add --name faucet
                               --code VIP
                               --state start
                               [--descr <DESCR>]
```

где

- `name` — имя сервиса;
- `code` — стратегия работы сервиса. Возможные значения: `VIP` — сервис работает на VIP-узлах, `ALL` — сервис работает на всех узлах;
- `state` — состояние сервиса. Значение по умолчанию — `start`. Возможные значения: `start` — запуск сервиса, `stop` — остановка сервиса;
- `descr` — описание сервиса.

После выполнения команды на узлах кластера запускается сервис **faucet**, который обновляет описание OpenFlow-контроллера.

9.5. Удаление сетей и сетевых параметров

1. Чтобы удалить сеть, введите команду

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

где `net_uuid` — идентификатор сети.

2. Удалить параметры сети

```
ipam param del
```

Примечание

Дополнительные действия с сетевыми параметрами выполняет Администратор ВЦОД см. [Руководство пользователя](#)

10. Управление ресурсами кластера

Раздел содержит настройку и управление ресурсами логического кластера.

Sharx Base использует систему управления ресурсами, которая обеспечивает доступность за счет резервирования ресурсов для системных компонентов и отказоустойчивости, эффективное использование с помощью переподписки для оптимизации загрузки оборудования и гибкое масштабирование для возможности расширения кластера без прерывания работы.

Для нового кластера выполните следующие шаги:

1. [Настройте резервирование для высокой доступности.](#)
2. [Настройте резервирование для системных нужд.](#)
3. [Настройте переподписку.](#)
4. [Проверьте доступные ресурсы перед созданием ВЦОД.](#)

При расширении кластера или изменении состава узлов в кластере:

1. [Добавьте новые узлы в кластер.](#)
2. [Обновите резервирование системных ресурсов.](#)

10.1. Резервирование ресурсов

Важно

Настройка параметров выполняется до создания ВЦОД

10.1.1. Резервировать ресурсы для высокой доступности

Sharx Base 6.3. Руководство администратора

Высокая доступность кластера (отказоустойчивость, High Availability, HA) – это свойство Платформы обеспечивать максимально продолжительную доступность даже при отказе компонентов из-за аварии. При корректных настройках HA без дополнительных затрат минимизируется время простоя, вызванное сбоем серверов или операционных систем.

Процент резерва дискового пространства, памяти и ВЦПУ зависит от предположения, сколько узлов может выйти из строя одновременно. В этом случае зарезервированные ресурсы получают ВЦОД, которые в них нуждаются.

Например, для кластера из 3 узлов рекомендуется устанавливать значение 20%, так как HA предполагает возможность потери одного узла без нарушения работоспособности.

1. Чтобы настроить отказоустойчивость, в командной строке введите

```
aaa cluster resource ha add [--reserved_ha_cpu <RESERVED_HA_CPU>]
                             [--reserved_ha_ram <RESERVED_HA_RAM>]
                             [--reserved_ha_disk <RESERVED_HA_DISK>]
                             [--resources_share no]
```

где

- `reserved_ha_disk` – процентное резервирование дискового пространства для отказоустойчивости на узле кластера. Для РСХД-хранилища данный параметр применять нельзя;
- `reserved_ha_ram` – процентное резервирование оперативной памяти для отказоустойчивости на узле кластера;
- `reserved_ha_cpu` – процентное резервирование виртуального процессора для отказоустойчивости на узле кластера;
- `resources_share` – совместное использование ресурсов кластера. Возможное значение – *no*.

2. Просмотр текущих настроек

```
aaa cluster resource ha show
```

3. Обновить текущие настройки высокой доступности

```
aaa cluster resource ha update [--reserved_ha_cpu <RESERVED_HA_CPU>]
                                [--reserved_ha_ram <RESERVED_HA_RAM>]
                                [--reserved_ha_disk <RESERVED_HA_DISK>]
                                [--resources_share no]
```

4. Удалить текущие настройки высокой доступности

```
aaa cluster resource ha del
```

10.1.2. Резервировать ресурсы под системные нужды

Sharx Base может автоматически рассчитать и зарезервировать ресурсы ЦПУ и ОЗУ для системных нужд на основе конфигурации контрольных групп на узлах кластера. Чтобы функция резервирования выполнялась корректно, необходимо:

1. Убедиться в корректной настройке отслеживания контрольных групп.
2. Применить функцию автоматического резерва ресурсов под системные нужды.

При необходимости можно вручную изменить резервирование ресурсов согласно разделу [Ручное управление резервированием](#).

КОНТРОЛЬНЫЕ ГРУППЫ

Важно

Перед настройкой резервирования ресурсов необходимо убедиться в корректной конфигурации контрольных групп на узлах кластера и настроить отслеживание этих групп

Контрольная группа – механизм контроля групп ресурсов (control groups или cgroups) для изоляции и ограничения потребления ресурсов разными компонентами системы. Контрольные группы настраиваются в директории `/etc/cgconfig.d/` на каждом узле кластера. Каждая контрольная группа резервирует определенную долю физических ресурсов узла для конкретных задач.

По умолчанию в системе созданы следующие контрольные группы:

- `machine.slice` для расчета резерва ОЗУ;
- `storpool.slice` для РСХД;
- `user.slice` для пользовательских процессов;
- `system.slice` для системных служб операционной системы.

Система автоматического расчета резервирования отслеживает только те контрольные группы, которые указаны в переменной `SDC_CGROUPS_RESOURCES_MONITOR` в файле `/etc/sdc-env`.

Отслеживание контрольных групп по умолчанию

```
SDC_CGROUPS_RESOURCES_MONITOR="user,system,storpool"
```

Возможно ввести дополнительные группы, предварительно определив их конфигурацию в директории `/etc/cgconfig.d/`.

Отслеживание дополнительных контрольных групп

```
SDC_CGROUPS_RESOURCES_MONITOR="user,system,storpool,monitoring,security"
```

Внимание

После изменения конфигурации групп или переменной необходимо выполнить команду для перерасчета резервирования

```
aaa cluster resource reserved apply
```

АВТОМАТИЧЕСКОЕ РЕЗЕРВИРОВАНИЕ РЕСУРСОВ КЛАСТЕРА

Система вычисляет объем ЦПУ и ОЗУ, который необходимо зарезервировать на уровне кластера. При расчете учитываются доступные лимиты ресурсов на виртуализацию, заданные на уровне операционной системы.

Чтобы зарезервировать ресурсы для системных нужд:

1. Опционально посмотрите предварительный расчет ресурсов кластера, которые будут зарезервированы на всех узлах

```
aaa cluster resource reserved calc
```

Эта команда предназначена только для ознакомления с расчетами системы. Вы можете пропустить этот шаг и сразу перейти к применению настроек, так как система использует эти же внутренние расчеты для команды `apply`.

2. Чтобы применить рассчитанные системой значения резерва ресурсов, выполните команду

```
aaa cluster resource reserved apply
```

Важно

Данную команду необходимо выполнять каждый раз при изменении конфигурации контрольных групп или переменной отслеживания

РУЧНОЕ УПРАВЛЕНИЕ РЕЗЕРВИРОВАНИЕМ

Рекомендуется устанавливать резерв ресурсов под системные нужды, рассчитанный с помощью команд в разделе [Автоматическое резервирование ресурсов кластера](#). При необходимости Администратор может изменять данные значения, выполнив следующие действия:

Sharx Base 6.3. Руководство администратора

1. Просмотрите текущие настройки резерва ресурсов

```
aaa cluster resource reserved show --uuids *
```

где `uuids` — список идентификаторов узлов. При `uuids`, равном `*`, выведется информация по всем узлам. Чтобы получить информацию по конкретному узлу, введите его идентификатор.

2. Чтобы изменить настройки резерва ресурсов, введите

```
aaa cluster resource reserved update --uuid <UUID>
    [--reserved_system_cpu <RESERVED_SYSTEM_CPU>]
    [--reserved_system_ram <RESERVED_SYSTEM_RAM>]
    [--reserved_system_disk <RESERVED_SYSTEM_DISK>]
```

где

- `uuid` — идентификатор узла;
- `reserved_system_cpu` — резервирование ЦПУ для системных нужд на узле кластера;
- `reserved_system_ram` — резервирование оперативной памяти для системных нужд на узле кластера;
- `reserved_system_disk` — резервирование дискового пространства для системных нужд на узле кластера.

Чтобы удалить текущие параметры резервирования

```
aaa cluster resource reserved del --uuids <UUIDS>
```

10.2. Переподписка ресурсов

Переподписка позволяет виртуально выделить виртуальным машинам больше ресурсов, чем физически доступно в кластере.

Внимание

Изменение параметров переподписки возможно только когда в кластере не созданы ВЦОД (за исключением ВЦОД управления)

10.2.1. Переподписка ЦПУ

Переподписка ЦПУ — это метод оптимизации использования ресурсов, который позволяет виртуальным машинам использовать больше виртуальных ядер (ВЦПУ), чем имеется физических ядер (ЦПУ) в кластере.

Sharx Base 6.3. Руководство администратора

Основная формула перерасчета ВЦПУ в ЦПУ

```
CPU = VCPU/cpu_overcommit_ratio
```

Ключевой параметр `default_cpu_overcommit_ratio` — глобальный коэффициент переподписки в диапазоне от 1 до 4. Чем он выше, тем больше виртуальных ядер можно создать на одном физическом ядре. Этот коэффициент применяется ко всем VM по умолчанию, но может быть переопределен для каждой виртуальной машины индивидуально.

МЕХАНИЗМЫ ПЕРЕПОДПИСКИ ЦПУ

В Sharx Base доступны два механизма переподписки ЦПУ: тип доли `shares` и тип квоты `quotas`.

1. Тип доли `shares`.

Механизм доли `shares` определяет относительные доли процессорного времени, которые получают VM. Он не гарантирует абсолютное время ЦПУ, но обеспечивает оптимальное распределение процессорных ресурсов относительно других VM.

Формула расчета доли

```
shares = 4096/cpu_overcommit_ratio
```

Команда настройки

```
aaa cluster resource overcommit add --cpu_overcommit_type shares --  
default_cpu_overcommit_ratio {1-4}
```

где

- `cpu_overcommit_type` — тип переподписки ЦПУ. Установлено значение `share` (доли);
- `default_cpu_overcommit_ratio` — коэффициент переподписки ЦПУ в кластере по умолчанию. Возможные значения от 1 до 4.

2. Тип квоты `quotas`.

Механизм квоты `quotas` задает фиксированную квоту процессорного времени в микросекундах, которое ВЦПУ может использовать в течение определенного периода. Квоты обеспечивают более строгий и предсказуемый контроль в распределении ресурсов.

Чем выше коэффициент переподписки, тем ниже квота процессорного времени, т.е. допустимое время выполнения ВЦПУ VM.

Формула расчета квоты

```
cpu_quota = cpu_quota_period/cpu_overcommit_ratio
```

Команда настройки

Sharx Base 6.3. Руководство администратора

```
aaa cluster resource overcommit add --cpu_overcommit_type quotas --cpu_quota_period {10000-500000} --default_cpu_overcommit_ratio {1-4}
```

где

- `cpu_overcommit_type` – тип переподписки ЦПУ. Установлено значение `quotas` (квоты);
- `cpu_quota_period` – период, в течение которого отслеживается квота. Задается в микросекундах. Возможные значения от `10000` до `500000`;
- `default_cpu_overcommit_ratio` – коэффициент переподписки ЦПУ в кластере по умолчанию. Возможные значения от `1` до `4`.

КОМАНДЫ УПРАВЛЕНИЯ ПЕРЕПОДПИСКОЙ ЦПУ

Внимание

Изменение или удаление параметров переподписки возможно только когда в кластере не созданы ВЦОД (за исключением ВЦОД управления)

1. Чтобы проверить статус переподписки ЦПУ, введите

```
aaa cluster resource overcommit show
```

2. Обновить параметры переподписки ЦПУ

```
aaa cluster resource overcommit update --cpu_overcommit_type <CPU_OVERCOMMIT_TYPE>
--default_cpu_overcommit_ratio <DEFAULT_CPU_OVERCOMM
IT_RATIO>
```

3. Удалить параметры переподписки

Важно

Команда удаляет все параметры переподписки: и ЦПУ и ОЗУ.
Эта команда выполнится только в случае отсутствия ВЦОД в кластере

```
aaa cluster resource overcommit del
```

Примечание

Пример изменения коэффициента переподписки ЦПУ для определенной VM описан в **Руководстве пользователя в командной строке** в статье [Оптимизация ресурсов](#)

Sharx Base 6.3. Руководство администратора

СРАВНЕНИЕ МЕХАНИЗМОВ ПЕРЕПОДПИСКИ ЦПУ

Таблица – Сравнение механизмов переподписки ЦПУ.

Критерий	Тип <code>shares</code> доли	Тип <code>quotas</code> квоты
Принцип работы	Относительные доли процессорного времени	Абсолютная квота процессорного времени за период
Гибкость	Высокая	Ограниченная
Предсказуемость	Относительная, зависит от общей загрузки	Высокая, гарантированная квота
Рекомендуемые сценарии	Смешанные нагрузки, веб-сервисы	Приложения с жесткими требованиями к производительности. Например, базы данных

Рекомендации по выбору

Тип `shares` рекомендуется для рабочих нагрузок с переменной активностью, где важно относительное распределение ресурсов.

Тип `quotas` рекомендуется для нагрузок, требующих гарантированного времени ЦПУ и предсказуемой производительности.

Коэффициент переподписки `default_cpu_overcommit_ratio` следует выбирать в зависимости от ожидаемой нагрузки и требований к производительности.

10.2.2. Переподписка ОЗУ

Переподписка ОЗУ – технология увеличения доступной оперативной памяти для виртуальных машин через механизм `ballooning`, позволяющий VM делиться частью своей памяти с другими виртуальными машинами на том же узле.

Последовательность настройки переподписки ОЗУ:

1. Проверьте работу `sgroup`-экспортера.

Sharx Base 6.3. Руководство администратора

2. Настройте параметры переподписки ОЗУ.
3. Включите механизм переподписки ОЗУ.
4. Контролируйте работу кластера.

ТРЕБОВАНИЯ К СИСТЕМЕ ДЛЯ ПЕРЕПОДПИСКИ ОЗУ

Для работы переподписки ОЗУ необходимо выполнить следующие условия на каждом узле кластера:

1. В конфигурации плагина `sdc-plgn-metrics` должен быть настроен `sgroup`-экспортер. Чтобы проверить работу `sgroup`-экспортера, введите

```
scheduler report mem
```

Пример

В результате выполнения команды должен вернуться список ресурсов по каждому узлу

```
{
  "2a808504-a385-4856-aa14-f7c1240d81cb": {
    "CurrentMemory": "0Gb",
    "MaxMemory": "0Gb",
    "memballoon": "0Gb",
    "memtotal": "1315Mb",
    "memusage": "0Gb",
    "memusage_ballooned": "0Gb",
    "memusage_percent": 0
  },
  "336b18d5-9a2a-4ac5-90d2-6dc479ec709e": {
    "CurrentMemory": "0Gb",
    "MaxMemory": "0Gb",
    "memballoon": "0Gb",
    "memtotal": "1315Mb",
    "memusage": "6Mb",
    "memusage_ballooned": "6Mb",
    "memusage_percent": 0
  },
  "d4de5a59-e11d-49aa-990c-8ca589611fe8": {
    "CurrentMemory": "0Gb",
    "MaxMemory": "0Gb",
    "memballoon": "0Gb",
    "memtotal": "1315Mb",
    "memusage": "0Gb",
    "memusage_ballooned": "0Gb",
    "memusage_percent": 0
  }
}
```

Если `sgroup`-экспортер не настроен, то добавьте его в считывание командой

```
metrics exporter add --name sgroup_exporter --port 9198
```

После добавления экспортера проверьте его считывание

Sharx Base 6.3. Руководство администратора

```
scheduler report mem
```

2. На каждом узле в файле `/etc/sdc-env` переменные окружения имеют фактические значения. В примерах ниже показаны их значения по умолчанию, которые система использует автоматически.

