Руководство пользователя
системы серверной виртуализации

Р-Виртуализация

ООО «Р-Платформа»
ОГРН 1167746349858, ИНН 9715253528
Россия, г. Москва,
Вн.Тер.г. Муниципальный Округ Пресненский, Ул. Садовая-Кудринская д.11, стр.1
Тел.: 8-800-700-7460
www.rosplatforma.ru

Содержание

Основы ПК Р-Виртуализация 10
Обзор 10
Виртуализация на уровне ОС 10
Основы виртуализации на уровне ОС 10
Контейнеры 11
Дедупликация памяти и операций ввода-вывода 13
Шаблоны 14
Виртуализация на аппаратном уровне 14
Основы аппаратной виртуализации 15
Виртуальные машины 16
Аппаратное оборудование виртуальной машины 16
Файлы виртуальной машины 17
Поддержка виртуальных и реальных носителей 18
Конфигурация ПК Р-Виртуализация 18
Управление ресурсами 19
Лицензирование 19
Доступность физического сервера 20
Управление виртуальными средами 21
Создание виртуальных сред 21
Выбор EZ-шаблонов ОС для контейнеров 21
Создание контейнеров 22
Создание виртуальных машин 23
Поддерживаемые гостевые операционные системы 23
Первичная настройка виртуальных сред 24
Использование cloud-init для инициализации виртуальных машин 25
Установка гостевых инструментов 25
Настройка сети 27
Создание учетных записей пользователей виртуальных сред 28
Настройка параметров запуска 29

Запуск, остановка, перезапуск и запрос статуса виртуальных сред 29
Просмотр списка виртуальных сред 30
Клонирование виртуальных сред 30
Создание связанных клонов 31
Настройка стандартных директорий 31
Приостановка виртуальных сред 32
Выполнение команд в виртуальных средах 32
Удаление виртуальных сред 33
Просмотр подробной информации о виртуальных средах 34
Управление резервными копиями виртуальных сред 34
Создание резервных копий для виртуальных сред 35
Просмотр списка резервных копий виртуальных сред 36
Восстановление виртуальных сред из резервных копий 36
Удаление резервных копий виртуальных сред 38
Резервное копирование целых серверов 38
Подсоединение резервных копий к виртуальным средам 39
Отсоединение резервных копий от виртуальных сред 41
Управление шаблонами 42
Создание шаблонов 42
Просмотр списка шаблонов 43
Развертывание шаблонов 43
Хранение шаблонов в ПК Р-Хранилище 44
Управление снапшотами 44
Создание снапшотов 45
Просмотр списка снапшотов 46
Возврат к снапшотам 47
Удаление снапшотов 47
Миграция виртуальных сред 48
Типы миграции 49
Миграция виртуальных сред между серверами с новой версией ПК Р-Виртуализация 51
Миграция виртуальных сред с ПК Р-Виртуализация старой версии на ПК Р-Виртуализация новой версии 51
Миграция шаблонов виртуальных сред 52
Миграция EZ-шаблонов 52

Выполнение операций для контейнеров 53
Переустановка контейнеров 53
Изменение параметров переустановки контейнеров 54
Настройка VPN для контейнеров 55
Установка NFS-сервера в контейнерах 55
Монтирование общего NFS-ресурса при запуске контейнера 56
Управление виртуальными дисками контейнеров 56
Перезапуск контейнеров 59
Создание контейнеров на базе SimFS 59
Выполнение операций для виртуальных машин 60
Пауза виртуальных машин 60
Управление устройствами виртуальных машин 60
Создание скриншотов 67
Настройка диапазонов IP–адресов для сетей Host-Only 67
Настройка режима отказа виртуальных машин 68
Управление ресурсами 69
Управление ресурсами ЦП 69
Настройка единиц ЦП 69
Настройка привязки виртуальных сред к ЦП 70
Настройка лимитов ЦП 70
Привязка ЦП к узлам NUMA 72
Включение горячего подключения ЦП для виртуальных машин 73
Настройка топологии ЦП для виртуальных машин 74
Управление дисковыми квотами 74
Управление виртуальными дисками 75
Изменение емкости диска 75
Сжатие дисков 76
Управление интерфейсами дисков виртуальных машин 76
Управление ресурсами и пропускной способностью сети 77
Параметры сетевого трафика 77
Настройка сетевых классов 78
Просмотр статистики сетевого трафика 79
Настройка управления трафиком 80
Управление параметрами дискового ввода-вывода 84
Настройка уровней приоритета 84

Настройка пропускной способности дискового ввода-вывода ... 84
Настройка количества операций ввода-вывода в секунду ... 85
Просмотр статистики дискового ввода-вывода ... 86
Настройка лимитов ввода-вывода для операций резервного копирования и миграции 87
Управление параметрами памяти контейнеров ... 87
Настройка основных параметров VSwap ... 88
Настройка гарантированной памяти контейнеров ... 89
Настройка ограничения выделенной памяти для контейнеров ... 89
Настройка поведения OOM Killer для контейнеров ... 90
Настройка VSwap ... 91
Управление параметрами памяти виртуальных машин ... 91
Настройка размера памяти виртуальных машин ... 91
Настройка размера видеопамяти виртуальных машин ... 92
Включение горячего подключения памяти для виртуальных машин ... 92
Настройка гарантированной памяти виртуальных машин ... 93
Управление конфигурацией ресурсов контейнеров ... 93
Разделение сервера на равные части ... 94
Применение новых образцов конфигурации к контейнерам ... 94
Управление конфигурацией ресурсов виртуальных машин ... 95
Создание образца конфигурации ... 96
Применение образцов конфигурации к виртуальным машинам ... 96
Параметры, применяемые из образцов конфигурации ... 96
Мониторинг ресурсов ... 97
Управление службами и процессами ... 98
Что представляют собой службы и процессы ... 98
Основные операции с процессами и службами ... 99
Управление процессами и службами ... 100
Просмотр активных процессов и служб ... 100
Мониторинг процессов в реальном времени ... 102
Определение UUID контейнеров по ID процессов ... 103
Управление сетью ... 104
Управление сетевыми адаптерами на физическом сервере ... 104
Сетевые режимы в ПК Р-Виртуализация ... 104
Сетевые режимы контейнеров ... 105
Сетевые режимы виртуальных машин ... 108

Различия между режимами Host-Routed и Bridged ... 111
Настройка виртуальных машин и контейнеров в режиме Host-Routed ... 112
Настройка IP-адресов ... 112
Настройка адресов DNS-сервера ... 112
Настройка доменов поиска DNS ... 112
Переключение адаптеров виртуальных машин в режим Host-Routed ... 113
Настройка виртуальных машин и контейнеров в режиме Bridged ... 113
Управление виртуальными сетями ... 113
Управление виртуальными сетевыми адаптерами в виртуальных средах ... 120
Управление лицензиями ... 123
Установка лицензии ... 123
Просмотр лицензии ... 123
Статусы лицензии ... 124
Поддержание системы в актуальном состоянии ... 125
Обновление ПК Р-Виртуализация ... 126
Обновление всех компонентов ... 126
Обновление ядра ... 126
Обновление гипервизора KVM/QEMU в виртуальных машинах ... 127
Обновление EZ-шаблонов ... 128
Проверка обновлений ... 128
Дополнительные параметры yum ... 128
Обновление ядра ПК Р-виртуализация с помощью ReadyKernel ... 129
Автоматическая установка патчей ReadyKernel ... 129
Управление патчами ReadyKernel вручную ... 129
Отключение загрузки патчей ReadyKernel при запуске ... 131
Управление журналами ReadyKernel ... 131
Обновление ПО в виртуальных машинах ... 131
Обновление гостевых инструментов в виртуальных машинах ... 131
Обновление контейнеров ... 132
Обновление пакетов EZ-шаблонов в контейнерах ... 133
Обновление кэша EZ-шаблонов ОС ... 134
Управление кластерами высокой доступности ... 135
Проверка требований для высокой доступности ... 136
Включение и отключение высокой доступности на серверах ... 137
Отключение высокой доступности для отдельных виртуальных сред ... 138

Включение высокой доступности для целей iSCSI ... 138
Отключение высокой доступности на серверах ... 138
Настройка режимов перераспределения ресурсов ... 139
Настройка режимов перераспределения ресурсов на серверах, участвующих в экспорте iSCSI ... 141
Настройка приоритета высокой доступности для виртуальных сред ... 141
Управление пулами ЦП ... 142
Добавление серверов в пулы ЦП ... 142
Мониторинг пулов ЦП ... 144
Исключение серверов из пулов ЦП ... 144
Мониторинг статуса кластера ... 144
Управление ресурсами кластера с помощью скриптов ... 146
Дополнительные задачи ... 147
Настройка политик управления автоматической памятью ... 147
Оптимизация памяти виртуальных машин с помощью KSM ... 148
Управление службами хоста с помощью VCMMD ... 149
Управление службами ПК Р-Хранилище с помощью VCMMD ... 149
Создание специализированных контейнеров ... 150
Использование функции образа типа golden ... 151
Использование специализированных EZ-шаблонов ... 152
Создание RPM-пакетов специализированных EZ-шаблонов ... 154
Установка Docker в контейнеры ПК Р-Виртуализация ... 154
Установка Docker для запуска в режиме роя ... 155
Ограничения для Docker в контейнерах ПК Р-Виртуализация ... 156
Управление шифрованием виртуального жесткого диска контейнеров ... 156
Установка генератора запросов для ключей шифрования ... 157
Шифрование и дешифрование виртуальных жестких дисков контейнеров ... 157
Шифрование системной подкачки ... 158
Предоставление доступа по VNC к виртуальным средам ... 159
Защита VNC-соединения с помощью SSL ... 159
Предоставление доступа по VNC к виртуальным машинам ... 160
Предоставление доступа по VNC к контейнерам ... 160
Подключение с помощью VNC-клиента ... 160
Управление модулями iptables ... 161
Использование модулей iptables в ПК Р-Виртуализация ... 161