```
# Имя sgroup-экспортера. По умолчанию sgroup_exporter
SDC_CGROUP_EXPORTER_NAME="cgroup_exporter"

# Порт sgroup-экспортера. По умолчанию 9198
SDC_CGROUP_EXPORTER_PORT="9198"
```

Примечание

Если вам необходимо изменить значения по умолчанию, вы можете явно задать их в файле `/etc/sdc-env`. Убедитесь, что заданные значения соответствуют фактической конфигурации экспортера

ПРИНЦИП РАБОТЫ И ОСНОВНЫЕ ПОНЯТИЯ

Основные понятия:

Балунинг `ballooning` — технология управления оперативной памятью в виртуализации, которая позволяет гипервизору временно «забирать» неиспользуемую память у одних виртуальных машин и перераспределять ее другим VM на том же физическом узле.

Порог срабатывания балунинга `ballooning_threshold` — уровень загрузки ОЗУ узла, с которого начинает работать механизм переподписки ОЗУ.

Лимит памяти узла `node_memory_limit` — максимальная загрузка ОЗУ узла, при достижении которой прекращается размещение новых VM.

Толерантность к переподписке `overcommit_tolerance` — уровень, определяющий объем памяти, которым может поделиться VM.

При включенной переподписке ОЗУ работает балунинг, при котором каждая виртуальная машина может отдавать часть своей памяти другим VM на том же узле.

Объем отдаваемой памяти зависит от показателей:

- Толерантности VM. Чем выше толерантность, тем больше памяти может отдать VM.
- Текущей загрузки узла. Чем выше загрузка, тем активнее работает балунинг.

Важно

При толерантности, равной 0, VM не делится своей памятью

НАСТРОЙКА ПЕРЕПОДПИСКИ ОЗУ

Действия для настройки переподписки ОЗУ:

1. Чтобы настроить переподписку ОЗУ, введите

```
aaa cluster resource overcommit update --ram_overcommit <RAM_OVERCOMMIT>
                                         --node_memory_limit <NODE_MEMORY_LIMIT>
                                         --default_overcommit_tolerance <DEFAULT_OVERCOMMIT_T
OLERANCE>
```

где

- `ram_overcommit` — процент переподписки ОЗУ;
- `node_memory_limit` — лимит использования ОЗУ узла в процентах;
- `default_overcommit_tolerance` — толерантность VM к переподписке. Возможные значения от 0 до 3.

В результате команды Sharx Base рассчитывает порог срабатывания балунинга `ballooning_threshold` по формуле

```
ballooning_threshold = 100 - ram_overcommit
```

Критические условия для назначения параметров переподписки ОЗУ:

- Для запуска балунинга необходимо выполнить условие `ballooning_threshold < node_memory_limit`.
- Разница между лимитами `node_memory_limit - 2% - ballooning_threshold ≥ 3%`.

Пример

```
aaa cluster resource overcommit update --ram_overcommit 40
                                         --node_memory_limit 98
                                         --default_overcommit_tolerance 0
```

В данном случае Sharx Base рассчитает

```
ballooning_threshold = 100 - 40 = 60%
```

Разница между лимитами

```
node_memory_limit - 2% - ballooning_threshold = 98% - 2% - 60% = 36%
```

Так как $36\% \geq 3\%$, параметры переподписки ОЗУ установлены корректно

Sharx Base 6.3. Руководство администратора

2. Чтобы обновить параметры переподписки ОЗУ, введите

```
aaa cluster resource overcommit update --ram_overcommit <RAM_OVERCOMMIT>
--node_memory_limit <NODE_MEMORY_LIMIT>
--default_overcommit_tolerance <DEFAULT_OVERCOMMIT_TOLERANCE>
```

3. Удалить параметры переподписки

Важно

Команда удаляет все параметры переподписки: и ЦПУ и ОЗУ.
Эта команда выполнится только в случае отсутствия ВЦОД в кластере

```
aaa cluster resource overcommit del
```

Внимание

Изменение или удаление параметров переподписки возможно только когда в кластере не созданы ВЦОД (за исключением ВЦОД управления)

УПРАВЛЕНИЕ ПЕРЕПОДПИСКОЙ ОЗУ

Внимание

Переподписку ОЗУ можно включить или выключить только из ВЦОД управления

1. Чтобы включить переподписку ОЗУ, введите

```
aaa cluster resource overcommit ballooning enable
```

2. Чтобы выключить переподписку ОЗУ, введите

```
aaa cluster resource overcommit ballooning disable
```

Внимание

Выключение переподписки возможно только при загрузенности ОЗУ каждого узла ниже лимита использования ОЗУ `node_memory_limit`

3. Чтобы проверить статус переподписки ОЗУ, введите

```
aaa cluster resource overcommit show
```

Sharx Base 6.3. Руководство администратора

После выполнения всех шагов система начнет автоматически перераспределять память между VM при достижении порога срабатывания балунинга на узлах.

Примечание

Пример изменения толерантности определенной VM к переподписке описан в **Руководстве пользователя в командной строке** в статье [Оптимизация ресурсов](#)

10.3. Просмотр ресурсов и управление узлами кластера

10.3.1. Просмотр доступных ресурсов

Чтобы посмотреть доступные ресурсы для ВЦОД, введите команду

```
aaa cluster resource show
```

Чтобы посмотреть ресурсы на каждом узле, введите

```
aaa cluster nodes resource show
```

10.3.2. Управление узлами кластера

Sharx Base позволяет расширить или уменьшить общие ресурсы логического кластера.

Чтобы управлять узлами кластера, выполните следующие шаги:

1. Просмотрите список доступных узлов

```
cluster freenodes list
```

2. Выберите узел, который будет добавлен в кластер

```
cluster nodes add --node <NODE>
```

где `node` – идентификатор узла.

3. Чтобы удалить узел из кластера, выполните следующую команду

```
cluster nodes del --node <NODE>
```

11. ВЦОД

ВЦОД – виртуальный центр обработки данных.

Sharx Base 6.3. Руководство администратора

ВЦОД может создавать **Администратор кластера**, когда находится во [ВЦОД управления](#).

11.1. Создать ВЦОД

✘ Внимание

Перед созданием ВЦОД настройте [резервирование и переподписку ресурсов](#)

1. Чтобы создать или инициализировать ВЦОД, введите в командной строке

```
aaa namespace add --cluster <CLUSTER>
                  --ns <NS>
                  --cpu <CPU>
                  --ram <RAM>
                  [--paths <PATHS>]
                  [--descr <DESCR>]
                  [--config_name <CONFIG_NAME>]
```

где

- `cluster` — имя кластера;
- `ns` — имя создаваемого ВЦОД;
- `cpu` — количество ЦПУ;
- `ram` — объем оперативной памяти. Указывается в мегабайтах Mb или гигабайтах Gb. Примеры: 2048Mb, 2Gb, 4Gb;
- `descr` — описание ВЦОД;
- `paths` — структура директорий ВЦОД в формате JSON. По умолчанию имеет значение "/" (корневая директория ВЦОД);
- `config_name` — имя конфигурационного файла. Возможные значения:
 - `fstek_virt` — конфигурация по требованиям ФСТЭК,
 - `cb_virt` — конфигурация по требованиям ЦБ.

Конфигурация ВЦОД может быть указана на уровне переменной окружения `CONFIG_ROLES_NAME`, тогда название конфигурационного файла указывать необязательно.

Пример paths

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

где

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

Важно

Для ВЦОД управления задайте значение ram, равное 0Gb, значение cpu, равное 0, так как ВЦОД управления предназначен только для настройки других ВЦОД, в нем будет отсутствовать виртуализация

2. Посмотреть информацию и статус ВЦОД

```
aaa namespace show --cluster <CLUSTER>  
--ns <NS>
```

3. Проверить наличие ВЦОД в общем списке

```
aaa namespace list --cluster <CLUSTER>
```

Примечание

При необходимости [настройте метки ВЦОД](#)

11.2. Удалить ВЦОД

Перед удалением ВЦОД сначала проверьте, что в нем нет объектов, правил, пользователей и других связанных объектов. После проверки для удаления ВЦОД введите команду

```
aaa namespace del --cluster <CLUSTER>  
--ns <NS>
```

Если во ВЦОД есть объекты, и вы уверены в удалении, используйте флаг force, указав в нем имя удаляемого ВЦОД

Внимание

Флаг --force приводит к полному удалению всех объектов ВЦОД

```
aaa namespace del --cluster <CLUSTER>
                  --ns <NS>
                  --force <NS>
```

Пример с force

```
aaa namespace del --cluster cluster-test-version
                  --ns ns-test
                  --force ns-test
```

12. Хранилище

12.1. Типы хранилищ

Sharx Base позволяет работать с тремя типами хранилища:

1. **Libvirt.**
2. **РСХД.**
3. **NFS.**

Тип хранилища выбирают с учетом требований к производительности, отказоустойчивости, масштабируемости и особенностей инфраструктуры.

Подготовку хранилища выполняет Администратор кластера в зависимости от выбранного типа.

12.1.1. Краткое описание типов хранилищ

Libvirt – локальное хранилище на дисках узла. Не требует дополнительной сетевой инфраструктуры, простое в настройке и использовании.

Пулы Libvirt создаются Администратором кластера и закрепляются за конкретным ВЦОД. В рамках ВЦОД пользователи создают тома, которые используются как диски виртуальных машин.

Когда использовать Libvirt:

- тестовые стенды;
- ВМ с невысокой критичностью;
- сценарии без требований к миграции и высокой доступности.

Sharx Base 6.3. Руководство администратора

РСХД – распределенная система хранения данных, которая объединяет напрямую подключенные хранилища стандартных серверов в единое пространство хранения. Поддерживает репликацию, масштабирование и высокую доступность.

Администратор кластера настраивает **группы размещения и шаблоны**. На основе шаблонов пользователи создают тома для виртуальных машин.

Когда использовать РСХД:

- промышленные кластеры;
- системы с высокими требованиями к доступности;
- среды с требованиями к отказоустойчивости и сохранности данных;
- виртуальные среды, где важна производительность и масштабируемость.

NFS-хранилище – сетевое файловое хранилище. Обеспечивает доступ к файлам, расположенным на удаленных серверах, и позволяет работать с этими файлами точно так же, как и с локальными. NFS-хранилище простое в настройке, поддерживает миграцию VM. Зависит от качества сетевой инфраструктуры и конфигурации NFS-сервера.

Когда использовать NFS:

- для хранения образов VM, ISO, резервных копий;
- в небольших и средних кластерах, где отказоустойчивость можно обеспечить средствами NFS-сервера;
- для VM со средними нагрузками.

Ограничения:

- зависит от качества сетевой инфраструктуры;
- возможны ограничения на операции со снимками и миграцией виртуальных машин;
- производительность зависит от конфигурации NFS-сервера.

12.1.2. Этапы работы с хранилищами

Работа с хранилищами в Sharx Base разделена по ролям:

- **Администратор кластера** настраивает хранилище: пулы, шаблоны, подключения;
- **Администратор ВЦОД** и **Разработчик VM** создают и используют тома.

Дальнейшие действия с хранилищами для Администратора кластера описаны в данном руководстве.

12.2. Libvirt. Пулы

Работа с Libvirt организована следующим образом:

1. Администратор кластера создает пул.
2. Пул закрепляется за конкретным ВЦОД.
3. В рамках ВЦОД Администратор ВЦОД или Разработчик ВМ создает тома.
4. Тома используются как диски виртуальных машин.

Пул – это логическая единица хранилища, в рамках которой создаются тома виртуальных машин. Пулы делятся на тома, которые назначаются виртуальным машинам в качестве дисков.

✘ Внимание

Пул, прикрепленный к конкретному ВЦОД, недоступен из другого ВЦОД

12.2.1. Создать пул

Чтобы создать пул, введите в командной строке

```
storage libvirt pool add --name <NAME>
                        --capacity <CAPACITY>
                        --ns <NS>
                        [--descr <DESCR>]
```

где

- `name` – имя создаваемого пула;
- `capacity` – объем пула в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `ns` – имя ВЦОД, к которому подключен пул;
- `descr` – описание пула.

На физическом узле пул появится в директории `/pools/<vcod_name>/<pool_name>`.

12.2.2. Работа с пулами

1. Обновить конфигурацию пула

```
storage libvirt pool update --name <NAME>
                            [--capacity <CAPACITY>]
                            [--descr <DESCR>]
```

где

Sharx Base 6.3. Руководство администратора

- `name` — имя пула;
- `capacity` — объем пула в мегабайтах или гигабайтах. Число и единицы измерения указываются слитно в любом регистре. Например: `100MB`, `200Mb`, `300Gb`, `400gb`;
- `descr` — описание пула.

2. Просмотреть информацию и статус определенного пула

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

где

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

3. Просмотреть список всех пулов

```
storage libvirt pool list [--ns <NS>]
```

где `ns` — имя ВЦОД.

4. Просмотреть права доступа пользователей к пулам

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

5. Просмотреть доступное количество ресурсов для создания пулов

```
storage libvirt pool resources show
```

6. Удалить неиспользуемый пул

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

где

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

7. Пул может зависнуть в статусе *Pending* (Ожидание).

В таких случаях удалите его из базы данных командой

Sharx Base 6.3. Руководство администратора

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

где

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

После этого создайте пул 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

12.2.3. Дальнейшая работа

После создания пула Администратор ВЦОД и Разработчик ВМ могут создавать тома Libvirt в рамках соответствующего ВЦОД. Подробная информация приведена в **Руководстве пользователя в командной строке** в статье [Создать том](#).

При первичной настройке системы рекомендуется создавать пул после настройки ВЦОД. В дальнейшем можно изменять пулы в любой необходимый момент.

Правами на создание и управление пулами обладает **Администратор кластера**.

Правами на создание и использование томов обладают **Администратор ВЦОД** и **Разработчик ВМ**.

12.3. РСХД. Настроить лимиты использования ресурсов

В статье описана только настройка объектов РСХД: групп хранения и шаблонов, так как эти действия выполняет **Администратор кластера**. Остальные шаги настройки рассмотрены в

Лимиты использования ресурсов в хранилище РСХД — это ограничения, которые устанавливаются для управления и контроля потребления ресурсов системы хранения данных.

В рамках лимитов использования создаются **группы хранения** — наборы дисков, на которые будут реплицироваться объекты. Например, наборы по типу SSD.

Затем на основе групп хранения необходимо создать **шаблоны**, которые определяют

Sharx Base 6.3. Руководство администратора

параметры для создания томов в РСХД. Шаблоны позволяют упростить процесс создания новых томов, обеспечивая быстрое и единообразное развертывание томов с заданными характеристиками.

✘ Внимание

Только Администратор кластера имеет права на настройку лимитов использования ресурсов в хранилище РСХД

12.3.1. Резервировать ресурсы

Резерв ресурсов происходит через создание и управление группами размещений.

1. Чтобы просмотреть созданные группы размещения, введите команду