Использование модулей iptables в контейнерах ... 161
Использование SCTP в контейнерах и виртуальных машинах ... 162
Создание конфигурационных файлов для новых дистрибутивов Linux ... 163
Выравнивание дисков и разделов в виртуальных машинах ... 164
Выравнивание разделов ... 165
Проверка выравнивания разделов в виртуальных машинах ... 165
Выравнивание дисков для виртуальных машин Linux ... 166
Выравнивание разделов для виртуальных машин Windows ... 167
Создание шаблона виртуальной машины с выровненными разделами ... 168
Удаление гостевых инструментов из виртуальных машин ... 169
Удаление гостевых инструментов из виртуальных машин Linux ... 169
Удаление гостевых инструментов из виртуальных машин Windows ... 169
Включение режима отладки для устаревших виртуальных машин ... 171
Установка дополнительных пакетов ПК Р-Виртуализация ... 172
Включение вложенной виртуализации в виртуальных машинах ... 172
Устранение неисправностей ... 174
Общие положения ... 174
Устранение неисправностей ядра ... 176
Использование последовательностей клавиш ALT+SYSRQ ... 176
Сохранение ошибок ядра (OOPS) ... 176
Обнаружение функции ядра, вызвавшее у процесса состояние D ... 178
Проблемы управления контейнерами ... 178
Отказ при запуске контейнера ... 178
Отказ в доступе к контейнеру по сети ... 179
Отказ при входе в контейнер ... 179
Получение технической поддержки ... 180

Глава 1

Основы ПК Р-Виртуализация

В данной главе дается краткое описание Программного Комплекса Р-Виртуализация, виртуальных машин и контейнеров, требований к ним и используемых технологий.

Обзор

Программный Комплекс Р-Виртуализация представляет собой решение виртуализации, которое устанавливается на сервер, не имеющий операционной системы, и включает в себя контейнерную виртуализацию, а также гипервизор для создания виртуальных машин на основе KVM. ПК Р-Виртуализация работает поверх собственного дистрибутива на основе RHEL.

ПК Р-Виртуализация является лучшим решением для организаций, которые стремятся минимизировать издержки, и позволяет:

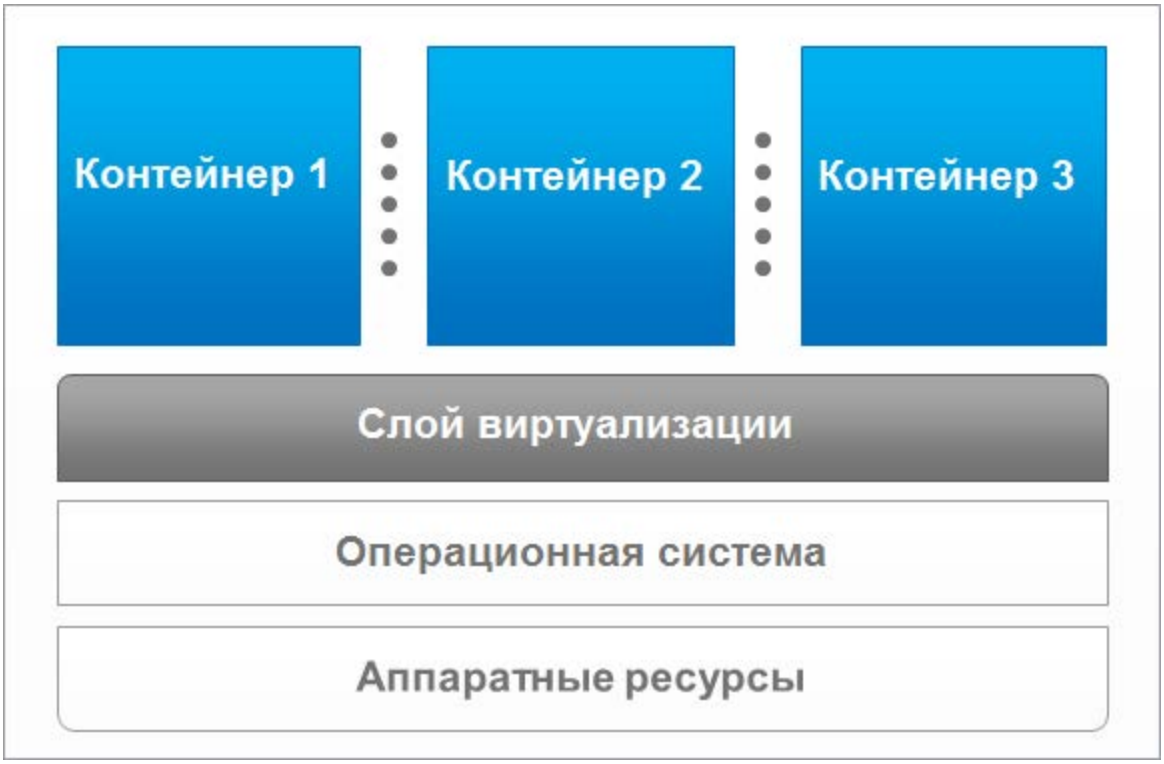
- стандартизировать аппаратные платформы серверов,
- эффективно совместить серверные ресурсы,
- объединить и поддерживать ОС и приложения прежних версий,
- облегчить развертывание, обслуживание серверов и приложений, а также управление ими,
- упростить тестирование и разработку программного обеспечения,
- оптимизировать доступность серверов и приложений.

Виртуализация на уровне ОС

В данном разделе представлена информация о виртуализации на уровне операционной системы, которая отвечает за поддержку контейнерной виртуализации.

Основы виртуализации на уровне ОС

Виртуализация на уровне ОС позволяет виртуализировать физические серверы на уровне операционной системы (ядра). На схеме ниже изображена архитектура данного типа виртуализации.



Слой виртуализации обеспечивает изоляцию и безопасность ресурсов в разных контейнерах. Каждый контейнер представляется в виде изолированного сервера со своими собственными приложениями и рабочей загрузкой. Виртуализация на уровне ОС оптимизирована для лучшей производительности, управления и надежности и имеет следующие преимущества:

- Производительность контейнеров соответствует производительности физического сервера, на котором они находятся. Контейнеры не имеют виртуализованных аппаратных ресурсов, но используют аппаратные ресурсы сервера, обеспечивая за счет этого более высокую производительность.
- Ресурсы контейнера могут быть увеличены до ресурсов всего физического сервера.
- Технология виртуализации на уровне ОС обеспечивает самую высокую плотность из всех решений виртуализации.
- Контейнеры используют одну операционную систему, что значительно упрощает их обслуживание и обновление.

Контейнеры

С точки зрения пользователей и приложений контейнер является независимой системой. Эта независимость обеспечивается слоем виртуализации. Следует отметить, что виртуализация потребляет только незначительную часть ресурсов ЦП. Отличительными чертами виртуализации на уровне ОС, используемой в ПК Р-Виртуализация, являются:

- Сходство контейнера с обычной системой Linux. Контейнер имеет стандартные загрузочные скрипты и позволяет запускать сторонние приложения без их модификации.
- Возможность для пользователя изменять любой конфигурационный файл системы.
- Полная изоляция контейнеров друг от друга (файловая система, процессы, переменные `sysctl`).
- Общее использование динамических библиотек для экономии памяти.
- Распределение процессов контейнера, находящихся в очереди на выполнение, по всем доступным ЦП. Таким образом, контейнеры не привязаны к одному ЦП и могут использовать всю доступную производительность сервера.

Двумя главными составляющими контейнера являются его содержимое и конфигурация. По умолчанию все файлы контейнера хранятся на физическом сервере в директории `/vz/private/<UUID>`, также называемой *собственной областью* контейнера.

Имя файла	Описание
`/vz/private/<UUID>`	Собственная область контейнера.
`/vz/private/<UUID>/root.hdd/root.hdd`	Виртуальный жесткий диск с содержимым контейнера. Максимальный размер виртуального жесткого диска 50 ТБ.
`/vz/root/<UUID>`	Точка монтирования контейнера.
`ve.conf`	Конфигурационный файл контейнера: • Является символьной ссылкой для `/etc/vz/conf/<UUID>.conf`. • Определяет параметры контейнера, такие как ограничения на выделенные ресурсы, IP-адрес, имя хоста и др. • Заменяет соответствующие параметры в глобальном конфигурационном файле.