```
storage sp placementGroup list
```

🧪 Пример

Пример вывода списка групп размещения

```
json [ "ssd" ]
```

2. Создать группу размещения

```
storage sp placementGroup resources reserved add --place <ssd> --reserved <value>
```

где

- `place` — имя группы размещения;
- `reserved` — количество резервируемой физической памяти.

3. Просмотреть ресурсы, доступные для групп размещения

```
storage sp placementGroup resources show --place <name_place>
```

Пример

Пример вывода доступных ресурсов для групп размещения

```
{
  "allocated": "100Gb",
  "available": "175Gb",
  "limit": "275Gb",
  "placementGroup": "ssd"
}
```

4. Изменить количество резервируемой физической памяти у группы размещения

```
storage sp placementGroup resources reserved update --place <name_place> [--reserved <value>]
```

5. Удалить группу размещения

```
storage sp placementGroup resources reserved del --place <name_place>
```

12.3.2. Создать шаблон

Внимание

Шаблоны создаются только во ВЦОД управления с учетом прав доступа.

Возможно создать шаблон только для существующего ВЦОД

Шаблон создается **Администратором кластера** для использования в рамках определенного ВЦОД.

1. Чтобы создать шаблон, введите команду

```
storage sp template add [--bw <BW>
                        [--iops <IOPS>]
                        [--descr <DESCR>]
                        --name <name_template>
                        [--parent <PARENT>]
                        --place <name_place>
                        --replication <rf_quantity>
                        --capacity <volume_capacity>
                        --ns <namespace_name>
```

где

- `bw` – ограничение пропускной способности в килобайтах;
- `iops` – ограничение количества операций ввода-вывода в секунду;

Sharx Base 6.3. Руководство администратора

- `descr` — описание шаблона;
- `name` — имя шаблона;
- `parent` — имя снимка, на основе которого будут созданы тома;
- `place` — имя группы размещения;
- `replication` — фактор репликации;
- `capacity` — ограничение размера шаблона;
- `ns` — имя ВЦОД.

Важно

`capacity`, умноженная на `replication`, должен быть меньше или равен доступному ресурсу группы размещения

12.3.3. Команды для работы с шаблонами

1. Просмотреть список созданных шаблонов

```
storage sp template list
```

2. Просмотреть подробную информацию о шаблоне

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

где `name` — имя шаблона.

3. Обновить параметры шаблона

```
storage sp template update [--bw <BW>]
                             [--iops <IOPS>]
                             [--descr <DESCR>]
                             --name <name_template>
                             [--parent <PARENT>]
                             [--replication <rf_quantity>]
                             [--reuse_server <REUSE_SERVER>]
                             [--capacity <volume_capacity>]
                             [--ns <namespace_name>]
```

где

- `bw` — ограничение пропускной способности в килобайтах;
- `iops` — ограничение количества операций ввода-вывода в секунду;
- `descr` — описание шаблона;
- `name` — имя шаблона;

Sharx Base 6.3. Руководство администратора

- `parent` — имя снимка, на основе которого будут созданы тома;
- `replication` — количество копий для репликации;
- `reuse_server` — разрешение размещения реплик на одном сервере;
- `capacity` — ограничение размера шаблона;
- `ns` — имя ВЦОД.

4. Просмотреть список шаблонов с лимитами и доступными ресурсами

```
storage sp template limit list
```

5. Шаблон может зависнуть в статусе *Pending (Ожидание)*. В таких случаях удалите запись о нем из базы данных командой

```
storage sp template clear --name <name_template>
```

6. Удалить шаблон

```
storage sp template del --name <name_template>
```

После добавления шаблона во ВЦОД **Администратор ВЦОД** может [создавать тома РСХД](#). При первой настройке системы рекомендуется подготавливать тома после создания виртуального кластера. При последующей работе в уже настроенной системе создавать и работать с томами можно в любой необходимый момент времени.

Правами на создание и работу с томами обладают **Администратор ВЦОД** и **Разработчик ВМ**.

12.4. NFS. Настроить хранилище

1. Добавьте информацию о новом подключении

Примечание

Команда производит только запись о подключении в БД. Монтирования при ее выполнении не происходит

```
storage nfs mount add --name <NAME>
                    --nfs_server <NFS_SERVER>
                    --shared_folder_path <SHARED_FOLDER_PATH>
                    [--mount_point <MOUNT_POINT>]
                    [--params <PARAMS>]
                    --nodes <NODES>
                    --ns <NS>
                    [--autocheck <AUTOCHECK>]
```

Sharx Base 6.3. Руководство администратора

```
[--check_timeout <CHECK_TIMEOUT>]
[--descr <DESCR>]
```

где

- `name` — название подключения;
- `nfs_server` — IP-адрес сервера NFS;
- `shared_folder_path` — директория на сервере, которая будет смонтирована;
- `mount_point` — точка монтирования на одном или нескольких узлах. Если не указана, то формируется автоматически как `/nfs_shares/<cluster>/<ns>/<name>`, если в параметре `ns` указан определенный ВЦОД, и `/nfs_shares/<cluster>/<name>`, если в параметре `ns` указаны все ВЦОД;
- `params` — параметры, которые должны быть записаны в файл `/etc/fstab`. Если не указаны, то автоматически используется строка `defaults`;
- `ns` — список ВЦОД, которые должны иметь доступ к этому подключению. Можно указать конкретные ВЦОД или все сразу;
- `nodes` — список узлов, на которые нужно будет монтировать директорию;
- `autocheck` — нужно ли будет проводить автоматическую проверку доступности для этого подключения после монтирования;
- `check_timeout` — с какой периодичностью нужно проводить автоматическую проверку. Можно указать любое значение между 60 и 300 секундами. Значение по умолчанию — 300;
- `descr` — описание подключения.

2. Просмотрите список всех подключений

```
storage nfs mount list
```

3. Чтобы просмотреть подробную информацию об определенном подключении, введите

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

где `uuid` — идентификатор подключения из БД.

4. Выполните монтирование директории на узлы

```
storage nfs mount connect --uuid <UUID>
```

где `uuid` — идентификатор подключения из БД.

Монтирование производится сразу на все узлы, указанные при создании подключения. На каждом узле в файл `/etc/fstab` записывается информация о подключении и выполняется команда `mount <mount_point>`.

5. Чтобы обновить настройки подключения, введите

✘ Внимание

Команду обновления можно выполнять только для размонтированного подключения

```
storage nfs mount update --uuid <UUID>
                        [--name <NAME>]
                        [--nfs_server <NFS_SERVER>]
                        [--shared_folder_path <SHARED_FOLDER_PATH>]
                        [--mount_point <MOUNT_POINT>]
                        [--params <PARAMS>]
                        [--autocheck <AUTOCHECK>]
                        [--check_timeout <CHECK_TIMEOUT>]
                        [--descr <DESCR>]
```

где

- `uuid` — идентификатор подключения из БД;
- `name` — название подключения;
- `nfs_server` — IP-адрес сервера NFS;
- `shared_folder_path` — директория на сервере, которая будет смонтирована;
- `mount_point` — точка монтирования на одном или нескольких узлах;
- `params` — параметры, которые должны быть записаны в файл `/etc/fstab`;
- `autocheck` — нужно ли будет проводить автоматическую проверку доступности для этого подключения после монтирования;
- `check_timeout` — с какой периодичностью нужно проводить автоматическую проверку. Можно указать любое значение между 60 и 300 секундами. Значение по умолчанию — 300;
- `descr` — описание подключения.

6. Добавить новые узлы к уже смонтированному подключению

```
storage nfs mount nodes add --nodes <NODES>
                             --uuid <UUID>
```

где

- `nodes` — список узлов, которые нужно добавить;
- `uuid` — идентификатор подключения из БД.

При выполнении команды монтирование директории на новых узлах выполнится автоматически.

7. Удаление узлов из подключения

✘ Внимание

Команду можно выполнить только для несмонтированного подключения

Sharx Base 6.3. Руководство администратора

```
storage nfs mount nodes del --nodes <NODES>
                                --uuid <UUID>
```

где

- `nodes` — список узлов, которые нужно удалить;
- `uuid` — идентификатор подключения из БД.

8. Тестировать доступность подключения

```
storage nfs mount test --uuid <UUID>
                        [--node <NODES>]
```

где

- `uuid` — идентификатор подключения из БД;
- `node` — узел, на котором нужно провести проверку. Если узел не указан, тест выполнится на всех подключенных узлах.

Тест подключения на каждом узле состоит из двух этапов: выполнения команды `findmnt <mount_point>` и теста на запись в точке монтирования `timeout 10 dd if=/dev/zero bs=4096 count=20 > <mount_point>/test-<node_uuid>`.

Статус подключения и ошибки выполнения теста сохраняются в БД.

Если при добавлении подключения был указан параметр `autocheck`, то тест подключения будет выполняться автоматически с указанной периодичностью в `check_timeout`.

9. Размонтировать подключение

Внимание

Подключение можно размонтировать только после удаления всех ВЦОД из него

```
storage nfs mount disconnect --uuid <UUID>
```

где `uuid` — идентификатор подключения из БД.

Размонтирование происходит на всех подключенных узлах сразу. На каждом узле из файла `/etc/fstab` удаляется запись о подключении и выполняется команда `umount <mount_point>`.

10. Удалить подключение из БД

Внимание

Подключение можно удалить только после размонтирования

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

Sharx Base 6.3. Руководство администратора

где `uuid` — идентификатор подключения из БД.

11. Добавить новые ВЦОД к подключению

```
storage nfs mount ns add --ns <NS>
                        --uuid <UUID>
```

где

- `uuid` — идентификатор подключения из БД;
- `ns` — список ВЦОД, которые нужно добавить.

12. Удалить ВЦОД из подключения

Внимание

Команду можно выполнить только для несмонтированного подключения

```
storage nfs mount ns del --ns <NS>
                        --uuid <UUID>
```

где

- `uuid` — идентификатор подключения из БД;
- `ns` — список ВЦОД, которые нужно удалить.

Примечание

Работа с NFS-томами описана в [Руководстве пользователя в командной строке](#)

13. Структура ВЦОД

Ресурсы логического кластера разделяются на виртуальные единицы: множество ВЦОД — виртуальных центров обработки данных (см. статью [Общие сведения](#)).

По умолчанию у ВЦОД управления имеется корневая директория, в которой размещены **Администратор кластера** и **Администратор безопасности кластера**.

Далее в пределах ВЦОД управления создаются поддиректории, изолированные друг от друга. При этом Администратор кластера, находясь в корневой директории ВЦОД управления, имеет доступ ко всем ресурсам пользователей, находящимся в поддиректориях ВЦОД управления.

Директории ВЦОД управления — это средство управления доступом пользователей к объектам, находящимся внутри этих директорий.

Примечание

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

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

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.

14. Уведомления

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**.

14.1. Создать почтовый сервер

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>
```

14.2. 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**.

Sharx Base 6.3. Руководство администратора

2. Запрос информации о всех адресах

```
notif mailing list
```

3. Запрос информации об определенном адресе

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

14.3. 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>
```

14.4. 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>
```

14.5. 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 АРМ 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.

14.6. Контроль отправленных сообщений

Sharx Base 6.3. Руководство администратора

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.

15. Права доступа и роли

15.1. 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:

- **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.

15.2. 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.

Во ВЦОД управления по умолчанию создаются две роли: администратор кластера и администратор безопасности кластера.

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.

15.2.1. Роли по умолчанию

АДМИНИСТРАТОР КЛАСТЕРА

Функции **Администратора кластера**:

- настройка параметров и операции с программными модулями (плагины) Sharx Base;
- настройка параметров безопасности и интеграции Sharx Base с каталогом LDAP;
- настройка ролей во ВЦОД управления;
- делегирование и отзыв прав доступа во ВЦОД;
- управление сессиями доступа во ВЦОД управления;
- создание, удаление, настройка параметров учетных записей пользователей ВЦОД управления;
- настройка параметров использования ресурсов кластера;
- просмотр журналов и конфигурационных параметров журналирования событий Sharx Base;

Sharx Base 6.3. Руководство администратора

- настройка параметров и выполнение операций с сетевым стеком и виртуальными сетями платформы виртуализации;
- настройка параметров и выполнение операций с журналами событий узлов кластера;
- настройка параметров механизма уведомлений и рассылок Sharx Base;
- перевод, удаление узлов кластера из сервисного режима;
- выполнение операций с метками и их привязка к ВЦОД;
- настройка параметров и выполнение операций с ключами и сертификатами;
- настройка параметров работы Sharx Base с СХД (PCXD, Libvirt, NFS);
- настройка параметров и выполнение операций по обновлению ПО Sharx Base;
- настройка параметров и выполнение операции с лицензиями;
- просмотр параметров и результатов операций контроля целостности;
- настройка параметров и выполнение операций мониторинга.

АДМИНИСТРАТОР БЕЗОПАСНОСТИ КЛАСТЕРА

Функции **Администратора безопасности кластера:**

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

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;

Sharx Base 6.3. Руководство администратора

- 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 ВМ (включение, выключение, перезагрузка);
- выполнение операций со снимками ВМ 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 ВМ, снимки ВМ, копии дисков);
- выполнение операций мониторинга ВЦОД;
- доступ к VNC-консоли ВМ.

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;

Sharx Base 6.3. Руководство администратора

- 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.

15.2.2. Действия с ролями

Администратор кластера может управлять ролями в рамках ВЦОД управления.

Роль имеет следующие параметры:

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

Sharx Base 6.3. Руководство администратора

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

16. Пользователи ВЦОД управления

Важно

При передаче пользователю контроля над ВЦОД управления формируется специальный пользователь Технической поддержки ООО «ШарксДЦ»: **sharx_tech**. У него точно такие же права, как и у Администратора кластера

Во ВЦОД управления по умолчанию созданы два пользователя на основе ролей **Администратор кластера** и **Администратор безопасности кластера**. При необходимости можно создать дополнительных пользователей.

16.1. Создать пользователя

Примечание

В статье описано ручное создание пользователя.

Также Sharx Base поддерживает создание пользователей с использованием LDAP. Подробная инструкция описана в [Руководстве пользователя в командной строке](#).

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;

Sharx Base 6.3. Руководство администратора

- `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>
```

16.2. 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

```
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`

Sharx Base 6.3. Руководство администратора

```
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

16.3. Ограничить время активности пользователя

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

16.4. 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*

17. 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

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;
- `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

Sharx Base 6.3. Руководство администратора

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.

18. 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. В данном случае – это ВЦОД управления

18.1. Команды управления базовыми параметрами механизмов безопасности

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

```
aaa param show
```

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>
```

18.2. 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>

Sharx Base 6.3. Руководство администратора

Параметр ВЦОД	Описание
<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. Допустимые значения – целочисленные. Диапазон 0, далее от 10 до 365, где 0 – отключение проверки
<code>acc_delete_days</code>	Количество дней после удаления пользователя, по истечении которых вся информация о нем будет окончательно удалена из системы. По умолчанию – 1095 дней
<code>auth_type</code>	Тип аутентификации. Возможные значения: <ul style="list-style-type: none">• <i>BASIC</i> – логин и пароль (значение по умолчанию),• <i>TFA</i> – двухфакторная аутентификация
<code>cert</code>	Использование сертификата при аутентификации. Возможные значения: <ul style="list-style-type: none">• <i>no</i> – не использовать (значение по умолчанию),• <i>yes</i> – использовать
<code>ldap</code>	Имя конфигурации LDAP , подключаемого к Sharx Base

Sharx Base 6.3. Руководство администратора

Параметр ВЦОД	Описание
<code>ldap_sync</code>	Включение синхронизации с LDAP -сервером. Возможные значения: <ul style="list-style-type: none">• <i>no</i> – синхронизация выключена (значение по умолчанию),• <i>yes</i> – синхронизация включена
<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. Маршрут отправки уведомлений
<code>ns_owner_access</code>	Разрешение входа во ВЦОД Администратору кластера. Возможные значения: <ul style="list-style-type: none">• <i>yes</i> – разрешен (значение по умолчанию),• <i>no</i> – не разрешен
<code>otp_code_live_period_years</code>	Срок действия OTP-ключа, выраженный в годах. Возможные значения – любое положительное целое число. По умолчанию – 3 года
<code>password_diff_cnt</code>	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</code>
<code>password_exp_days</code>	Срок действия пароля. По умолчанию – 60 дней. 0 – отключение проверки
<code>password_min_change</code>	Минимальное количество символов, которое надо изменить при замене пароля. По умолчанию – 4

Sharx Base 6.3. Руководство администратора

Параметр ВЦОД	Описание
<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: <ul style="list-style-type: none">• <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> . Допустимые значения – целочисленные. Диапазон от 0 до 10, где 0 – отключение проверки
<code>sessions_timeout_sec</code>	Время отключения сессии при бездействии. По умолчанию – <i>300 с (5 мин)</i> . Допустимые значения – целочисленные. Диапазон 0, далее от 180 с (3 мин) до 43 200 с (12 ч), где 0 – отключение проверки
<code>sessions_multi_origin</code>	Определяет, разрешено ли создание нескольких параллельных сессий для одного пользователя с разных IP-адресов. Возможные значения: <ul style="list-style-type: none">• <i>yes</i> – разрешено (значение по умолчанию),• <i>no</i> – запрещено. Примечание: При смене значения с <i>yes</i> на <i>no</i> уже существующие сессии с разными IP-адресами не будут автоматически разорваны
<code>sessions_timeout_sec</code>	Время отключения сессии при бездействии. По умолчанию – <i>300 с (5 мин)</i> . Допустимые значения – целочисленные. Диапазон

Sharx Base 6.3. Руководство администратора

Параметр ВЦОД	Описание
	0, далее от 300 с (5 мин) до 43 200 с (12 ч), где 0 – отключение проверки
<code>tfa_client</code>	Клиент для двухфакторной аутентификации. По умолчанию – <i>OTP</i>
<code>tfa_wait_sec</code>	Таймаут для двухфакторной аутентификации По умолчанию – <i>60</i> с
<code>validation_ip</code>	Проверка IP-адреса пользователя. Возможные значения: <ul style="list-style-type: none">• <i>no</i> – выключена (значение по умолчанию),• <i>yes</i> – включена
<code>whitelist_networks</code>	Список белых адресов, с которых можно подключаться к Sharx Base

18.3. 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>	Количество попыток ввода некорректного логина и пароля. По умолчанию – <i>4</i> . 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>
<code>acc_block_try_suspend_sec_privileged</code>	

Sharx Base 6.3. Руководство администратора

Параметр ВЦОД	Описание
<code>acc_block_try_timeout_sec_privileged</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_unused_days_privileged</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_privileged</code>	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
<code>auth_type_privileged</code>	Тип аутентификации. Возможные значения: • <i>BASIC</i> – логин и пароль (значение по умолчанию), • <i>TFA</i> – двухфакторная аутентификация
<code>cert_privileged</code>	Использование сертификата при аутентификации. Возможные значения: • <i>no</i> – не использовать (значение по умолчанию), • <i>yes</i> – использовать
<code>password_diff_cnt_privileged</code>	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

Sharx Base 6.3. Руководство администратора

Параметр ВЦОД	Описание
	отключается проверка <code>password_min_exp_days_privileged</code>
<code>password_exp_days_privileged</code>	Срок действия пароля. По умолчанию – <i>60 дней</i> . 0 – отключение проверки
<code>password_min_change_privileged</code>	Минимальное количество символов, которое надо изменить при замене пароля. По умолчанию – <i>4</i>
<code>password_min_exp_days_privileged</code>	Минимальный срок, который должен пройти после установки пароля, прежде чем его можно будет сменить. По умолчанию – <i>1 день</i>
<code>password_pattern_privileged</code>	Сложность пароля Минимум 8 символов, включающих прописные буквы, строчные буквы, цифры, специальные знаки
<code>sessions_max_cnt_privileged</code>	Максимальное количество сессий пользователя. По умолчанию – <i>2</i> . 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 – <i>300 с (5 мин)</i> . Допустимые значения – целочисленные. Диапазон 0, далее от 180 с (3 мин) до 43 200 с (12 ч), где 0 – отключение проверки

19. Валидация объектов ВЦОД

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

Sharx Base 6.3. Руководство администратора

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.

19.1. Управление ключами ВЦОД управления

1. Чтобы посмотреть список ключей во ВЦОД управления, 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.

19.2. 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 кластера** 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

Sharx Base 6.3. Руководство администратора

```
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

19.3. 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:

Sharx Base 6.3. Руководство администратора

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.

19.4. 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

Sharx Base 6.3. Руководство администратора

- `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 `sdс-plgn-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

Sharx Base 6.3. Руководство администратора

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>
```

Sharx Base 6.3. Руководство администратора

где `model` — имя модели. Обязательный аргумент.

b. 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
```

c. 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.

a. Чтобы добавить объект в список исключений, введите команду

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

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

b. 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

19.5. Оповещение пользователя о событиях КЦ

Платформа позволяет настроить оповещение Администратора кластера 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 Sharx Base](#).

19.6. Рекомендации по действиям Администратора кластера 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»

20. Резервное копирование данных кластера

Sharx Base позволяет настроить резервное копирование логов, настроек и данных кластера.

Основные логи находятся в директории `/var/lib/sharx/logs/`.

База данных, содержащая описание кластера, находится в директории `/var/lib/sharx/db/`.

Логи **Cassandra** находятся в `/var/log/cassandra/`.

20.1. Настроить резервное копирование кластера

Чтобы настроить резервное копирование, выполните следующие шаги:

1. Предварительно настройте параметры экспорта командой

```
logdump export param add [--pre_exec <PRE_EXEC>]
                        [--post_exec <POST_EXEC>]
                        [--dirs <DIRS>]
                        [--files <FILES>]
```

Sharx Base 6.3. Руководство администратора

```
[--db <DB>]
[--sosreport <SOSREPORT>]
[--path <PATH>]
[--passwd <PASSWD>]
```

где

- `pre_exec` — команда для выполнения до операции экспорта;
- `post_exec` — команда для выполнения после операции экспорта;
- `dirs` — список директорий, которые будут добавлены в ZIP-архив для экспорта. Задается в виде JSON-строки;
- `files` — список файлов, которые будут добавлены в ZIP-архив для экспорта. Задается в виде JSON-строки;
- `db` — экспорт всех данных из БД Cassandra.
Возможные значения:
 - `yes` — экспорт выполняется,
 - `no` — экспорт не выполняется;
- `sosreport` — экспорт в архив различных журналов: веб-сервера, событий узлов кластера, логов отладки и РСХД, распределенной БД, ОС, плагинов ядра.
Возможные значения:
 - `yes` — экспорт выполняется,
 - `no` — экспорт не выполняется;
- `path` — путь к директории, в которую будет экспортирован архив. По умолчанию — директория `/var/tmp/`;
- `passwd` — пароль для распаковки архива.

2. Чтобы посмотреть параметры экспорта в пределах ВЦОД, введите команду

```
logdump export param show
```

3. Обновить параметры

```
logdump export param update [--pre_exec <PRE_EXEC>]
                             [--post_exec <POST_EXEC>]
                             [--dirs <DIRS>]
                             [--files <FILES>]
                             [--db <DB>]
                             [--sosreport <SOSREPORT>]
                             [--path <PATH>]
                             [--passwd <PASSWD>]
```

4. После определения параметров выполните команду экспорта

```
logdump export add --node <UUID>
                  [--pre_exec <PRE_EXEC>]
                  [--post_exec <POST_EXEC>]
```

Sharx Base 6.3. Руководство администратора

```
[--dirs <DIRS>]
[--files <FILES>]
[--db <DB>]
[--sosreport <SOSREPORT>]
[--path <PATH>]
[--passwd <PASSWD>]
[--delete_after <DELETE_AFTER>]
```

где

- `node` — идентификатор узла;
- `pre_exec` — команда для выполнения до операции экспорта;
- `post_exec` — команда для выполнения после операции экспорта;
- `dirs` — список директорий, которые будут добавлены в ZIP-архив для экспорта. Задается в виде JSON-строки;
- `files` — список файлов, которые будут добавлены в ZIP-архив для экспорта. Задается в виде JSON-строки;
- `db` — экспорт всех данных из БД Cassandra.
Возможные значения:
 - `yes` — экспорт выполняется,
 - `no` — экспорт не выполняется;
- `sosreport` — экспорт в архив различных журналов: веб-сервера, событий узлов кластера, логов отладки и РСХД, распределенной БД, ОС, плагинов ядра.
Возможные значения:
 - `yes` — экспорт выполняется,
 - `no` — экспорт не выполняется;
- `path` — путь к директории, в которую будет экспортирован архив. По умолчанию — директория `/var/tmp/`;
- `passwd` — пароль для распаковки архива;
- `delete_after` — дата удаления архива. Формат: `YYYY-MM-DD` или `n`, где `n` — целое число, которое обозначает количество дней, через которые архив будет удален. Например, 2, 3 и т.д.

При корректном экспорте в директории `/var/tmp/` появится ZIP-архив с соответствующим именем узла, с которого был произведен экспорт журнала. Формат имени ZIP-архива `uuid.zip`

5. Просмотр списка всех выполненных операций экспорта

```
logdump export list
```

6. Просмотр определенной операции экспорта

Sharx Base 6.3. Руководство администратора

```
bash
```

```
logdump export show --uuid <export_uuid>  
[--node <NODE>]
```

где

- `uuid` — идентификатор операции;
- `node` — идентификатор узла.

Важно

Импорт резервных копий на другой кластер выполняет техническая поддержка Sharx Base

20.2. Сбор логов по операции

После выполнения команд в конце вывода или ответа всегда присутствует трейс-токен, имеющий вид `1efd1c0f-aa18-6c96-bc4a-f6d36bd9b3a5`.

По трейс-токену можно собрать в архив логи всех плагинов, которые были затронуты в процессе вызова операции.

Чтобы собрать логи в архив, выполните команду

```
logdump trace --trace_token <token>
```

где `trace_token` - имя трейс-токена вида `1efd1c0f-aa18-6c96-bc4a-f6d36bd9b3a5`.

Процесс сборки архива будет отображен в общем перечне архивов логдамп. После успешного завершения процесса на узле появится архив в директории `/var/tmp`.

Примечание

Данная команда доступна только **Администратору кластера**.
При возникновении ошибки во ВЦОД **Администратор ВЦОД** должен передать токен Администратору кластера, чтобы он собрал архив для изучения технической поддержкой

21. Журнал безопасности

Примечание

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

Sharx Base 6.3. Руководство администратора

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.](#)
3. Архивировать журнал безопасности:
 - [настроить регулярный процесс автоматического архивирования;](#)
 - [принудительно архивировать журнал событий.](#)

21.1. 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 <event_group></code>	Отображение событий в пределах указанной группы
	Уровень событий

Параметр	Функции параметра
<code>level</code> <INFO, WARNING, CRITICAL>	
<code>limit</code> <number_events>	Отображение заданного количества событий
<code>login</code> <user_login>	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
```

21.2. Настроить параметры записи событий

21.2.1. 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

Sharx Base 6.3. Руководство администратора

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

4. Удалить параметры

```
aaaevents param del
```

21.2.2. 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.

21.3. Архивировать журнал безопасности

Архивирование позволяет сохранить данные на длительный срок, оптимизировать пространство для хранения, а также обеспечить безопасность и восстановление данных.

После архивирования журнал безопасности очищается, информация о зарегистрированных событиях находится в архиве. Новые события, происходящие после архивирования, дальше записываются в журнал безопасности до следующего запуска процесса архивирования.

Sharx Base позволяет настроить:

- [Регулярное архивирование](#), которое будет выполняться автоматически по заданным параметрам без участия администратора.
- [Ручное архивирование журнала безопасности](#), инициируемое администратором.

21.3.1. Настроить регулярное архивирование журнала безопасности

Чтобы настроить регулярный процесс архивирования журнала безопасности, необходимо задать параметры архивирования и указать маршрут отправки уведомлений о занятом месте в архиве.

Sharx Base 6.3. Руководство администратора

ПАРАМЕТРЫ АРХИВИРОВАНИЯ ЖУРНАЛА БЕЗОПАСНОСТИ

1. `SDC_PLGN_AAAEVENTS_ALL_EVENTS_DEFAULT_DAYS`. Период удаления событий из БД. Значение по умолчанию — *365 дней*.

Чтобы изменить период удаления событий, в файле `/etc/sdc-env` задайте новое значение параметра

```
SDC_PLGN_AAAEVENTS_ALL_EVENTS_DEFAULT_DAYS=<new_count_days>
```

Перезагрузите **Sharx Base**.

Примечание

Изменения применяются только к новым событиям. События, имеющиеся в БД до внесения изменений, удалены не будут

2. `SDC_PLGN_AAAEVENTS_ARCHIVE_FOLDER`. Директория для хранения архивированных данных. Значение по умолчанию — `/var/lib/sharx/mziarch`.

Чтобы изменить директорию для хранения архивированных данных, в файле `/etc/sdc-env` задайте новое значение параметра

```
SDC_PLGN_AAAEVENTS_ARCHIVE_FOLDER=<new_folder>
```

Перезагрузите **Sharx Base**.

Архивирование происходит 1 раз в месяц. В архив попадают данные за позапрошлый месяц. Например, в марте архивируются события, произошедшие в январе

Примечание

Архивы, созданные до внесения изменений, автоматически не переносятся в новую директорию

3. `SDC_PLGN_AAAEVENTS_ARCHIVE_FILENAME_TEMPLATE`. Формат имени файла архива, созданного **автоматически**.

Значение по умолчанию — `ns_{ns}/aaaevents_{ns}{cluster}{yyyyymm}`

где

- `ns` — имя ВЦОД заархивированных данных;
- `cluster` — имя кластера, которому принадлежит ВЦОД;
- `yyyyymm` — год и месяц архивирования.