Все файлы контейнера хранятся в виде единого образа (`/vz/private/<UUID>/root.hdd/root.hdd`), который схож с жестким диском виртуальной машины и позволяет:

- Упростить операции миграции и резервного копирования, так как последовательный доступ к образу контейнера осуществляется быстрее, чем к отдельным его файлам.
- Не использовать шаблоны ОС и приложений после создания контейнера.
- Использовать журналируемые дисковые квоты Linux, которые не требуют повторного вычисления после сбоев.

Примечание: Использование контейнеров, которые хранят все файлы в файле образа, поддерживается только для разделов `/vz` формата `ext4`.

Аппаратное оборудование контейнера

Контейнер может иметь следующее виртуальное аппаратное оборудование:

Оборудование	Теоретически	Сертифицировано
ЦП	Не более количества всех потоков на сервере.	Не более 64
ОЗУ	Не более размера всего физического ОЗУ сервера	Не более 1 ТБ
Жесткие диски	Не более 15: жесткие диски, подключенные к файлам образов QCOW2, и DVD-диски, подключенные к файлам ISO-образов, не более 50 ТБ каждый	
Сетевые интерфейсы	Не более 15	

Дедупликация памяти и операций ввода-вывода

ПК Р-Виртуализация поддерживает дедупликацию страниц памяти и операций ввода-вывода, тем самым помогая уменьшить использование памяти и операций ввода-вывода на сервере и увеличить максимальное число контейнеров.

Дедупликация обеспечивается R-Virtualization File Cache, который включает демон `pfcached` и образ ploop, смонтированный в директорию на сервере. Ploop кэширования файлов содержит копии пригодных файлов, находящиеся в контейнерах. Чтобы быть пригодными для кэширования, файлы в контейнерах должны соответствовать определенным конфигурируемым требованиям, например, считываться в определенном числе контейнеров, быть определенного размера, храниться в определенных директориях контейнеров.

Когда ядро получает запрос на чтение файла, который находится в ploop контейнера, оно ищет ploop кэширования файлов для копии данного файла с помощью хеша SHA-1, хранящегося в виде расширенного атрибута файла. Если он находится, то считывается копия в ploop кэширования файлов вместо оригинала файла в ploop контейнера. В противном случае, считывается оригинальный файл в ploop контейнера.

Чтобы включить в ploop кэширования файлов файлы с большим числом обращений, `pfcached` периодически получает от ядра статистику об операциях чтения из файлов контейнера, анализирует ее и копирует файлы, пригодные для ploop кэширования файлов. Если у ploop кэширования файлов заканчивается место, то наименее востребованные файлы удаляются из него.

R-Virtualization File Cache имеет следующие преимущества:

- Дедупликация памяти. Требуется загрузить в память только один файл из ploop кэширования файлов вместо загрузки многочисленных идентичных файлов, находящихся в многочисленных контейнерах.
- Дедупликация операций ввода-вывода. Требуется считать только один файл из ploop кэширования файлов вместо чтения многочисленных идентичных файлов, находящихся в многочисленных контейнерах.

Если на физическом сервере имеются накопители с разной производительностью, например, IDE и SSD, то ploop кэширования файлов лучше работает, если находится на самом быстром накопителе на сервере, например, SSD. В любом случае:

- Если не выделяется больше памяти, чем есть на сервере, кэш файла в основном помогает ускорить запуск контейнера, в процессе которого происходит чтение

большинства файлов. В таком случае кэш, находящийся в памяти, очищается нечасто, поэтому копии в ploop кэширования файлов после их считывания во время запуска контейнера не нужно повторно считывать в ходе работы контейнера.

- Если выделяется больше памяти, чем есть на сервере, R-Virtualization File Cache помогает ускорить не только запуск контейнера, но и его работу. В таком случае кэш, находящийся в памяти, может очищаться часто, поэтому файлы в ploop кэширования файлов нужно повторно считывать также часто.

R-Virtualization File Cache можно управлять при помощи утилиты `pfcache`.

Шаблоны

Шаблон (или набор пакетов) представляет собой набор файлов приложений, перекомпонованных для использования в ПК Р-Виртуализация. Обычно шаблоном является набор пакетов RPM для RHEL-подобных систем. ПК Р-Виртуализация предоставляет инструменты для создания, установки, обновления шаблонов, их добавления к контейнеру и удаления из него.

Использование шаблонов имеет следующие преимущества:

- Размещение похожих приложений, запущенных в разных контейнерах, в общеиспользуемой памяти для экономии памяти.
- Одновременное развертывание приложений во многих контейнерах.
- Использование разных версий приложения в разных контейнерах (например, обновление приложения только в определенных контейнерах).

Доступны два типа шаблонов: ОС и приложений.

- Шаблон ОС представляет собой операционную систему со стандартным набором приложений. Шаблоны ОС используются для создания новых контейнеров с предустановленной операционной системой.
- Шаблон приложений является набором перекомпонованных пакетов приложений, к которым прилагаются конфигурационные скрипты. Шаблоны приложений используются для добавления приложений в существующие контейнеры.

Например, можно создать контейнер по шаблону ОС `redhat` и добавить в него приложение MySQL с помощью шаблона `mysql`.

Виртуализация на аппаратном уровне

В данном разделе описывается второй компонент ПК Р-Виртуализация—виртуализация на аппаратном уровне, позволяющая создавать виртуальные машины и управлять ими.

Основы аппаратной виртуализации

В основе аппаратной виртуализации лежит гипервизор, который загружается непосредственно на сервер без ОС и выступает в роли посредника между аппаратными ресурсами сервера и виртуальными машинами. Для того чтобы выделить аппаратные средства и ресурсы виртуальным машинам, гипервизор виртуализирует все аппаратные средства на физическом сервере. Используя виртуальные аппаратные ресурсы, виртуальная машина запускает свои копии операционной системы и приложений.

На схеме ниже изображена архитектура аппаратной виртуализации.

ПК Р-Виртуализация использует гипервизор KVM/QEMU и осуществляет управление виртуальными машинами посредством API libvirt.

Аппаратная виртуализация позволяет:

- Создавать виртуальные машины с различными операционными системами на одном физическом сервере.
- Одновременно запускать множество гостевых операционных систем и их приложений на физическом сервере без перезагрузки.
- Объединить и виртуализировать вычислительные среды, сократить расходы на оборудование и эксплуатацию и увеличить производительность.
- Использовать открытые API и SDK для интеграции с собственными и сторонними приложениями.

Виртуальные машины

С точки зрения пользователей и приложений, виртуальная машина (ВМ) представляет собой независимую систему с отдельным набором виртуальных аппаратных ресурсов. Эта независимость обеспечивается гипервизором. Отличительными чертами аппаратной виртуализации, используемой в ПК Р-Виртуализация, являются:

- Сходство виртуальной машины с обычным компьютером. Каждая виртуальная машина имеет собственные виртуальные аппаратные ресурсы и позволяет запускать сторонние приложения без их модификации.
- Возможность для пользователя изменять конфигурацию виртуальной машины (например, добавлять новые виртуальные диски или увеличивать доступную память).
- Полная изоляция виртуальных машин друг от друга (файловая система, процессы, переменные `sysctl`).
- Возможность установить в виртуальную машину любую поддерживаемую гостевую операционную систему. Гостевая ОС и ее приложения изолированы в виртуальной машине и имеют общие аппаратные ресурсы сервера вместе с другими виртуальными машинами.

Поддержка технологий виртуализации Intel

ПК Р-Виртуализация обеспечивает поддержку технологии виртуализации Intel, которая оптимизируют использование процессора и работу решений виртуализации. Использование этой технологии позволяет снизить загрузку аппаратных ресурсов, обеспечивая высокую производительность гостевых операционных систем.

Аппаратное оборудование виртуальной машины

Виртуальная машина работает как обычный автономный компьютер.

По умолчанию созданные виртуальные машины имеют следующие виртуальные аппаратные ресурсы:

- 1 жесткий диск VirtIO SCSI, формата `expanded` (расширяемый),
- 1 CD-ROM (IDE для гостевых ОС Windows и Debian, VirtIO SCSI для гостевых ОС Linux, кроме Debian),
- 1 сетевой адаптер VirtIO, типа bridged (мост),
- Видеокарта с 32МБ памяти.

В зависимости от выбранного дистрибутива в виртуальную машину можно добавить другие аппаратные ресурсы (см. **Создание виртуальных машин** (стр. 23)).

В таблице ниже приведен полный список виртуального оборудования для виртуальной машины.

ЦП	Не более 64
ОЗУ	Не более 1 ТБ
Графический адаптер	Видеоадаптер VGA/SVGA с VBE 3.0
Видеопамять	Не более 256 МБ видеопамяти
Флоппи-диск	Флоппи-диск размером 1.44 МБ, подключенный к файлу образа
IDE-устройства	Не более 4 IDE-устройств: • жесткие диски, подключенные к файлам образов QCOW2 (не более 16 ТБ каждый) • DVD-диски, подключенные к файлам ISO-образов
SCSI-устройства	Не более 15 SCSI-устройств: • жесткие диски, подключенные к файлам образов QCOW2 (не более 16 ТБ каждый) • DVD-диски, подключенные к файлам ISO-образов
VirtIO-устройства	Не более 15 жестких дисков VirtIO, подключенных к файлам образов QCOW2 (не более 16 ТБ каждый)
Сетевые интерфейсы	Не более 15 виртуальных сетевых адаптеров VirtIO (по умолчанию), Intel 82545EM или Realtek RTL8029
Последовательные порты (COM)	Не более 4 последовательных портов (COM), подключенных к сокету, физическому порту или файлу вывода
Клавиатура	Любая USB или PS/2 клавиатура
Мышь	Любая USB или PS/2 мышь с колесом прокрутки

Файлы виртуальной машины

Виртуальная машина имеет как минимум два файла: конфигурационный файл (`.psv`) и образ жесткого диска (`.hdd`). Она также может иметь: файл на каждый дополнительный виртуальный жесткий диск и файлы вывода для виртуальных портов. По умолчанию все файлы виртуальной машины хранятся на физическом сервере в директории `/vz/vmprivate/<`*UUID*`>`.