Чтобы изменить формат имени файла архива, созданного **автоматически**, в файле `/etc/sdc-env` задайте новое значение параметра

Sharx Base 6.3. Руководство администратора

```
SDC_PLGN_AAAEVENTS_ARCHIVE_FILENAME_TEMPLATE=<new_format>
```

Перезагрузите **Sharx Base**.

4. `SDC_PLGN_AAAEVENTS_ARCHIVE_MANUAL_FILENAME_TEMPLATE`. Формат имени файла архива, созданного вручную.

Значение по умолчанию — `ns_{ns}/aaaevents_manual_{ns}{cluster}{to}`

где

- `ns` — имя ВЦОД заархивированных данных;
- `cluster` — имя кластера, которому принадлежит ВЦОД;
- `to` — архивирование событий, созданных включительно до указанной даты.

Чтобы изменить формат имени файла архива, созданного вручную, в файле `/etc/sdc-env` задайте новое значение параметра

```
SDC_PLGN_AAAEVENTS_ARCHIVE_MANUAL_FILENAME_TEMPLATE=<new_format>
```

Перезагрузите **Sharx Base**.

5. `SDC_PLGN_AAAEVENTS_FREESPACE_LIMIT_FOLDER`. Раздел диска для контроля заполненности. По умолчанию не задано.

Чтобы задать параметр, в файле `/etc/sdc-env` введите

```
SDC_PLGN_AAAEVENTS_FREESPACE_LIMIT_FOLDER=<new_folder>
```

Перезагрузите **Sharx Base**.

6. `SDC_PLGN_AAAEVENTS_FREESPACE_LIMIT_BYTES`. Минимально допустимое значение свободного места на отслеживаемом разделе диска, байт. По умолчанию — *1000000000 байт (1 Гбайт)*.

Чтобы изменить параметр, в файле `/etc/sdc-env` задайте

```
SDC_PLGN_AAAEVENTS_FREESPACE_LIMIT_BYTES=<new_value>
```

Перезагрузите **Sharx Base**.

МАРШРУТ ОТПРАВКИ УВЕДОМЛЕНИЙ О ЗАНЯТОМ МЕСТЕ В РАЗДЕЛЕ ХРАНЕНИЯ АРХИВОВ

Укажите маршрут отправки уведомлений о занятом месте в директории с архивами.

В **Sharx Base** в командой строке введите

```
aaaevents archive freespacelimitnotifroute --setroute <route>
```

где `setroute` — имя маршрута для отправки уведомлений о событии.

21.3.2. Принудительно архивировать и очистить журнал безопасности

Примечание

При архивировании все события за указанный период переносятся из журнала безопасности в архив, журнал безопасности при этом очищается

Чтобы принудительно архивировать события, введите команду

```
aaaevents archive make --up_to_date <date>
```

где `up_to_date` — переменная для установки периода архивирования.

Возможные значения `up_to_date`:

- при установке значения `1` все события из журнала безопасности **до текущего момента** будут заархивированы и удалены из журнала безопасности;
- при установке конкретной даты в формате `YYYY-MM-DD` будут собраны все события **до этой даты включительно**

Пример

```
aaaevents archive make --up_to_date 2025-03-09
```

В архив будут собраны все события с кластера до 9 марта 2025 года включительно

- можно указать отрицательное число, например `-10`. В архив будут собраны все события, которые произошли **до даты десятидневной давности**

Пример

```
aaaevents archive make --up_to_date -10
```

20 января 2025 г во флаге указали `-10`. В архив попали все события до 10 января 2025 г. включительно

22. Логирование

Примечание

Статья актуальна для подключения к rsyslog-серверу

В **Sharx Base** возможно настроить логирование событий в рамках ВЦОД на rsyslog-сервер.

Sharx Base 6.3. Руководство администратора

rsyslog-сервер – это централизованная система сбора, хранения и анализа логов с различных сетевых устройств. Он позволяет собирать информацию о событиях с множества источников в одном месте для последующего мониторинга и анализа.

Основные функции:

- централизованный сбор логов;
- мониторинг событий в реальном времени;
- отслеживание ошибок в работе системы;
- анализ производительности сетевых устройств;
- обеспечение безопасности путем сбора информации о подозрительных действиях.

22.1. Создать конфигурацию подключения к rsyslog-серверу

✘ Внимание

Во ВЦОД может существовать только одна конфигурация конечной точки rsyslog-сервера

Конфигурация позволяет настроить параметры подключения для отправки событий и логов на удаленный сервер.

Чтобы создать новую конфигурацию подключения к rsyslog-серверу, введите

```
aaaevents syslog add --protocol <PROTOCOL>
                    --address <ADDRESS>
                    [--port <PORT>]
                    [--use_tls <USE_TLS>]
                    [--ca_data <CA_DATA>]
                    [--ca_file <CA_FILE>]
                    [--log_enabled <LOG_ENABLED>]
```

где

- `protocol` – протокол, который будет использоваться для записи в syslog. Возможные значения `UDP` или `TCP`;
- `address` – адрес syslog-сервера, куда будут отправляться логи. Укажите IP-адрес или доменное имя сервера;
- `port` – порт для подключения к syslog-серверу. Если порт не указан, по умолчанию присваивается `514` для UDP и `1468` для TCP;

Sharx Base 6.3. Руководство администратора

- `use_tls` – параметр включения защищенного соединения с syslog-сервером через TLS. Подключение по TLS доступно только для протокола TCP. Для подключения с TLS необходимо предоставить сертификат сервера. Пример подключения с TLS описан [ниже](#);
- `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`;
- `log_enabled` – включение или отключение логирования на syslog-сервер. Возможные значения `enable` или `disable`.

Пример

Пример настройки конечной точки с подключением по UDP или TCP

```
aaaevents syslog add --protocol <udp | tcp> --address <rsyslog-server-addr>
```

22.2. Управлять конфигурацией

1. Просмотреть параметры конфигурации syslog-подключения

```
aaaevents syslog show
```

2. Обновить конфигурацию

```
aaaevents syslog update [--protocol <PROTOCOL>]
                        [--address <ADDRESS>]
                        [--port <PORT>]
                        [--use_tls <USE_TLS>]
                        [--ca_data <CA_DATA>]
                        [--ca_file <CA_FILE>]
                        [--log_enabled <LOG_ENABLED>]
```

где

- `protocol` – протокол, который будет использоваться для записи в syslog. Возможные значения `UDP` или `TCP`;
- `address` – адрес syslog-сервера, куда будут отправляться логи. Укажите IP-адрес или доменное имя сервера;
- `port` – порт для подключения к syslog-серверу. Если порт не указан, по умолчанию присваивается `514` для `UDP` и `1468` для `TCP`;

Sharx Base 6.3. Руководство администратора

- `use_tls` – параметр включения защищенного соединения с syslog-сервером через TLS. Подключение по TLS доступно только для протокола TCP. Для подключения с TLS необходимо предоставить сертификат 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`;
- `log_enabled` – включение или отключение логирования на syslog-сервер. Возможные значения `enable` или `disable`.

3. Удалить конфигурацию

```
aaaevents syslog del
```

22.3. Пример настройки конечной точки подключения по TCP с TLS

1. Обратитесь в техническую поддержку Sharx Base для подготовки сервера к записи событий. Когда предварительная настройка будет закончена, техническая поддержка вышлет соответствующее сообщение и сгенерированный сертификат `<cert_name.crt>` для дальнейшей настройки.
2. Поместите файл `<cert_name.crt>`, предоставленный поставщиком, в хранилище доверенных сертификатов

ОС Debian

Директория `/usr/local/share/ca-certificates/`

ОС AlmaLinux

Директория `/etc/pki/ca-trust/source/anchors/`

3. Обновите сертификаты

ОС Debian

Выполните команду

```
update-ca-certificates
```

ОС AlmaLinux

Выполните команду

```
1 update-ca-trust
2 openssl x509 -in /etc/pki/ca-trust/extracted/openssl/ca-bundle.trust.crt -text | less
```

4. Настройте конечную точку

```
aaaevents syslog add --protocol tcp --address <rsyslog-server-addr> --use_tls yes --ca_file <full-path-to-certificate>
```

где `<full-path-to-certificate>` – путь к файлу сертификата.

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.

23.2.1. Экспортеры

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 командах группы `metrics ...`.

Имена сервисов на узлах отличаются от имен экспортеров в Sharx Base.

Таблица – Соответствие имен экспортеров на узлах и в Sharx Base.

Sharx Base 6.3. Руководство администратора

Имя сервиса на узле	Имя экспортера в Sharx Base	Порт	Назначение
sdс-cgroup_exporter	cgroup_exporter	9198	Метрики ограничений и использования ресурсов системными группами
sdс-disk_exporter	disks_exporter_sharx	9196	Метрики дисков
sdс-domain-exporter	domain_exporter	9190	Метрики состояния виртуальных машин
sdс-ipmi_exporter	ipmi_exporter_sharx	9195	Метрики состояния аппаратного обеспечения
sdс-node_exporter	node_exporter_sharx	9191	Метрики общего состояния узлов кластера
sdс-sp_exporter	sp_exporter_sharx	9197	Метрики хранилища РСХД

Примечание

В командах группы `metrics ...` используйте имена из столбца **Имя экспортера в Sharx Base**

Предустановки для мониторинга выполняются сотрудниками технической поддержки.

Доступные действия **Администратора кластера** перечислены ниже.

23.2.2. Настройка мониторинга

1. Установить конфигурацию экспортера для сбора метрик

```
metrics exporter add --name <NAME>
                    --port <PORT>
                    [--metrics_path <METRICS_PATH>]
```

Sharx Base 6.3. Руководство администратора

```
[--nodes <NODES ...>]
[--retention_period <RETENTION_PERIOD>]
[--filters <FILTERS>]
[--cron <CRON>]
```

где

- `name` — имя экспортера. Возможные значения:
 - `domain_exporter` — метрики состояния виртуальных машин;
 - `node_exporter_sharx` — метрики общего состояния узлов кластера;
 - `ipmi_exporter_sharx` — метрики состояния аппаратного обеспечения;
 - `sp_exporter_sharx` — метрики хранилища РСХД;
 - `disks_exporter_sharx` — метрики дисков;
 - `cgroup_exporter` — метрики ограничений и использования ресурсов системными группами.
- `port` — HTTP-порт экспортера;
- `metrics_path` — путь для получения метрик. По умолчанию — `/metrics`;
- `nodes` — узлы кластера для мониторинга. Укажите идентификатор одного или нескольких узлов. При значении `*` мониторинг будет настроен для всех узлов;
- `retention_period` — период мониторинга в секундах. По умолчанию — **600 с**;
- `filters` — фильтры по имени метрики и меткам;
- `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.



Установить экспортеры с параметрами по умолчанию

```

metrics exporter add --name domain_exporter
    --port 9190
    --cron '*/1 * * * *'
    --nodes * --retention_period 1800
    --filters ' [{"name": "libvirt_domain_info_vmstate"},
{"name": "libvirt_domain_vcpu_time_seconds_total"},
{"name": "libvirt_domain_info_maximum_memory_bytes"},
{"name": "libvirt_domain_memory_stats_used_percent"},
{"name": "libvirt_domain_interface_stats_receive_bytes_total"},
{"name": "libvirt_domain_interface_stats_transmit_bytes_total"} ]'

metrics exporter add --name node_exporter_sharx
    --port 9191 --retention_period 43200
    --nodes * --metrics_path /metrics
    --cron '*/5 * * * *'
    --filters ' [{"name": "node_cpu_seconds_total", "labels": {"mode": "idle"}},
{"name": "node_memory_MemFree_bytes", "labels": {}},
{"name": "node_memory_Buffers_bytes", "labels": {}},
{"name": "node_memory_Cached_bytes", "labels": {}},
{"name": "node_memory_MemAvailable_bytes", "labels": {}},
{"name": "node_network_receive_bytes_total", "labels": {"device": "^eno[0-9][0-9]?$"}},
{"name": "node_network_transmit_bytes_total", "labels": {"device": "^eno[0-9][0-9]?$"}},
{"name": "node_network_receive_errs_total", "labels": {"device": "^eno[0-9][0-9]?$"}},
{"name": "node_network_transmit_errs_total", "labels": {"device": "^eno[0-9][0-9]?$"} } ]'

metrics exporter add --name ipmi_exporter_sharx
    --port 9194 --retention_period 43200
    --nodes * --metrics_path /metrics
    --cron '*/5 * * * *'
    --filters ' [{"name": "^ipmi_metrics$", "labels": {"verb": "^P[12]
Temperature_celsius$"}}, {"name": "^ipmi_metrics$", "labels":
{"state": "ps_current_consumption"} } ]'

metrics exporter add --name sp_exporter_sharx
    --port 9192 --retention_period 600
    --nodes * --metrics_path /metrics
    --cron '*/5 * * * *'

metrics exporter add --name disks_exporter_sharx
    --port 9193 --retention_period 600
    --nodes * --metrics_path /metrics
    --cron '*/5 * * * *'

metrics exporter add --name cgroup_exporter
    --port 9198
    --nodes *

```

2. Просмотреть список подключенных экспортеров

```
metrics exporter list
```

3. Просмотреть информацию об определенном экспортере

```
metrics exporter show --name <NAME>
```

Sharx Base 6.3. Руководство администратора

где `name` — имя экспортера.

4. Обновить конфигурацию экспортера

```
metrics exporter update --name <NAME>
                        [--port <PORT>]
                        [--metrics_path <METRICS_PATH>]
                        [--nodes <NODES ...>]
                        [--retention_period <RETENTION_PERIOD>]
                        [--filters <FILTERS>]
                        [--cron <CRON>]
```

5. Получить метрики

```
metrics data fetch --name <NAME>
                  --sc_node <SC_NODE>
                  [--filters <FILTERS>]
                  [--current <yes|no>]
```

где

- `name` — имя экспортера;
- `sc_node` — идентификатор узла для мониторинга;
- `filters` — фильтры по имени метрики и меткам;
- `current` — флаг выбора определенной метрики. Возможные значения:
 - YES — выбор определенной метрики,
 - NO — выбор всех доступных метрик;

Внимание

Для параметра `sc_node` нельзя использовать значение *

Запрос метрик ipmi_exporter_sharx

Запрос

```
metrics data fetch --sc_node e87fc5bf-9341-4a3c-ac9d-c12913d34d1f
                  --name ipmi_exporter_sharx
                  --filters '[
                        {"name": "^ipmi_device_status$"},
                        {"name": "^ipmi_sensor$", "labels": {"name": "AP[12]
Temperature_celsius$"}},
                        {"name": "^ipmi_sensor$", "labels": {"name": "APS[12] Input Power$"}}
                    ]'
```

Пример ответа см. в приложенном файле [example_ipmi_response.json](#)

Запрос метрик `sp_exporter_sharx`

Запрос

```
metrics data fetch --sc_node e87fc5bf-9341-4a3c-ac9d-c12913d34d1f
                    --name sp_exporter_sharx
```

Пример ответа см. в приложенном файле [example_sp_response.json](#)

6. Удалить конфигурацию экспортера

```
metrics exporter del --name <NAME>
```

где `name` — имя экспортера.