Список файлов, относящихся к виртуальной машине, представлен в таблице ниже:

Имя файла	Описание
`.pvs`	Конфигурационный файл виртуальной машины. Он определяет конфигурацию оборудования и ресурсов виртуальной машины. Конфигурационный файл автоматически создается в процессе создания виртуальной машины.
`.sav`	Файл дампа, который создается, когда виртуальная машина приостанавливается. Данный файл содержит состояние виртуальной машины и ее приложения на момент приостановки.
`.mem`	Файл дампа памяти для приостановленной виртуальной машины. Для запущенной виртуальной машины он является временным файлом виртуальной памяти.
`.hdd`	Образ жесткого диска в формате QCOW2. Для новой виртуальной машины можно создать новый или использовать уже существующий виртуальный жесткий диск. В виртуальной машине может быть несколько виртуальных жестких дисков.

`.iso`	Образ CD/DVD диска. Виртуальные машины принимают образы ISO за реальные CD/DVD диски.
`.txt`	Файлы вывода для последовательных портов. Файлы вывода `.txt` генерируются, когда последовательный порт, подключенный к файлу вывода, добавляется к конфигурации виртуальной машины.

Поддержка виртуальных и реальных носителей

В данном разделе перечислены типы дисков, которые могут быть использованы виртуальными машинами, и представлена информация об основных операциях, которые можно с ними выполнять.

Поддерживаемые типы жестких дисков

Виртуальные машины могут использовать в качестве жестких дисков только файлы образов виртуальных жестких дисков.

Виртуальные жесткие диски

Емкость виртуального жесткого диска можно задать от 100 МБ до 16 ТБ.

ПК Р-Виртуализация использует расширяемые виртуальные жесткие диски. Файл образа такого диска обычно имеет небольшой размер (меньше, чем заданный размер виртуального диска), который увеличивается по мере добавления в него данных в гостевой ОС.

CD/DVD образы дисков

ПК Р-Виртуализация поддерживает те же образы CD/DVD дисков, что поддерживаются гостевой операционной системой.

Конфигурация ПК Р-Виртуализация

ПК Р-Виртуализация позволяет изменять настройки как всего физического сервера, так и каждого контейнера. Изменяемые настройки включают в себя дисковые и пользовательские квоты, сетевые параметры, директории для хранения файлов по умолчанию, файлы образцов конфигурации и т.д.

Вся конфигурационная информация, относящаяся к виртуализации на уровне ОС, хранится в глобальном конфигурационном файле `/etc/vz/vz.conf`, который определяет такие параметры контейнеров, как шаблоны ОС по умолчанию дисковые квоты, ведение журналов и др.

Конфигурационный файл считывается при запуске ПК Р-Виртуализация и/или контейнера. Настройки контейнера можно также менять “на лету” при помощи стандартных утилит (например, `prlctl`), сохраняя или не сохраняя эти изменения в конфигурационном файле

данного контейнера. Несохраненные изменения будут удалены при перезапуске контейнера.

Управление ресурсами

Управление ресурсами в ПК Р-Виртуализация позволяет менять объемы ресурсов, доступные для виртуальных машин и контейнеров и включающие в себя мощность ЦП, дисковое пространство и память. Управление ресурсами позволяет:

- виртуальным машинам и контейнерам эффективно использовать общие ресурсы физического сервера,
- гарантировать качество обслуживания в соответствии с соглашением об уровне предоставляемой услуги (SLA),
- обеспечивать изоляцию работы и ресурсов и защиту от DoS-атак,
- одновременно назначать и контролировать ресурсы для многих виртуальных машин и контейнеров,
- собирать информацию по потреблению ресурсов для мониторинга состояния системы.

Управление ресурсами в ПК Р-Виртуализация играет важную роль, так как потребление ресурсов сервера в подобной системе значительно выше, чем на обычном компьютере.

Лицензирование

Чтобы начать использовать ПК Р-Виртуализация, необходимо иметь лицензию ПК Р-Виртуализация. Данная лицензия должна быть установлена на сервер после или в процессе установки ПК Р-Виртуализация. Каждый физический сервер с виртуальными машинами и контейнерами должен иметь свою лицензию. Лицензии выдаются ПК Р-Виртуализация и определяют ряд параметров в отношении физического сервера. Основные параметры лицензии перечислены ниже:

- Количество физических ЦП, которое можно установить на физический сервер. Двухъядерный процессор или процессор с hyper-threading считается как один ЦП.
- Дата окончания срока действия лицензии. Лицензия может быть временной или постоянной. Лицензии ПК Р-Виртуализация имеют дату начала срока действия, и если они временные, в них также может быть указана дата окончания срока действия. Следует правильно настроить системное время. В противном случае, валидация лицензии может не удаться.
- Количество виртуальных машин и контейнеров, которое можно одновременно запускать на физическом сервере.
- Платформа и архитектура, совместимые с ПК Р-Виртуализация.

Для получения инструкций по установке, обновлению, просмотру и переносу лицензий см. **Управление лицензиями** (стр. 123).

Доступность физического сервера

Доступность физического сервера с ПК Р-Виртуализация более критична, чем доступность обычного ПК-сервера. Так как на нем запускается большое число виртуальных сред и ряд важных служб, то простой физического сервера может оказаться дорогостоящим. Одновременный отказ нескольких серверов с важными службами может иметь катастрофические последствия.

Для улучшения доступности физического сервера следует придерживаться следующих рекомендаций:

- Следует использовать RAID-хранилище для важных виртуальных сред. Стоит отдавать предпочтение аппаратным RAID, но в крайнем случае программные зеркальные RAID также могут подойти.
- Не следует запускать программы на самом сервере. Нужно создать специальные виртуальные среды для запуска необходимых служб, таких как BIND, FTPD, HTTPD и др. На сервере должен находиться только демон SSH. Рекомендуется, чтобы он принимал соединения только от заранее заданного набора IP-адресов.
- Не следует создавать пользователей на самом сервере. В любой виртуальной среде можно создать любое количество пользователей. Не стоит забывать: угроза серверу также означает угрозу всем виртуальным средам, находящимся на нем.

Глава 2

Управление виртуальными средами

В данной главе описаны основные операции, которые можно выполнять с виртуальными машинами и контейнерами в ПК Р-Виртуализация.

Создание виртуальных сред

Новые виртуальные среды можно создать в ПК Р-Виртуализация с помощью команды `prlctl create`. Параметры, доступные для данной команды, зависят от типа создаваемой виртуальной среды: виртуальная машина или контейнер.

Выбор EZ-шаблонов ОС для контейнеров

Перед созданием контейнера необходимо выбрать EZ-шаблон ОС, по которому он будет создан.

Просмотр списка EZ-шаблонов ОС

Узнать, какие EZ-шаблоны ОС уже установлены на сервере и кэшированы (т.е. готовы к использованию), можно с помощью команды `vzpkg list`. Например:

```
# vzpkg list -O
centos-6-x86_64            2012-05-10 13:16:43
```

Временная метка напротив названия шаблона указывает на дату и время, когда он был кэширован.

Если добавить параметр `-O` после команды `vzpkg list`, то выводится список только тех EZ-шаблонов ОС, которые установлены, но не кэшированы. Для отображения краткого описания шаблонов используется параметр `--with-summary`:

```
# vzpkg list -O --with-summary
centos-6-x86_64            :CentOS 6 (for Intel EM64T) EZ OS Template
```

Установка и кэширование EZ-шаблонов ОС

Некоторые поддерживаемые EZ-шаблоны ОС могут не быть предварительно установлены, тогда необходимо будет выполнить дополнительные действия перед созданием контейнеров на базе данных шаблонов. Чтобы отобразить список шаблонов, доступных для установки в официальных удаленных репозиториях, нужно выполнить следующую команду:

```
# vzpkg list --available
fedora-23-x86_64 factory
sles-11-x86_64 factory
sles-12-x86_64 factory
suse-42.1-x86_64 factory
suse-42.2-x86_64 factory
vzlinux-6-x86_64 factory
```

Чтобы подготовить шаблон для создания контейнеры, необходимо выполнить следующие шаги:

1 Установите пакет шаблона. Например:

```
# vzpkg install template sles-11-x86_64
```

2 При необходимости настройте дополнительные параметры шаблона. Для некоторых EZ-шаблонов могут потребоваться особые подготовительные действия в зависимости от операционной системы. Чтобы подготовить, например, шаблон SLES 11 для создания контейнера, нужно дополнительно задать параметр $RCE и учетные данные репозитория SUSE (полученные от SUSE Customer Center) в файле `/etc/vztt/url.map`:

```
$RCE          %24RCE
$SLES11_PASS  <password>
$SLES11_USER  <user>
```

3 Создайте кэш шаблона:

```
# vzpkg create cache sles-11-x86_64
```

Теперь можно приступить к созданию контейнеров на базе подготовленного шаблона.

Создание контейнеров

Для создания контейнера используется команда `prlctl create`. Например:

```
# prlctl create MyCT --vmtype ct
```

Данная команда создает новый контейнер с именем `MyCT` и стандартными параметрами, указанными в глобальном конфигурационном файле `/etc/vz/vz.conf`, включая операционную систему, которая будет установлена в контейнере.

Создать контейнер с операционной системой отличной от той, что указана в глобальном конфигурационном файле, можно, добавив параметр `--ostemplate` и указав имя EZ-шаблона. Например:

```
# prlctl create MyCT --vmtype ct –-ostemplate centos-6-x86_64
```

Все содержимое контейнера хранится в собственной области данного контейнера. Узнать, где находится собственная область, можно с помощью команды `prlctl list` следующим образом:

```
# prlctl list MyCT -i | grep "Home"
Home: /vz/private/26bc47f6-353f-444b-bc35-b634a88dbbcc
```

Примечания:

1. При первой установке операционной системы в контейнер создается ее кэш. Для этого необходимо иметь активное соединение с Интернетом, чтобы иметь доступ к репозиториям, в которых хранятся пакеты для соответствующей операционной системы. Можно также создать локальный репозиторий пакетов и использовать его для своей операционной системы. Локальный репозиторий пакетов необходим для некоторых коммерческих дистрибутивов (например, Red Hat Enterprise Linux).

2. Для получения информации по созданию контейнеров с заранее установленными приложениями см. **Использование функции образа типа golden** (стр. 151).

Создание виртуальных машин

Создание новой виртуальной машины подразумевает создание конфигурации виртуальной машины на основе указанного дистрибутива.

Для создания виртуальной машины используется команда `prlctl create`. Например:

```
# prlctl create MyVM --distribution win-2008 --vmtype vm
```

Данная команда создает конфигурацию для виртуальной машины `MyVM`, настраивает ее для запуска гостевой операционной системы Windows Server 2008 и помещает все необходимые файлы в директорию `/vz/vmprivate/`<*VM_UUID*>.

Когда конфигурация виртуальной машины готова, необходимо установить в нее поддерживаемую гостевую ОС по VNC (см. **Предоставление доступа по VNC к виртуальным машинам** (стр. 160)).

Примечание: Перед установкой Windows Server 2012 (не R2) в виртуальную машину необходимо вручную монтировать образ с нужными драйверами `floppy_win8.vfd` с помощью команды `prlctl set --device set ffd0 --image <путь_к_файлу>`. В ПК Р-Виртуализация новой версии образ находится в директории `/usr/share/vz-guest-tools`, таким образом, нужно выполнить следующую команду: `prlctl set --device set ffd0 --image /usr/share/vz-guest-tools/floppy_win8.vfd`.

При выборе дистрибутива для установки следует иметь в виду, что ПК Р-Виртуализация поддерживает инициализацию виртуальных машин с помощью cloud-init, что позволяет выполнять некоторые задачи первичной настройки над остановленными виртуальными машинами. Чтобы использовать данную функцию, можно установить дистрибутив " cloud-enabled " вместо обычного дистрибутива. Для получения подробной информации см. **Использование cloud-init для инициализации виртуальных машин** (стр. 25).

Поддерживаемые гостевые операционные системы

ПК Р-Виртуализация поддерживает следующие гостевые операционные системы:

Для виртуальных машин

- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2 с SP 1
- CentOS 7.x (x64)
- CentOS 6.x (x64)
- Debian 9.x (x64)
- Debian 8.x (x64)
- Ubuntu 18.04 (x64)
- Ubuntu 17.10 (x64)
- Ubuntu 16.04 LTS (x64)
- Ubuntu 14.04 LTS (x64)
- CloudLinux 7.x (x64)
- CloudLinux 6.x (x64)

Для контейнеров

- CentOS 7.x (x64)
- CentOS 6.x (x64)
- Debian 9.x (x64)
- Debian 8.x (x64)
- Ubuntu 18.04 (x64)
- Ubuntu 17.10 (x64)
- Ubuntu 16.04 LTS (x64)
- Ubuntu 14.04 LTS (x64)
- SUSE Linux Enterprise Server 12 SP 1 (x64)
- SUSE Linux Enterprise Server 11 SP 1, 2, 3, 4 (x64)

Первичная настройка виртуальных сред

Перед первым запуском созданной виртуальной среды ее необходимо настроить. В данном разделе описываются основные шаги этой настройки.

Использование cloud-init для инициализации виртуальных машин

ПК Р-Виртуализация осуществляет поддержку инициализации виртуальных машин с помощью cloud-init, тем самым позволяя выполнять некоторые задачи первичной настройки, описанные ниже, с остановленными виртуальными машинами. Поддерживаются следующие задачи: установка гостевых инструментов, изменение имени пользователя и пароля, настройка сетевых параметров.

После выполнения перечисленных задач изменения не применяются к виртуальной машине сразу, а сохраняются в виде инструкций, чтобы руководствоваться ими в процессе загрузки гостевой ОС с cloud-init. Поэтому при выполнении соответствующей команды (например, `prlctl set --userpasswd`), происходит следующее: образ вместе с инструкциями cloud-init копируется в домашнюю директорию виртуальной машины, устройство CD-ROM добавляется к виртуальной машине, и образ монтируется в указанный CD-ROM. Однако изменения (например, имени пользователя и пароля) будут применены после установки и начала загрузки гостевой ОС.

Как указано выше, для работы данной функции необходимо иметь гостевую ОС с установленным пакетом cloud-init. Для гостевых ОС Linux самым простым способом получения cloud-init является установка дистрибутива "cloud-enabled". Также cloud-init можно установить вручную (например, с помощью команды `yum install cloud-init` на CentOS 7). Для гостевых ОС Windows можно создать собственные дистрибутивы с cloud-init или установить данный пакет вручную. Версия для Windows доступна по ссылке https://cloudbase.it/cloudbase-init/.

Установка гостевых инструментов

Вместе с ПК Р-Виртуализация поставляются гостевые инструменты для виртуальных машин Linux и Windows. Они позволяют:

- Получать и настраивать параметры сети для виртуальных машин.
- Устанавливать пароли для пользователей в виртуальных машинах с помощью команды `prlctl set --userpasswd`. Если пользователя не существует, он будет создан.
- Выполнять команды в виртуальных машинах с помощью команды `prlctl exec`.

Установка гостевых инструментов также включает еженедельную автоматическую “зачистку” файловых систем в гостевых ОС Linux, выполняемую службой `fstrim`. Она высвобождает неиспользуемое пространство носителя путем удаления блоков данных, неиспользуемых файловой системой виртуальной машины. Если вы отключите службу, она не будет включена заново при обновлении гостевых инструментов. В гостевых ОС Windows по умолчанию подобную работу выполняет Storage Optimizer.

Примечание: Если сервер включен в кластер ПК Р-Хранилище, обеспечивающий избыточность данных при помощи репликации, то автоматическое сжатие (зачистка) файловой системы (или файловых систем), которая хранит реплики данных, отключено. В случае, когда избыточность данных обеспечивается с помощью избыточного кодирования, зачистка включена.

Рекомендуется установить cloud-init в гостевую ОС (см. раздел выше). В этом случае вам потребуется выполнить следующие действия:

1 Подмонтируйте образ гостевых инструментов, поставляемый вместе с ПК Р-Виртуализация, к оптическому диску виртуальной машины. Например:

```
# prlctl installtools MyVM
```

2 Запустите виртуальную машину:

```
# prlctl start MyVM
```

Cloud-init автоматически установит гостевые инструменты.

Если cloud-init не установлен в гостевой ОС виртуальной машины, можно установить гостевые инструменты автоматически или вручную.

Чтобы установить гостевые инструменты автоматически без cloud-init, выполните следующие действия:

1 Убедитесь, что выполняются следующие условия:

- На сервере установлен пакет `vz-guest-tools-updater`.
- В файле `/etc/vz/tools-update.conf` параметру `InstallTools` задано значение `true` (поведение по умолчанию).
- У виртуальной машины параметр `--tools-autoupdate` имеет значение `on` (поведение по умолчанию).

2 Остановите виртуальную машину перед установкой гостевых инструментов:

```
# prlctl stop MyVM
```

3 Запустите `vz-guest-tools-updater` для виртуальной машины:

```
# vz-guest-tools-updater MyVM
```

4 Запустите виртуальную машину:

```
# prlctl start MyVM
```

После запуска виртуальной машины инструмент `vz-guest-tools-updater` начнет установку гостевых инструментов, которая может занять несколько минут.

Важно: В процессе обновления образ гостевых инструментов принудительно монтируется к оптическому дисководу виртуальной машины, даже если он в это время используется.