23.2.3. Проверка работоспособности

После настройки внутреннего мониторинга проверьте, что конфигурации экспортеров добавлены, и метрики доступны для получения.

1. Просмотрите список подключенных экспортеров

```
metrics exporter list
```

Убедитесь, что в списке отображаются [все экспортеры](#).

2. Просмотрите конфигурацию нужного экспортера

```
metrics exporter show --name <NAME>
```

В ответе проверьте:

- имя экспортера;
- HTTP-порт экспортера;
- узлы, для которых настроен сбор метрик;
- период хранения метрик;
- путь получения метрик;
- расписание запуска;
- фильтры, если они заданы.

3. Получите тестовую метрику

```
metrics data fetch --name <NAME>
                  --sc_node <SC_NODE>
                  [--filters <FILTERS>]
                  [--current <YES|NO>]
```

Sharx Base 6.3. Руководство администратора

4. Проверьте ответ команды. В ответе должны отображаться:

- имя метрики;
- значение метрики;
- метки метрики.

Пример ответа

```
{"metric_name": "libvirt_up", "metric_value": [1.0], "labels": "{}"}
```

5. Если команда возвращает значение метрики, внутренний мониторинг работает корректно для выбранного экспортера и узла.

6. Если метрика не возвращается, проверьте:

- добавлена ли конфигурация экспортера;
- корректно ли указано имя экспортера;
- доступен ли узел, указанный в параметре `sc_node`;
- корректно ли указан HTTP-порт экспортера;
- корректно ли заданы фильтры;
- запущен ли соответствующий экспортер на узле.

Если проблемы с мониторингом сохраняются, обратитесь в [техническую поддержку 000 «Шаркс ДЦ»](#).

23.3. Работа с шаблонами

Шаблоны используются во внутреннем мониторинге. Они позволяют извлекать информацию из экспортера и фильтровать ее согласно заданным требованиям.

Пример

Шаблон файла отображения метрик для экспортеров **sdс-node_exporter**, **sdс-ipmi_exporter**, **sdс-disk_exporter** – [metrics_display_template.yaml](#).

Для экспортера **sdс-sp_exporter** – шаблон [metrics_display_template_sp.yaml](#)

1. Добавить шаблон

```
metrics template add --name <NAME>
                        [--data <DATA>]
```

где

Sharx Base 6.3. Руководство администратора

- `name` — имя шаблона;
- `data` — данные шаблона.

2. Просмотреть список созданных шаблонов

```
metrics template list
```

3. Просмотреть подробную информацию о конкретном шаблоне

```
metrics template show --name <NAME>
```

где `name` — имя шаблона.

4. Обновить информацию в шаблоне

```
metrics template update --name <NAME>  
                        [--data <DATA>]
```

5. Отобразить метрики по фильтрам, указанным в шаблоне

```
metrics template execute --name <NAME>
```

где `name` — имя шаблона.

После выполнения команды проверьте, что в ответе отображаются метрики, соответствующие фильтрам шаблона.

Шаблон работает корректно, если команда возвращает отфильтрованный набор метрик.

Если команда не возвращает данные, проверьте:

- существует ли шаблон с указанным именем;
- корректно ли указаны данные шаблона;
- доступны ли экспортеры, из которых шаблон получает метрики;
- возвращают ли экспортеры метрики без применения шаблона;
- соответствуют ли фильтры шаблона фактическим именам метрик и меткам.

6. Удалить шаблон

```
metrics template del --name <NAME>
```

23.4. Внешний мониторинг. Sharx ProView

Sharx ProView — это дополнительное решение для сбора метрик платформы виртуализации Sharx Base, которое позволяет:

- хранить метрики неограниченное время;

Sharx Base 6.3. Руководство администратора

- 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 длительного хранения и расширенной визуализации

24. Сервисный режим узлов

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.

24.1. 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.

24.2. Отключить сервисный режим

```
scheduler drain del --nodes <NODES>
```

Sharx Base 6.3. Руководство администратора

После выполнения команда метка `system_drain_node` удаляется, и узлы возвращаются в общее планирование.

25. Обновить компоненты кластера

25.1. Автоматически обновить плагины

1. Чтобы настроить автообновление компонентов, создайте правило в **Sharx Base**.

В командной строке введите

```
updater param add --cron <cron> --enable <yes_or_no> --mods <package_list> --profile <dev_or_prod>
```

где

- `cron` — периодичность выполнения автообновления. Период автообновления 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` — автообновление выполняется каждое воскресенье,

`* 2 * * 7` — каждые два часа по воскресеньям

- `enable` — включение системы автообновления.

Возможные значения:

- `no` — система выключена,
- `yes` — система включена. Проверка обновлений будет происходить в указанный период времени (`cron`). При обнаружении обновлений произойдет автоматическая установка;
- `mods` — список плагинов и компонентов для обновления.

Возможные значения: `sdс-core`, `sdс-gateway-api`, `sdс-schema`, `sdс-pyenv3`, `sdс-plgn-faucet`, `sdс-plgn-ipam`, `sdс-plgn-hardware`, `sdс-plgn-logdump`, `sdс-plgn-updater`, `sdс-plgn-aaa`, `sdс-plgn-scheduler`, `sdс-plgn-services`, `sdс-plgn-storage`, `sdс-plgn-domain`, `sdс-plgn-goss`, `sdс-plgn-signer`, `sdс-plgn-aaaevents`, `sdс-plgn-notif`;

Sharx Base 6.3. Руководство администратора

- `profile` – профиль инициализации пространства ключей Cassandra.

2. Просмотр текущих настроек автообновления

```
updater param show
```

3. Изменить параметры

```
updater param update --cron <cron> --enable <yes_or_no> --mods <package_list> --profile <dev_or_prod>
```

4. Удалить все параметры

```
updater param del
```

25.2. Обновить плагины вручную

Ручное обновление компонентов используется при необходимости **внепланового обновления** командой

```
updater update add --mods <package_list> --node <node_uuid>
```

где

- `mods` – список компонентов для обновления;
- `node` – идентификатор узла в кластере.

Примечание

При отсутствии параметра `node` указанные пакеты `mods` обновятся на всех узлах кластера.
При отсутствии параметра `mods` обновятся все имеющиеся пакеты

25.3. Проверить обновления

1. Чтобы просмотреть список выполненных обновлений, введите

```
updater update list
```

2. Просмотр конкретного обновления

```
updater update show --uuid <update_uuid>
```

3. Удалить определенную запись об обновлении

```
updater update clear --uuid <update_uuid>
```

где `uuid` – идентификатор записи. При значении `update_uuid`, равном `*`, происходит удаление всех записей об обновлении.

26. Система автоматизации процедур checker

26.1. Checker

Плагин checker – система автоматизации рутинных операций и комплексных задач в кластере и ВЦОД. Он позволяет:

- Определять процедуры – последовательности шагов для выполнения типовых задач.
- Делегировать процедуры в конкретные ВЦОД.
- Запускать процедуры вручную, по расписанию или в ответ на события.
- Управлять выполнением процедур в рамках одного или нескольких ВЦОД.
- Обрабатывать ошибки, логировать выполнение и управлять зависимостями между задачами.
- Отслеживать статус выполнения в реальном времени.

26.1.1. Ключевые понятия

Процедура – YAML-файл, описывающий последовательность шагов, условия, переменные и обработку ошибок.

Шаг – отдельный этап процедуры, содержащий условия и действия.

Делегирование – назначение процедуры для использования в определенных ВЦОД.

Планирование – настройка автоматического выполнения процедуры по расписанию.

26.1.2. Как работает Checker?

1. Определение процедуры. Администратор кластера создает YAML-описание процедуры.
2. Делегирование. Процедура становится доступной в выбранных ВЦОД.
3. Планирование или запуск. Администратор ВЦОД запускает процедуру вручную или по расписанию.
4. Выполнение. Checker выполняет шаги, проверяет условия, обрабатывает ошибки.
5. Завершение. Процедура завершается с одним из статусов.

26.1.3. Статьи по работе с системой Checker

1. [Управление процедурами через CLI](#). Базовые команды.
2. [YAML-структура процедур](#). Создание сложных процедур.
3. [Примеры процедур](#). Готовые сценарии для типовых задач.

26.2. Управление процедурами

Ключевые понятия описаны в статье [Checker](#).

В статье описано управление процедурами:

- [Добавление, обновление и удаление](#) процедуры.
- [Делегирование](#) процедуры во ВЦОД.
- [Планирование](#) выполнения по расписанию.
- [Запуск](#) процедуры вручную.
- [Отслеживание](#) статуса выполнения.

26.2.1. Определить процедуру

Примечание

Действия выполняются пользователем с ролью **Администратор кластера**

Чтобы определить процедуру, выполните следующие шаги:

1. Проверьте название и описание процедуры на корректность без добавления процедуры в систему

```
checker procedure add --name <NAME>
                    --definition <DEFINITION>
                    [--json <no|yes>]
                    --validate "yes"
```

где

- `name` — имя процедуры. Должно быть уникально в пределах кластера;
- `definition` — определение процедуры. Передается как строка в формате YAML. Подробнее см. [YAML-структура процедур](#) и [Примеры процедур](#);
- `json` — формат возвращения ошибки. Возможные значения:
 - `"no"` — ошибка возвращается в человекочитаемом формате. Значение по умолчанию;
 - `"yes"` — ошибка возвращается в формате JSON;

Sharx Base 6.3. Руководство администратора

- `validate` – проверка названия и описания процедуры на корректность без добавления процедуры в систему. В данной команде проверка должна быть включена `"yes"`.

Если при проверке получены ошибки, исправьте их.

Если проверка прошла успешно, приступайте к следующему шагу.

2. Чтобы определить процедуру, введите

```
checker procedure add --name <NAME>
                    --definition <DEFINITION>
```

где

- `name` – имя процедуры;
- `definition` – определение процедуры. Передается как строка в формате YAML.

26.2.2. Обновить процедуру

1. Чтобы обновить процедуру, сначала проверьте корректность нового описания без обновления в системе

```
checker procedure update --name <NAME>
                       --definition <DEFINITION>
                       [--json <no|yes>]
                       --validate "yes"
```

где

- `name` – имя обновляемой процедуры;
- `definition` – новое определение процедуры;
- `json` – формат возвращения ошибки. Возможные значения:
 - `"no"` – ошибка возвращается в человекочитаемом формате. Значение по умолчанию;
 - `"yes"` – ошибка возвращается в формате JSON;
- `validate` – проверка названия и описания процедуры на корректность без добавления процедуры в систему. В данной команде проверка должна быть включена `"yes"`.

Если при проверке получены ошибки, исправьте их.

Если проверка прошла успешно, приступайте к следующему шагу.

2. Обновите процедуру в системе

```
checker procedure update --name <NAME>
                       --definition <DEFINITION>
```

где

Sharx Base 6.3. Руководство администратора

- `name` — имя обновляемой процедуры;
- `definition` — новое проверенное определение процедуры.

26.2.3. 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.

26.2.4. Удалить процедуру

Чтобы удалить процедуру, введите

```
checker procedure del --name <NAME>
                     [--json <no|yes>]
```

где `name` — имя удаляемой процедуры.

26.2.5. Делегировать процедуру

После того, как процедура определена, необходимо ее делегировать, то есть сделать доступной для целевых ВЦОД

1. Чтобы делегировать процедуру во ВЦОД, введите команду

```
checker procedure delegate --name <NAME>
                          --ns <NS>
```

где

- `name` — имя процедуры;
- `ns` — имя ВЦОД. При значении `*` процедура будет делегирована во все ВЦОД.

Sharx Base 6.3. Руководство администратора

После делегирования процедура станет доступна в указанных ВЦОД для запуска вручную или по расписанию.

2. Чтобы убрать делегирование процедуры из ВЦОД, используйте следующую команду

```
checker procedure undelegate --name <NAME>
                             --ns <NS>
```

При значении `--ns *` процедура будет убрана из всех ВЦОД.

26.2.6. Планирование выполнения процедуры

Планирование процедур по расписанию доступно как для кластера во ВЦОД управления, так и для отдельных ВЦОД. Администратор кластера может планировать процедуры, предназначенные для выполнения на уровне кластера, например, фенсинг узлов. Для процедур, которые должны выполняться во ВЦОД, администратор кластера предварительно делегирует их, после чего администратор соответствующего ВЦОД может настроить для них расписание.

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

26.2.7. 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;

Sharx Base 6.3. Руководство администратора

- `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.

26.2.8. 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.

26.2.9. 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:

Sharx Base 6.3. Руководство администратора

- 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",
  <...>
}
```

26.2.10. Практические советы

ТЕСТИРОВАНИЕ НОВОЙ ПРОЦЕДУРЫ

1. Сначала проверьте имя и определение новой процедуры

```
checker procedure add --name "procedure_name" --definition "<YAML>" --validate "yes" --
json "no|yes"
```

2. Добавьте процедуру в систему

```
checker procedure add --name "procedure_name" --definition "<YAML>"
```

3. Делегируйте процедуру во ВЦОД

```
checker procedure delegate --name "procedure_name" --ns "ns_name"
```

4. Запустите процедуру вручную для проверки выполнения

```
checker procedure execute --name "procedure_name" --vars '{}' --json "no|yes"
```

5. Получите UUID задачи из ответа.
6. Отслеживайте выполнение

Sharx Base 6.3. Руководство администратора

```
checker procedure status show --uuid <UUID>
```

7. 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 нужный ВЦОД.
- Проверьте корректность cron-выражения.

26.2.11. Дополнительная информация

1. Статья с описанием [YAML-структуры процедур](#).
2. [Примеры процедур](#) с готовыми сценариями для типовых задач.

26.3. YAML-структура процедур

Статья описывает принципы YAML-описания процедур, чтобы выполнять следующие действия:

- Создавать новые процедуры.
- Использовать переменные, условия и действия.
- Обработать ошибки и настраивать логирование.
- Оптимизировать выполнение с помощью встроенных функций.

Общая структура YAML-файла

```
vars:  
##### Определение переменных  
steps:  
##### Последовательность шагов  
error-handlers:  
##### Обработчики ошибок  
config:  
##### Настройки процедуры
```

Переменные vars

Переменные объявляются в корневом блоке верхнего уровня `vars`.