Чтобы установить гостевые инструменты вручную без cloud-init, выполните следующие действия:

1 Подмонтируйте образ гостевых инструментов, поставляемый вместе с ПК Р-Виртуализация, к оптическому диску виртуальной машины. Например:

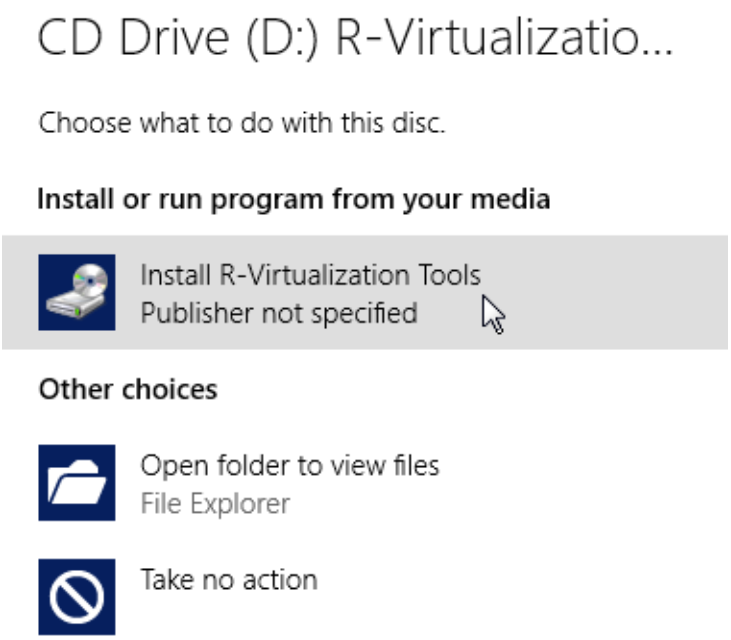
```
# prlctl installtools MyVM
```

2 Войдите в виртуальную машину и выполните следующие действия:

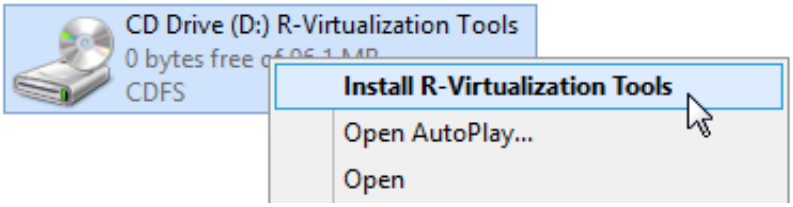
- В виртуальной машине Linux создайте точку монтирования для оптического диска с образом гостевых инструментов и запустите установщик:

```
# mount /dev/cdrom /mnt/cdrom
# bash /mnt/cdrom/install
```

- В виртуальной машине Windows, если включена программа автозапуска, запустите установщик в диалоговом окне AutoPlay.



В противном случае, щелкните правой кнопкой мыши по оптическому диску и выберите опцию **Install R-Virtualization Tools**.



Примечания:

1. Гостевые инструменты зависят от гостевого агента QEMU, который устанавливается вместе с инструментами. Для работы инструментов должен быть запущен демон/служба агента (`qemu-ga`).

2. Если гостевые инструменты несовместимы с каким-либо программным обеспечением виртуальной машины, вы можете удалить их (для получения подробной информации см. **Удаление гостевых инструментов из виртуальных машин** (стр. 169)).

Настройка сети

Чтобы виртуальные машины и контейнеры были доступны по сети, нужно назначить им IP-адреса и настроить DNS-серверы. Пример ниже показывает настройку данных параметров для виртуальной машины `MyVM` и контейнера `MyCT`:

- Назначение IPv4- и IPv6-адресов:

```
# prlctl set MyVM --device-set net0 --ipadd 10.0.186.100/24
# prlctl set MyVM --device-set net0 --ipadd 1fe80::20c:29ff:fe01:fb07
# prlctl set MyCT --ipadd 10.0.186.MyCT/24
# prlctl set MyCT --ipadd fe80::20c:29ff:fe01:fb08
```

  `net0` обозначает сетевую карту в виртуальной машине, которой назначается IP-адрес. Посмотреть полный список сетевых карт можно с помощью команды `prlctl list VM_name -i`.

- Назначение адресов DNS-серверов:

```
# prlctl set MyVM --nameserver 192.168.1.165
# prlctl set MyCT --nameserver 192.168.1.165
```

Примечания:

1. Настроить сеть можно только для виртуальных машин, в которых установлены гостевые инструменты.

2. Чтобы настроить маски сети для контейнеров, работающих в режиме `venet0`, необходимо убедиться, что параметр `USE_VENET_MASK` в конфигурационном файле `/etc/vz/vz.conf` имеет значение “`yes`”.

Создание учетных записей пользователей виртуальных сред

Параметр `--userpasswd` команды `prlctl set` позволяет создавать новые учетные записи пользователей в виртуальных машинах и контейнерах непосредственно с сервера. Созданную учетную запись можно использовать для входа в виртуальную машину или контейнер. Например:

```
# prlctl set MyVM --userpasswd user1:2wsx123qwe
```

Данная команда создает учетную запись `user1` в виртуальной машине `MyVM` и задает для нее пароль `2wsx123qwe`. Далее можно войти в виртуальную машину `MyVM` под учетной записью `user1` и управлять ей как автономным сервером: устанавливать дополнительное программное обеспечение, добавлять пользователей, настраивать службы и т.д.

Команду `prlctl set` можно также использовать для смены паролей существующих учетных записей в виртуальных машинах и контейнерах. Например, чтобы изменить пароль для `user1` в виртуальной машине `MyVM` на `0pi65jh9`, можно выполнить команду:

```
# prlctl set MyVM --userpasswd user1:0pi65jh9
```

Примечания:

1. Управлять учетными записями можно только в виртуальных машинах, в которых установлены гостевые инструменты.

2. Пароли должны удовлетворять минимальным требованиям к длине и сложности для соответствующих операционных систем. Например, Windows Server 2008 требует, чтобы пароль состоял из не менее шести символов и в нем должны присутствовать символы трех категорий из числа следующих: прописные буквы, строчные буквы, цифры и неалфавитные символы.

3. Не рекомендуется создавать учетные записи без пароля для виртуальных машин и контейнеров с операционными системами Linux.

Настройка параметров запуска

Команда `prlctl set` позволяет задать параметр запуска `onboot` для виртуальных машин и контейнеров. Если данному параметру задать значение `yes`, виртуальная машина или контейнер будет автоматически запускаться при загрузке физического сервера. Например, чтобы включить автоматический запуск для контейнера `MyCT` и виртуальной машины `MyVM` при загрузке сервера, можно выполнить следующие команды:

```
# prlctl set MyCT --onboot yes
# prlctl set MyVM --onboot yes
```

Следует иметь в виду, что параметр `onboot` будет применяться только при следующей загрузке сервера.

Запуск, остановка, перезапуск и запрос статуса виртуальных сред

Созданными виртуальными машинами и контейнерами можно управлять как обычным компьютером.

Запуск виртуальных сред

Запустить виртуальные машины и контейнеры можно командой `prlctl start`. Например:

```
# prlctl start MyCT
# prlctl start MyVM
```

Остановка виртуальных сред

Остановить виртуальные машины и контейнеры можно командой `prlctl stop`. Например:

```
# prlctl stop MyCT
# prlctl stop MyVM
```

Перезапуск виртуальных сред

Перезапустить виртуальные машины и контейнеры можно командой `prlctl restart`. Например:

```
# prlctl restart MyCT
# prlctl restart MyVM
```

Примечание: Перезапускать виртуальные машины можно, только если в них установлена оперативная система и гостевые инструменты.

Проверка статуса виртуальных сред

Проверить статус контейнера или виртуальной машины можно с помощью команды `prlctl status`. Например:

```
# prlctl status MyCT
CT MyCT exists running
# prlctl status MyVM
Vm MyVM exists stopped
```

Просмотр списка виртуальных сред

Чтобы посмотреть список всех виртуальных машин и контейнеров, существующих на физическом сервере, и получить дополнительную информацию о них (IP-адреса, имена хостов, текущее потребление ресурсов и т.д.) используйте команду `prlctl list`. Для вывода списка виртуальных машин и контейнеров в общем виде можно выполнить следующую команду:

```
# prlctl list -a
UUID                                    STATUS   IP_ADDR       T  NAME
{600adc12-0e39-41b3-bf05-c59b7d26dd73} running  10.10.1.101   CT MyCT
{b2de86d9-6539-4ccc-9120-928b33ed31b9} stopped  10.10.100.1   VM MyVM
```

Параметр `-a` команды `prlctl list` указывается, чтобы получить список как запущенных, так и остановленных виртуальных сред. По умолчанию отображаются только запущенные виртуальные машины и контейнеры. В колонках отображается следующая информация: идентификаторы, количество запущенных процессов, статус, IP-адреса и имена хостов виртуальных сред. Вывод можно настраивать по желанию с помощью параметра `-o`. Например:

```
# prlctl list -a -o name,ctid
NAME                        UUID
MyCT                        {26bc47f6-353f-444b-bc35-b634a88dbbcc}
MyVM                        {b8cb6d99-1af1-453d-a302-2fddd8f86769}
```

Примечание: Для просмотра списка всех колонок можно ввести команду `prlctl list -L`.

Клонирование виртуальных сред

ПК Р-Виртуализация позволяет создавать копию (клон) определенной виртуальной машины или контейнера, которая будет иметь те же данные и параметры ресурсов, что и оригинал. Клонирование может сэкономить время, так как для клонов не требуется изменять конфигурацию по сравнению с созданием новых виртуальных сред.

Клонировать можно остановленные виртуальные машины, а также остановленные и запущенные контейнеры. Например:

```
# prlctl clone MyCT --name MyCT_clone
# prlctl clone MyVM --name MyVM_clone
```

С помощью параметра `--name` указывается имя для клона.

При клонировании виртуальных машин Windows рекомендуется изменить их идентификаторы безопасности (SID), используя параметр `--changesid`.

Успешно клонированные виртуальные машины и контейнеры появятся в списке виртуальных сред на сервере. Например:

```
# prlctl list -a
UUID              STATUS    IP_ADDR         T    NAME
{62951c2a-...}    stopped   10.30.10.101    CT   MyCT
{49b66605-...}    stopped   10.30.10.101    CT   MyCT_clone
{7f4904ad-...}    stopped   10.30.128.115   VM   MyVM
{2afb2aa2-...}    stopped   10.30.128.134   VM   MyVM_clone
```

В примере выше видно, что клонированный контейнер имеет тот же IP-адрес, что и оригинальный контейнер. Перед использованием клонов следует убедиться, что им назначены разные IP-адреса (для получения инструкций о назначении IP-адресов виртуальным машинам и контейнерам см. **Настройка сети** (стр. 27)).

Создание связанных клонов

В ПК Р-Виртуализация можно создавать связанные клоны виртуальных машин. Связанный клон представляет собой копию виртуальной машины, которая имеет общие виртуальные диски с исходной виртуальной машиной. Создание связанных клонов занимает меньше времени и дискового пространства, так как они хранят только изменения исходных дисков, а не копируют их полностью. Связанный клон невозможно запустить без доступа к исходной машине, поэтому следует убедиться, что исходная виртуальная машина доступна и ее диски не повреждены.

Чтобы создать связанный клон, добавьте опцию `--linked` для команды `prlctl clone`. Например:

```
# prlctl clone MyVM --name MyVM_linked_clone --linked
```

На сервере связанные клоны отображаются как обычные виртуальные машины. Например:

```
# prlctl list -a
UUID             STATUS       IP_ADDR          T  NAME
{7f4904ad-...}   stopped      10.30.128.115    VM MyVM
{2e9862f6-...}   stopped      10.30.128.135    VM MyVM_linked_clone
```

Примечание: Для связанных клонов не поддерживаются операции миграции, резервного копирования, восстановления из резервной копии и отмены связывания.

Настройка стандартных директорий

При клонировании виртуальной машины или контейнера можно изменить следующие стандартные директории:

- `/vz/vmprivate/`<*dest_UUID*>, в которой будут храниться файлы клонированной виртуальной машины (где <*dest_UUID*> обозначает UUID полученной виртуальной машины). Чтобы изменить стандартную директорию для хранения файлов виртуальной машины `MyVM_clone`, нужно ввести следующую команду:

```
# prlctl clone MyVM --name MyVM_clone --dst /customVMs
```

В данном случае все файлы клонированной виртуальной машины будут помещены в директорию `/customVMs`. Следует иметь в виду, что указанная директория должна быть на сервере; в противном случае, команда не будет выполнена.

- `/vz/private/`<*dest_UUID*>, в которой будет находиться собственная область клонированного контейнера (где <*dest_UUID*> обозначает UUID полученного контейнера). Чтобы изменить стандартную директорию для собственной области контейнера `MyCT_clone`, нужно ввести следующую команду:

```
# prlctl clone MyCT1 --name MyCT_clone --dst /vz/private/customCTs
```

Примечание: Стандартные директории `/vz/vmprivate` и `/vz/private` действительны для серверов, не включенных в кластеры ПК Р-Хранилище.

Приостановка виртуальных сред

ПК Р-Виртуализация позволяет приостановить запущенную виртуальную машину или контейнер на физическом сервере путем сохранения их текущего состояния в специальный файл. Позже можно возобновить их работу из того же состояния, в котором они находились во время приостановки. Приостановка виртуальной машины или контейнера может быть удобна, например, если нужно перезагрузить физический сервер, но нежелательно:

- закрывать приложения, запущенные в данный момент в виртуальной машине или контейнере,
- тратить лишнее время на завершение работы гостевой операционной системы, а затем на ее включение.

Для сохранения текущего состояния виртуальной среды используется команда `prlctl suspend`. Например, для приостановки контейнера `MyCT`:

```
# prlctl suspend MyCT
```

В любое время можно возобновить работу контейнера `MyCT` с помощью команды:

```
# prlctl resume MyCT
```

Когда процесс восстановления будет завершен, все приложения, которые были запущены в контейнере `MyCT` во время его приостановки, будут запущены снова.

Выполнение команд в виртуальных средах

ПК Р-Виртуализация позволяет выполнять команды в виртуальных машинах и контейнерах с физического сервера, т.е. без входа в соответствующую виртуальную среду. Например, это удобно в следующих случаях:

- Если неизвестны учетные данные для виртуальной машины или контейнера, но необходимо выполнить некоторые диагностические команды, чтобы проверить работоспособность виртуальной среды.
- Если нет доступа к виртуальной машине или контейнеру по сети.

В обоих случаях можно использовать команду `prlctl exec` для выполнения команды в соответствующей виртуальной среде. По умолчанию выполнение `prlctl exec` <*command*> соответствует выполнению `bash -c` <*command*> в виртуальной машине и контейнере Linux или `cmd /c` <*command*> в виртуальной машине Windows. Использование параметра `--without-shell` позволяет выполнять команды напрямую без командной оболочки.

Пример ниже изображает запуск остановленного SSH-демона в виртуальной машине `My_Linux`:

```
# prlctl exec My_Linux /etc/init.d/sshd status
openssh-daemon is stopped
# prlctl exec My_Linux /etc/init.d/sshd start
Starting sshd: [ OK ]
# prlctl exec My_Linux /etc/init.d/sshd status
openssh-daemon (pid 26187) is running...
```

Примечания:

1. Команду `prlctl exec` можно использовать только в виртуальных машинах с установленными гостевыми инструментами.

2. Команда `prlctl exec` выполняется в виртуальной машине или контейнере из директории `/`, а не из директории `/root`.

Удаление виртуальных сред

Удалить ненужную виртуальную машину или контейнер можно с помощью команды `prlctl delete`. Следует заметить, что невозможно удалить запущенную или смонтированную виртуальную среду. В примере ниже показано удаление запущенного контейнера `MyCT`:

```
# prlctl delete MyCT
Removing the CT...
Failed to remove the CT: Unable to complete the operation. This operation cannot be
completed because the virtual machine "{4f27f27f-c056-4a65-abf6-27642b6edd21}"is in the
"running" state.
# prlctl stop MyCT
Stopping the CT...
The CT has been successfully stopped.
# prlctl delete MyCT
Removing the CT...
The CT has been successfully removed.
```

Просмотр подробной информации о виртуальных средах

Для просмотра подробной информации о виртуальной машине или контейнере используется команда `prlctl list -i`. Например, следующая команда показывает всю информацию о виртуальной машине `MyVM`.

```
# prlctl list -i MyVM
```

В таблице ниже описаны основные параметры, отображаемые командой `prlctl list -i`.

Параметр	Описание
`ID`	Идентификатор виртуальной машины. Обычно при выполнении операции с виртуальной машиной используется ее идентификатор или имя.
`EnvID`	Идентификатор ядра виртуальной машины.
`Name`	Имя виртуальной машины.
`Description`	Описание виртуальной машины.
`State`	Состояние виртуальной машины.
`OS`	Гостевая операционная система, установленная в виртуальной машине.
`Uptime`	Время, прошедшее с последнего сброса счетчика и показывающее, как долго работает виртуальная машина. **Примечание:** Счетчик времени работы, как и счетчик даты и времени запуска можно сбросить с помощью команды `prlctl reset-uptime`.
`Home`	Директория, в которой хранятся файлы виртуальной машины.
`Guest tools`	Указывает, установлены ли в виртуальной машине гостевые инструменты.
`Autostart`	Указывает, автоматически ли запускается виртуальная машина при включении физического сервера.
`Boot order`	Порядок проверки загрузочных устройств при запуске виртуальной машины.
`Hardware`	Аппаратное оборудование виртуальной машины.
`Offline management`	Указывает, включена ли функция автономного управления и, если включена, отображает список доступных автономных служб.

Примечание: Параметры контейнеров, отображаемые командой `prlctl list`, идентичны параметрам виртуальных машин.

Управление резервными копиями виртуальных сред

Регулярное резервное копирование виртуальных машин и контейнеров необходимо для надежности системы. ПК Р-Виртуализация позволяет создавать резервные копии и

восстанавливать из них виртуальные машины и контейнеры на локальном сервере с помощью утилит `prlctl` и `prlsrvctl`.

Создание резервных копий для виртуальных сред

Для резервного копирования виртуальных машин и контейнеров можно использовать команду `prlctl backup`. Полученные резервные копии могут храниться как на локальном, так и на удаленном сервере (например, на выделенном сервере резервирования данных).