```
# Базовое определение переменной
```

Sharx Base 6.3. Руководство администратора

```
vars:
  name_var:
    type: "тип"
    default: "значение_по_умолчанию" # необязательно
    visibility: "public|private" # видимость переменной. По умолчанию private
```

Тип данных

Возможные значения типов данных:

- `string` — текстовая строка;
- `number` — целое или дробное число;
- `boolean` — `true` или `false`;
- `array` — массив `["a", "b", "c"]`;
- `object` — набор пар `{"key": "value"}`;
- `null` — пустое значение.

Видимость переменных

Возможные значения:

- `private` — доступны только внутри процедуры. Значение по умолчанию;
- `public` — можно задать снаружи через команды [запуска по расписанию](#) `checker procedure schedule` или [запуска вручную](#) `checker procedure execute`.

Пример

```
vars:
  internal_var:
    type: "string"
    default: "secret"
    visibility: "private" # нельзя изменить при запуске

  user_param:
    type: "number"
    visibility: "public" # обязательно указать при запуске
```

Подстановка переменных

Используйте `{{ имя }}` для подстановки значений

```
vars:
  node_name:
    type: "string"
    default: "node-01"

steps:
  step1:
```

Sharx Base 6.3. Руководство администратора

```
action:  
  apicall: "cluster nodes show --node {{ node_name }}"
```

Расширенная подстановка с JMESPath

JMESPath позволяет выполнять сложные операции с данными в шаблонах переменных.

Базовый синтаксис

```
"{{var_name | jmespath_expression}}"
```

Основные примеры:

1. Фильтрация массива по условию

```
# Получить имена только узлов в статусе online  
"{{nodes | [?status=='online'].name}}"  
# Результат: ["node-01", "node-03", "node-05"]
```

2. Доступ к вложенным данным

```
# Извлечь значение из сложного объекта  
"{{vm_info | metrics.cpu_usage}}"  
# Результат: 45.7 (процент использования CPU)
```

3. Агрегация данных

```
# Подсчитать количество элементов  
"{{vms | length(@)}}"  
# Результат: 12 (количество VM)  
  
# Суммировать значения  
"{{vms | sum_by([].cpu_cores)}}"  
# Результат: 48 (общее количество CPU ядер)
```

4. Агрегация данных

```
# Выбрать VM с CPU > 2 и получить их имена  
"{{vms | [?cpu_cores > `2`].name}}"  
# Результат: ["vm-db", "vm-app"]
```

Примечание

Для реализации более сложных процедур введены [Расширения JMESPath](#)

Распаковка массивов

```
vars:  
  nodes:  
    type: "array[string]"
```

Sharx Base 6.3. Руководство администратора

```
default: ["node1", "node2", "node3"]
```

```
##### В шаблоне
"Проверка узлов: {{*nodes}}"
##### Результат:
##### "Проверка узла: node1"
##### "Проверка узла: node2"
##### "Проверка узла: node3"
```

Совместная распаковка массивов

✘ Внимание

Все массивы должны иметь одинаковую длину, иначе процедура завершится ошибкой

```
vars:
  nodes: ["a", "b"]
  ips: ["1.1.1.1", "2.2.2.2"]

##### В шаблоне
"{{*nodes}} имеет IP {{*ips}}"
##### Результат:
##### "a имеет IP 1.1.1.1"
##### "b имеет IP 2.2.2.2"
```

Встроенные переменные

Кроме переменных, определяемых в процедуре, есть встроенные переменные, позволяющие взаимодействовать с контекстом исполнения процедуры. Встроенные переменные отличаются от обычных префиксом `@` перед названием.

`@deps`

`@deps` позволяет добавлять зависимые подзадачи к текущей процедуре в конце шага. Все подзадачи должны завершиться перед тем, как процедура сможет перейти к следующему шагу. Эта переменная очищается в начале каждого шага.

```
@deps:
  type: "array[object]"
  default: []
```

Объекты, которые можно добавлять в переменную `@deps`, должны иметь формат `TaskId`.

На данный момент в таком формате выводят результат следующие команды:

- `checker procedure execute --json "yes" <...>`.
- `scheduler request reschedule`.

Sharx Base 6.3. Руководство администратора

@deps_status

@deps_status содержит результаты выполнения всех зависимых подзадач, добавленных в @deps.

```
@deps_status:  
  type: "array[object]"  
  default: []
```

Объекты переменной @deps_status имеют следующую структуру:

```
{  
  "step": "<название шага, на котором остановилась задача>",  
  "task_status": "<статус задачи, один из: SUCCESS, TIMEOUT, ERROR>",  
  "run_error": "<сообщение ошибки выполнения задачи>" | null,  
  "task_steps_total": "<общее число шагов задачи>",  
  "run_on_vip": "<выполняется ли задача только на vip-узле>",  
  "run_on_node": "<uuid узла, на котором задача должна выполняться>" | null,  
  "task_uuid": "<uuid задачи>",  
  "task_name": "<название задачи>" | null,  
  "task_message": "<лог сообщений задачи>" | null,  
  "worker_node": "<узел, на котором выполнялась задача>",  
  "created_at": "<дата и время создания задачи>",  
  "updated_at": "<дата и время последнего обновления задачи>"  
  // ...другие поля, значение которых на данный момент не задокументировано  
}
```

@result

@result содержит результат выполнения действия, очищается в начале каждого шага. Используется для логирования результата выполнения действия без сохранения в отдельную переменную.

```
@result:  
  type: "object"
```

Шаги

Процедура разделена на последовательность шагов. Каждый шаг определяется как совокупность условий `condition` и действия `action`. Перед исполнением процедуры запускается проверка всех условий. Пока все условия не будут соблюдены, процедура не может перейти к исполнению действия. После исполнения действия, процедура переходит на следующий шаг.

Основная часть условий и действий – это команда. В качестве команды можно указать одно из двух значений:

Sharx Base 6.3. Руководство администратора

- `apicall`: делает запрос с синтаксисом `sdc-cli` и передает результат исполнения команды на обработку в `expect/parse`
- `const`: передает указанное значение напрямую в `expect/parse`. В качестве значения команды может быть указан либо один шаблон, результат исполнения которого будет передан дальше на обработку, либо список шаблонов. В случае списка шаблонов на обработку будет передан массив с результатами каждого из перечисленных шаблонов.

Пример

```
# Команда
apicall: "plugin command"
# Результат
<plugin command result>

# Команда
const: ["123", "abc"]
# Результат
["123", "abc"]

# Команда
apicall:
- "plugin1 command"
- "plugin2 command"
# Результат
[
<plugin1 command result>,
<plugin2 command result>
]
```

Структура шага

```
steps:
  name_step:
    conditions: # условия для выполнения
      - ...
    action:     # действие, которое выполнится
      ...
    on-error:  # обработчики ошибок для шага
      - ...
    attempts: # настройки повторных попыток
      ...
```

Условия `conditions`

Перед тем, как перейти к исполнению действия шага `action`, нужно, чтобы были выполнены все условия `conditions`.

`conditions` — это список команд, которые будут запускаться каждый раз при исполнении шага. Все условия шага должны завершиться без ошибки и все выражения в `expect` должны быть `true`, после чего шаг перейдет к исполнению действия.

Sharx Base 6.3. Руководство администратора

Для обработки результата выполнения команды условия предусмотрено поле `expect`, которое является списком выражений. Каждое выражение имеет следующую структуру:

```
conditions:
- apicall: "cluster nodes list" # команда
  expect: # проверки результата
  - filter: "[?status=='online'] | length(@)" # JMESPath
    op: ">=" # операция сравнения
    value: 3 # ожидаемое значение
```

где

- `filter` — выражение `jmespath`, обрабатывающее результат исполнения команды. Результат этого выражения передается в `op`.
- `op` — операция сравнения: `==`, `!=`, `>`, `>=`, `<`, `<=`. Левым операндом будет результат исполнения `filter`, а правым `value`. Если слева от операции поставить `*` (`*==`, `*<=`, и т.д.), то операция повторится для каждого элемента левого операнда.
- `value` — значение, с которым `op` будет сравнивать результат исполнения `filter`.

Если хотя бы одно выражение вернуло `false`, то условие считается невыполненным, и шаг повторяется. Пока все условия не выполнены и ни одно из выражений не завершилось с ошибкой, процедура будет проверять текущий шаг. Это поведение можно изменить с помощью `attempts`.

Пример

```
vars:
  nodes_to_test:
    type: "array[string]"
    default: ['node1', 'node2']
steps:
  step1:
    conditions:
      - apicall: "cluster nodes list" # [{"status": "online", ...}, ...]
        expect:
          # Вариант 1. Все узлы, кроме nodes_to_test имеют статус `online`
          - filter: "[?!contains({{node_to_test}}, @.name)].status"
            op: "*=="
            value: "online"
          # Вариант 2. Все узлы, кроме nodes_to_test имеют статус `online`
          - filter: "[?status=='online'].name"
            op: "*!="
            value: "{{*nodes_to_test}}"
      - apicall: "cluster nodes show --node {{*nodes_to_test}}" # [{"node": "node1", ...}, {"node": "node2", ...}]
        expect:
          # Узлы из `nodes_to_test` имеют статус `online`
          - filter: "[].status"
            op: "*=="
            value: "online"
```

Sharx Base 6.3. Руководство администратора

Действия `action`

После того как все условия `conditions` были выполнены, шаг переходит к действию `action`. Действие — это команда, выполняющая определенную операцию и, возможно, обновляющая одну или несколько переменных.

В отличие от `expect` у условия, для работы с переменными используется поле `parse`. Формат выражений у `parse` отличается тем, что вместо проверки результата происходит обработка и сохранение в переменную.

Основные операции, поддерживаемые выражениями в `parse`:

- `store` — записать значение фильтра в переменную;
- `push` — добавить значение в конец массива. Используется только для переменных типа `array`;
- `extend` — добавить в конец массива все значения массива, полученного в результате исполнения команды. Используется только для переменных типа `array` и результата типа `array`.

Пример

```
vars:
  value_var:
    type: "string"
  array_var:
    type: "array[string]"
    default: []
steps:
  step1:
    action:
      apicall: "<...>"
      parse:
        - filter: "<...>"
          op: "extend"
          var: "array_var"
        - filter: "<...>"
          op: "push"
          var: "array_var"
        - filter: "<...>"
          op: "store"
          var: "value_var"
```

Попытки `attempts`

Попытки `attempts` ограничивают количество повторных выполнений команды. При необходимости в поле `attempts` указывается максимальное или минимальное количество попыток перед тем, как команда завершится с ошибкой. Между попытками можно определить минимальный интервал, с которым будут считаться попытки.

Пример

```
steps:
  # Эта команда будет выполнена до трех раз с интервалом в 60 секунд между каждой попыткой при
  # ошибке в действии. Если после третьего исполнения действие все еще завершается с ошибкой, то
  # весь шаг также завершится с последней полученной ошибкой.
  step1:
    action:
      apicall: "<...>"
    attempts:
      max: 3
      interval: 60

  # Это условие будет запущено как минимум 2 раза с интервалом 60 секунд между каждой попыткой.
  # Если во время одной из двух попыток будет ошибка или `expect` вернет false, то условие
  # завершится с ошибкой.
  step2:
    conditions:
      - apicall: "<...>"
      expect:
        - <...>
    attempts:
      min: 2
      interval: 60
```

Обработка ошибок

Во время выполнения шага может произойти ошибка: было достигнуто ограничение на количество попыток, во время парсинга вывода произошла ошибка, плагин вернул ошибку, и т.д. Для обработки ошибок предусмотрено общее для всех шагов поле `error-handlers`. Данное поле содержит команды, которые могут вызываться при ошибках, а также команды, вызываемые по умолчанию для определенных типов ошибок.

```
vars:
  <...>
steps:
  <...>
error-handlers:
  <...>
```

Для того, чтобы во время ошибки шага был выбран нужный обработчик ошибок, требуется указать его в поле `on-error`. Это поле может быть указано для шага, действия или условия, более глубокое указание имеет приоритет. `on-error` является списком названий обработчиков `handler` и типов ошибок `error-type`, для которых нужно использовать данный обработчик. Типы ошибок `error-type`:

- `"AttemptsLimitReached"` — ошибка, вызываемая при достижении ограничения в `attempts`;
- `"*"` — любой тип ошибки.

Пример

```
vars:
  <...>
steps:
  step1:
    conditions:
      - # <...>
        on-error: # выбор нужного обработчика ошибок для условия
        - error-type: "AttemptsLimitReached" # тип ошибки – достижение ограничения
          handler: "notify" # название обработчика
    action:
      # <...>
        on-error: # выбор нужного обработчика ошибок для действия
        - error-type: "*" # любой тип ошибки
          handler: "_continue" # название обработчика
        on-error: # выбор нужного обработчика ошибок для шага
        - error-type: "AttemptsLimitReached" # тип ошибки – достижение ограничения
          handler: "notify" # название обработчика
    error-handlers:
      <...>
```

Обработчики ошибок могут указывать, что делать после обработки ошибки. Для этого используется поле `control-flow` со следующими возможными вариантами:

- `repeat` – повторить шаг еще раз;
- `continue` – пропустить шаг и перейти к следующему;
- `exit` – завершить процедуру с статусом `SUCCESS`;
- `terminate` – завершить процедуры с статусом `ERROR`. Вариант по умолчанию.

Пример

```
vars:
  <...>
steps:
  step1:
    conditions:
      - # <...>
        on-error:
          - error-type: "AttemptsLimitReached"
            handler: "notify"
    action:
      # <...>
        on-error:
          - error-type: "*"
            handler: "_continue"
        on-error:
          - error-type: "AttemptsLimitReached"
            handler: "notify"
error-handlers:
  notify: # название обработчика
  error-type-default: "*" # любой тип ошибки по умолчанию
  apicall: "<...>" # команда, выполняющая запрос с синтаксисом sdc-cli
  control-flow: "continue" # действие после обработки ошибки
```

Встроенные обработчики ошибок

Для всех процедур по умолчанию определены следующие обработчики ошибок

```
error-handlers:
  _terminate:
    control-flow: "terminate" # завершить процедуры с статусом ERROR
  _exit:
    control-flow: "exit" # завершить процедуру с статусом SUCCESS
  _continue:
    control-flow: "continue" # пропустить шаг и перейти к следующему
  _repeat:
    control-flow: "repeat" #повторить шаг еще раз
```

Логирование

В рамках любой команды можно добавить поле `log`, добавляющие сообщение в лог исполнения процедуры. `log` может быть `before` и `after`, то есть до и после исполнения команды. Если не указывать `log` явно, то по умолчанию используется `after`.

Сообщение лога — шаблон, поэтому в нем можно использовать любые доступные переменные. Если в команде используется распаковка, то по умолчанию `log` будет вызван только один раз: в начале и в конце всех команд. Если в сообщении используется распаковка, то `log` будет вызван для каждой отдельной команды.

Пример

```
vars:
  somevar:
    type: "object"
    default: "{}"
steps:
  step1:
    conditions:
      - apicall: "<...>"
        log: "step1, condition 1, after: {{ somevar }}"
    action:
      apicall: "<...>"
      parse:
        - filter: "<...>"
          op: "store"
          var: "somevar"
      log:
        before: "step1, action, before: {{ somevar }}"
        after: "step1, action, after: {{ somevar }}"
```

Результат

```
step1, condition 1, after: {}
step1, action, before: {}
step1, action, after: <...>
```

Конфигурация `config`

Поле `config` позволяет указать дополнительные настройки процедуры.

```
config:
  run-on-node: <node_uuid> # Процедура будет исполняться на указанном узле
  run-on-vip: <bool> # Процедура будет исполняться только на vip-узле
  timeout: <int> # Максимальное время исполнения процедуры в секундах, по истечении которого
процедура завершится
```

Расширения JMESPath

Стандартный набор функций `jmespath`, который используется для преобразования результатов выполнения команд, ограничен, поэтому для реализации более сложных процедур введены следующие расширения

`union(array1, array2)`

Объединение массивов

```
filter: "union({{array1}}, {{array2}})"
```

Sharx Base 6.3. Руководство администратора

```
# intersection(array1, array2)
```

Пересечение массивов (общие элементы)

```
filter: "difference({{all_nodes}}, {{drained_nodes}})"
```

```
# difference(array1, array2)
```

Разность массивов (элементы из array1, которых нет в array2)

```
filter: "difference({{all_nodes}}, {{drained_nodes}})"
```

Дополнительная информация

1. [Управление процедурами.](#)
2. Примеры процедур описаны в [следующей статье](#) .

26.4. Примеры процедур

Статья содержит готовые примеры процедур для задач администрирования кластера. Каждый пример включает:

- Постановку задачи.
- Подробное объяснение структуры.
- Параметры запуска.
- Рекомендации по использованию.

Используйте эти примеры как шаблоны для создания собственных процедур.

26.4.1. 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.

YAML-ОПИСАНИЕ ПРОЦЕДУРЫ

Sharx Base 6.3. Руководство администратора

```
vars:
  value:
    type: "number"
  vcl:
    type: "string"
    visibility: "public"
  vmname:
    type: "string"
    visibility: "public"
  state:
    type: "string"
    visibility: "public"
steps:
  save_value:
    action:
      apicall: "scheduler request show --name {{ vmname }} --vcluster {{ vcl }} --kind vm --
metrics yes"
    parse:
      - filter: "metrics.libvirt_domain_info_vmstate[].value[] | [-1]"
        op: "store"
        var: "value"
    log:
      after: "{{ value }}"
  reboot-on-error:
    conditions:
      - apicall: "scheduler request show --vcluster {{ vcl }} --name {{ vmname }} --metrics ye
s --kind vm"
        expect:
          - filter: "metrics.libvirt_domain_info_vmstate[].value[] | [-1]"
            op: "!="
            value: 1
        attempts:
          max: 1
          interval: 15
        on-error:
          - error-type: AttemptsLimitReached
            handler: _exit
    action:
      apicall: "scheduler request state --vcluster {{ vcl }} --name {{ vmname }} --kind vm --
state {{ state }}"
  wait-for-reboot:
    conditions:
      - apicall: "scheduler request show --vcluster {{ vcl }} --name {{ vmname }} --metrics ye
s --kind vm"
        expect:
          - filter: "metrics.libvirt_domain_info_vmstate[].value[] | [-1]"
            op: "=="
            value: 1
        attempts:
          max: 3
          interval: 60
```

ПОДРОБНОЕ ОБЪЯСНЕНИЕ

Переменные vars:

Sharx Base 6.3. Руководство администратора

1. `value` (number, private) – хранит последнее значение метрики состояния VM.
2. `vc1` (string, public) – имя виртуального кластера.
3. `vmname` (string, public) – имя виртуальной машины.
4. `state` (string, public) – целевое состояние VM.

Шаг 1: `save_value`

Получить текущее состояние VM и сохранить в переменную.

Действие:

1. Выполняет команду `scheduler request show` с параметрами:

- `--name {{ vmname }}` – имя VM;
- `--vcluster {{ vc1 }}` – имя виртуального кластера;
- `--kind vm` – тип ресурса;
- `--metrics yes` – с метриками.

2. JMESPath выражение

```
"metrics.libvirt_domain_info_vmstate[].value[] | [-1]"
```

Берет массив значений метрики `libvirt_domain_info_vmstate`, выбирает последний элемент `([-1])`, сохраняет в переменную `value`.

3. Логирует полученное значение.

Шаг 2: `reboot-on-error`

Проверить, что VM не в состоянии 1 (`running`), и если нет – перезагрузить.

Условия:

1. Проверяет состояние VM той же командой, что в и в `save_value`.
2. Ожидает последнее значение метрики $\neq 1$.
3. Попытки: максимум 1 попытка с интервалом 15 секунд.
4. При ошибке: если достигнут лимит попыток происходит выход из процедуры `_exit`.

Действие:

Выполняет перезагрузку VM

```
scheduler request state --vcluster {{vc1}} --name {{vmname}} --kind vm --state {{state}}
```

Шаг 3: `wait-for-reboot`

Дождаться успешной перезагрузки VM.

Условия:

1. Проверяет состояние VM.
2. Ожидает последнее значение метрики = 1 (`running`).
3. Попытки: максимум 3 попытки с интервалом 60 секунд.

КАК ИСПОЛЬЗОВАТЬ

Возможны следующие варианты использования:

1. Запуск процедуры вручную

```
checker procedure execute --name vm_monitoring
                        --vars '{
                            "vcl": "production-vdc",
                            "vmname": "web-server-01",
                            "state": "reboot"
                        }'
```

2. Планирование проверки каждые 5 минут

```
checker procedure schedule add --name vm_monitoring
                               --vars
                               '{"vcl": "production-vdc", "vmname": "web-server-01", "state": "reboot"}'
                               --cron "* /5 * * * *"
```

26.4.2. Фенсинг узлов кластера

Процедура состоит из двух частей:

1. Основная процедура [Координация процесса](#).
2. Процедура `fence_node` [Обработка конкретного узла](#).

Важно

Процедуры **Фенсинг узлов кластера** и **Обработка конкретного узла** предназначены только для пространства ВЦОД управления.

Не делегируйте их в пользовательские ВЦОД — это может привести к сбоям в работе сервисов

ЗАДАЧА

Автоматическое восстановление работоспособности кластера при отказе физических узлов:

Sharx Base 6.3. Руководство администратора

- Обнаружение «упавших» узлов.
- Параллельный фенсинг всех проблемных узлов.
- Перераспределение VM на рабочие узлы.

ОСНОВНАЯ ПРОЦЕДУРА КООРДИНАЦИЯ ПРОЦЕССА

YAML-описание процедуры

```
config:
  run-on-vip: true

vars:
  exclude_current_node:
    type: boolean
    default: false
    visibility: public
  down_nodes:
    type: array[string]

steps:
  collect-down-nodes:
    conditions:
      # Выходим, если все узлы рабочие, и не требуется мигрировать vm
      - apicall:
          - "cluster nodes list"
          - "scheduler labels list"
        expect:
          - filter: 'length(difference([0][?status==`DOWN`].uuid, [1]."system_drain_node=yes".
nodes || `[]`))'
            op: ">"
            value: 0
        attempts:
          min: 2
          max: 2
          interval: 10
        on-error:
          - error-type: AttemptsLimitReached
            handler: _exit
    action:
      apicall:
        - "cluster nodes list"
        - "scheduler labels list"
      parse:
        - filter: 'difference([0][?status==`DOWN`].uuid, [1]."system_drain_node=yes".nodes ||
`[]`)'
            op: "store"
            var: "down_nodes"
        log: "Found down nodes: {{down_nodes}}"
    process-down-nodes:
      action:
        apicall: 'checker procedure execute --name fence_node --vars '{ "node": {{*down_nodes}}
, "exclude_current_node": {{exclude_current_node}} }' --json yes'
        parse:
          - filter: "@"
            op: "push"
```

Sharx Base 6.3. Руководство администратора

```
var: "@deps" # Добавляем подпроцедуру как зависимую для текущей процедуры
check-errors: # К этому шагу переходим только когда все подпроцедуры завершились
conditions:
  - const: "{{@deps_status}}" # Читаем текущие статусы подпроцедур
    expect:
      - filter: "length([?task_status != `SUCCESS`])"
        op: "=="
        value: 0
    attempts:
      max: 1
    log:
      before: "{{*@deps_status | (task_status != `SUCCESS` && join(':', ', ', ['Error', task_name || '', run_error || '', task_message || ''])) || '}'}"
```

Подробное объяснение

Конфигурация выполняется только на VIP-узле для гарантии доступности

```
config:
  run-on-vip: true
```

Переменные:

1. `exclude_current_node` (boolean, public, default: false) – исключать ли текущий узел из доступных для миграции VM.
2. `down_nodes` (array[string], private) – массив UUID "упавших" узлов.

Шаг 1: `collect-down-nodes`

Найти все узлы со статусом `DOWN`, которые не находятся на обслуживании.

Условия:

1. Выполняет две команды параллельно:
2. `cluster nodes list` – список всех узлов;
3. `scheduler labels list` – список меток для поиска узлов на обслуживании.
4. JMESPath выражение

```
'length(difference([0][?status==`DOWN`].uuid, [1]."system_drain_node=yes".nodes || `[]`)) > 0'
```

- `[0][?status==`DOWN`].uuid` – UUID всех узлов со статусом `DOWN`;
- `[1]."system_drain_node=yes".nodes` – узлы с меткой обслуживания;
- `difference(...)` – узлы в статусе `DOWN`, но не на обслуживании;
- `length(...) > 0` – проверка, что такие узлы есть.

Sharx Base 6.3. Руководство администратора

5. Попытки. Ровно 2 попытки с интервалом 10 секунд для того, чтобы убедиться, что узел действительно «упал».
6. При ошибке, если после 2 попыток нет узлов для фенсинга, происходит выход из процедуры `_exit`.

Действие:

1. Повторно выполняет те же команды.
2. Сохраняет найденные UUID узлов в переменную `down_nodes`.
3. Логирует результат `"Found down nodes: [uuid1, uuid2, ...]"`.

Шаг 2: `process-down-nodes`

Запустить подпроцедуру фенсинга для каждого проблемного узла.

Действие:

1. Запускает подпроцедуру `fence_node` для всех узлов параллельно

```
checker procedure execute --name fence_node --vars '{"node": [uuid1, uuid2],
"exclude_current_node": false}' --json yes
```

2. Распаковка массива `{{*down_nodes}}` создаст параметр `"node": [uuid1, uuid2]`.
3. Параметр `--json yes` — результат возвращается в машиночитаемом формате.
4. Сохранение в зависимости: результат UUID подзадачи добавляется в `@deps`.
5. Основная процедура будет ждать завершения всех подпроцедур.

Шаг 3: `check-errors`

Проверить результаты выполнения всех подпроцедур.

Условия:

1. Использует встроенную переменную `{{@deps_status}}` — статусы всех зависимостей.
2. JMESPath выражение

```
"length([?task_status != `SUCCESS`]) == 0"
```

Подсчитывает подпроцедуры со статусом `≠ SUCCESS` и ожидает, что таких подпроцедур нет `== 0`.

3. Логирование ошибок

```
"{{*@deps_status | (task_status != `SUCCESS` && join(':', ' ', ['Error', task_name || '', run_e
rror || '', task_message || ''])) || '}'"
```

Sharx Base 6.3. Руководство администратора

Для каждой неудачной подпроцедуры формирует сообщение об ошибке в формате "Error: <имя_задачи>: <ошибка>: <сообщение>".

Как использовать

Возможны следующие варианты использования:

1. Запустить с исключением текущего узла из доступных для миграции

```
checker procedure execute --name cluster_fencing
                          --vars '{"exclude_current_node": true}'
```

2. Фенсинг по расписанию не рекомендован, тк является критичной операцией и может привести к нежелательным последствиям:

- если мониторинг узлов даст сбой, процедура может начать фенсить работоспособные узлы;
- фенсинг перезагружает узлы и мигрирует VM, что может нарушить работу сервисов.

ПОДПРОЦЕДУРА ОБРАБОТКА КОНКРЕТНОГО УЗЛА fence_node

Важно

Процедура **Обработка конкретного узла** предназначена только для пространства ВЦОД управления.

Не делегируйте ее в пользовательские ВЦОД — это может привести к сбоям в работе сервисов

YAML-описание процедуры

```
config:
  run-on-vip: true
  timeout: 600 # Максимум 10 мин на работу этой процедуры

vars:
  # Задаем при запуске процедуры
  node:
    type: string
    visibility: public
  exclude_current_node:
    type: boolean
    visibility: public

steps:
  restart-node:
    action:
      apicall: "commands ipmi power reset --node_uuid {{node}}"
  exclude-node:
```

Sharx Base 6.3. Руководство администратора

```
conditions:
  - const: "{{exclude_current_node}}"
    expect:
      - filter: "@"
        op: "=="
        value: true
    attempts:
      max: 1
    on-error:
      - error-type: AttemptsLimitReached
        handler: _continue
action:
  apicall: "scheduler drain add --nodes {{node}}"
reschedule-vms:
action:
  apicall: "scheduler request reschedule --node_uuid {{node}}"
  parse:
    - filter: "@"
      op: "push"
      var: "@deps"
reininclude-node:
conditions:
  - const: "{{exclude_current_node}}"
    expect:
      - filter: "@"
        op: "=="
        value: true
    attempts:
      max: 1
    on-error:
      - error-type: AttemptsLimitReached
        handler: _continue
action:
  apicall: "scheduler drain del --nodes {{node}}"
check-errors:
conditions:
  - const: "{{@deps_status}}"
    expect:
      - filter: "length([?task_status != `SUCCESS`])"
        op: "=="
        value: 0
    attempts:
      max: 1
    log:
      before: "{{*@deps_status | (task_status != `SUCCESS` && join(':', ', ', ['Error', task_name || '', run_error || '', task_message || ''])) || '}}"
```

Подробное объяснение

Конфигурация выполняется на VIP-узле с таймаутом 10 минут (600 с)

```
config:
  run-on-vip: true
  timeout: 600
```

Sharx Base 6.3. Руководство администратора

Переменные:

- `node` (string, public) – UUID узла для фенсинга;
- `exclude_current_node` (boolean, public) – параметр из основной процедуры.

Шаг 1: `restart-node`

Выполнить `hard reset` узла через IPMI.

Действие:

1. Отправляет команду

```
commands ipmi power reset --node_uuid {{ node }}
```

2. Не ждёт завершения перезагрузки, только отправляет команду.
3. Если команда не отправлена, процедура завершается с ошибкой.

Шаг 2: `exclude-node`

Если `exclude_current_node == true`, добавить метку `drain` на узел.

Условия:

1. Проверяет значение `{{ exclude_current_node }}`.
2. Ожидание: значение = `true`.
3. При ошибке: если условие не выполняется, то пропуск шага `_continue`.

Действие:

1. Выполняет команду

```
scheduler drain add --nodes {{ node }}
```

2. Добавляет метку `system_drain_node=yes`.
3. Scheduler не будет размещать VM на этой узле.

Шаг 3: `reschedule-vm`

Запустить миграцию всех VM с проблемного узла.

Действие:

1. Выполняет команду

```
scheduler request reschedule --node_uuid {{node}}
```

Sharx Base 6.3. Руководство администратора

1. Отвязывает все VM от указанного узла.
2. Scheduler автоматически перераспределяет их на другие узлы.
3. Сохранение в зависимости: результат `UUID задачи` добавляется в `@deps`.

Текущий шаг будет ждать завершения миграции VM.

Шаг 4: `reinclude-node`

Если ранее добавляли метку `drain`, необходимо удалить ее.

Условия и действие: Аналогично шагу `exclude-node`, но для удаления метки:

```
scheduler drain del --nodes {{node}}
```

Шаг 5: `check-errors`

Проверить результат миграции VM.

Условия:

1. Использует `{{@deps_status}}` – статус задачи миграции VM.
2. Ожидание: все задачи миграции завершены успешно `SUCCESS`.
3. Логирование: аналогично основной процедуре, формирует сообщения об ошибках.

Как использовать подпроцедуру отдельно

Ручной запуск фенсинга для конкретного узла кластера

```
checker procedure execute --name fence_node
                          --vars '{
                            "node": "uuid_current_node,
                            "exclude_current_node": true
                          }'
```