По умолчанию создается частичная резервная копия, содержащая только файлы, которые были изменены с последнего полного или частичного резервного копирования. Если для виртуальной среды выполняется резервное копирование в первый раз, то создается полная резервная копия. (Также можно создать полную резервную копию для виртуальной среды с помощью параметра `-f`).

Примечание: Для обеспечения повышенной безопасности в процессе резервного копирования ПК Р-Виртуализация использует туннелирование соединения между локальным сервером и удаленным сервером резервирования данных. Туннелирование увеличивает время резервного копирования, поэтому, если вы хотите ускорить процесс и не нуждаетесь в безопасном канале между серверами, то можете отключить туннелирование с помощью параметра `--no-tunnel`.

Для резервного копирования виртуальной машины `MyVM` введите следующую команду:

```
# prlctl backup MyVM
...
The VM has been successfully backed up with backup id {746dba2a-3b10-4ced-9dd6-
76a2blcl4a69}
```

ID резервной копии, который показан в примере выше, используется для управления резервной копией.

Если вы хотите создать резервную копию виртуальной машины `MyVM` и хранить ее на удаленном сервере, необходимо указать IP-адрес или имя хоста удаленного сервера при помощи параметра `-s`, например:

```
# prlctl backup MyVM -s 192.168.0.10
```

По умолчанию для входа на удаленный сервер используется учетная запись root, поэтому вам необходимо будет указать от нее пароль, когда это потребуется. Также можно указать учетные данные (и порт) в формате `[<user>[:<passwd>]@]<server>[:<port>]]`.

По умолчанию все новые резервные копии помещаются в директорию `/vz/vmprivate/backups`. Для изменения стандартной директории резервных копий можно использовать команду `prlsrvctl set --backup-path` <*path*>.

Примечания:

1. Резервные копии можно создавать как для остановленных, так и для запущенных виртуальных машин и контейнеров.

2. Для согласованного резервного копирования запущенной виртуальной машины необходимо, чтобы в ней были установлены гостевые инструменты.

3. Невозможно создать резервные копии для виртуальных машин с подсоединенными физическими жесткими дисками, смонтированными ISO-образами или образами флоппи-дисков и т.д.

Просмотр списка резервных копий виртуальных сред

Посмотреть список резервных копий на сервере можно с помощью команды `prlctl backup-list`. Например:

```
# prlctl backup-list
    ID   Backup_ID      Node      Date                Type   Size
{c1dee22f...}  {209d54a0...}  test.com  2011-05-30 10:19:32 f      411566405
[The ID and Backup ID are reduced for better readability.]
```

Из вывода команды видно, что в данный момент на сервере существует только одна резервная копия с ID `209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca`. Информация о резервной копии представлена в виде таблицы со следующими колонками:

Колонка	Описание
`ID`	Уникальный ID виртуальной машины или контейнера.
Backup_ID	ID, назначенный для резервной копии. Данный ID необходимо указывать при выполнении операций, связанных с резервным копированием.
Node	Имя хоста физического сервера, на котором хранится резервная копия.
Date	Дата и время создания резервной копии.
Type	Тип резервной копии. На текущий момент доступно два типа резервного копирования: • полная, `f`, • частичная, `i`, содержащая только файлы, которые были изменены с последнего полного или частичного резервного копирования. Данный тип резервного копирования используется по умолчанию.
Size	Размер резервной копии, в байтах.

Восстановление виртуальных сред из резервных копий

Локальные и удаленные резервные копии виртуальных машин и контейнеров можно восстановить с помощью команды `prlctl restore`.

Примечание: Для обеспечения повышенной безопасности в процессе резервного копирования ПК Р-Виртуализация использует туннелирование соединения между локальным сервером и удаленным сервером резервирования данных. Туннелирование увеличивает время резервного копирования, поэтому, если вы хотите ускорить процесс и не нуждаетесь в безопасном канале между серверами, то можете отключить туннелирование с помощью параметра `--no-tunnel`.

При восстановлении применяются следующие правила и положения:

- Команды восстановления должны выполняться на целевом сервере (на который будут восстановлены виртуальные среды).
- Восстанавливать из резервной копии можно только остановленные виртуальные среды.
- Из резервных копий старой версии ПК Р-Виртуализация можно восстановить виртуальные среды на серверы с новой версией ПК Р-Виртуализация (с конвертацией в формат новой версии Р-Виртуализация):
  - Восстановление из резервных копий виртуальных сред с гостевыми ОС, не поддерживаемыми в новой версии ПК Р-Виртуализация, может быть выполнено некорректно (см. **Поддерживаемые гостевые операционные системы** (стр. 23)).
  - Перед восстановлением на сервер с новой версией ПК Р-Виртуализация контейнеры, созданные на базе VZFS, должны быть конвертированы в формат ploop, и для них должна быть заново создана резервная копия.

Примечание: Если конвертация восстановленных виртуальных машин не удается, то восстановленная виртуальная машина удаляется с целевого сервера и можно повторить попытку. Если вторая попытка также оказывается неуспешной, необходимо включить режим отладки для устаревших виртуальных машин на целевом сервере с новой версией ПК Р-Виртуализация (см. **Включение режима отладки для устаревших виртуальных машин** (стр. 171), предпринять еще одну попытку восстановления, отправить отчет об ошибке и обратиться в службу технической поддержки. В режиме отладки мигрированная виртуальная машина не будет удалена с целевого сервера с новой версией ПК Р-Виртуализация после неудачной конвертации. Она будет находиться на нем в остановленном или запущенном состоянии с отключенной сетью, чтобы команда технической поддержки имела возможность изучить дамп памяти и найти причину проблемы.

Чтобы восстановить из резервной копии виртуальную машину `MyVM` с UUID `a53f1184-333e-41cf-b410-2ec8ffea67d4`, нужно ввести команду:

```
# prlctl restore MyVM
```

или

```
# prlctl restore a53f1184-333e-41cf-b410-2ec8ffea67d4
```

Если виртуальная машина имеет несколько резервных копий, то она восстанавливается из последней копии. Чтобы восстановить из конкретной резервной копии, необходимо указать ее ID с помощью параметра `-t`. Например:

```
# prlctl restore -t 24a3011c-8667-4870-9e11-278f1398eab0
```

Если резервные копии хранятся на удаленном сервере и виртуальная среда, которую нужно восстановить, не существует на целевом сервере, то можно восстановить ее, указав UUID виртуальной среды или ID резервной копии параметром `-t` и IP-адрес или имя хоста удаленного сервера параметром `-s`. Например:

```
# prlctl restore -t 24a3011c-8667-4870-9e11-278f1398eab0 -s 192.168.0.10
```

Если виртуальная среда существует на целевом сервере (например, уже была однажды восстановлена из удаленной резервной копии), то можно также восстановить ее из последней резервной копии, указав имя виртуальной среды.

При необходимости можно перенести удаленные резервные копии на локальный сервер и выполнить восстановление из них локально. Для этого необходимо выполнить следующие действия:

1. Узнать стандартные директории резервных копий на исходном и целевом серверах, введя команду `prlsrvctl info | grep "Backup path"` на каждом из них.
2. Копировать резервные копии в стандартную директорию на целевом сервере. Если резервные копии находятся в сетевом хранилище, можно подсоединить сетевое хранилище к стандартной директории резервных копий на целевом сервере.
3. Восстановить из резервной копии с помощью команды `prlctl restore –t`, как показано в примере выше.

Удаление резервных копий виртуальных сред

Удалить резервные копии можно с помощью команды `prlctl backup-delete`.

Чтобы удалить определенную резервную копию виртуальной машины или контейнера, нужно указать имя или UUID виртуальной среды, а также ID резервной копии (ID резервных копий можно узнать с помощью команды `prlctl backup-list`). Например:

```
# prlctl backup-list
    ID   Backup_ID       Node      Date                Type   Size
{c1dee22f...}  {209d54a0...}  test.com  2011-05-30 10:19:32 f      411566405
[The ID and Backup ID are reduced for better readability.]
# prlctl backup-delete MyVM -t 209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca --keep-chain
```

Параметр `--keep-chain` используется для сохранения цепочки резервных копий после удаления из нее отдельных резервных копий.

Для удаления всех резервных копий виртуальной машины или контейнера необходимо указать только имя или UUID виртуальной среды:

```
# prlctl backup-delete MyVM
```

Резервное копирование целых серверов

В дополнение к резервному копированию отдельных виртуальных сред можно выполнить резервное копирование всех виртуальных сред на сервере с помощью команды `prlsrvctl backup`. Например:

```
# prlsrvctl backup -f
Backing up the CT MyCT
...
The CT has been successfully backed up with backup id {b14ec76d-c0e2-432f-859a-
4727c0042065}
Backing up the VM MyVM
...
The VM has been successfully backed up with backup id {746dba2a-3b10-4ced-9dd6-
76a2blcl4a69}.
```

Примечания:

1. Резервные копии можно создавать как для остановленных, так и для запущенных виртуальных машин и контейнеров.

2. Для согласованного резервного копирования запущенной виртуальной машины необходимо, чтобы в ней были установлены гостевые инструменты.

3. Невозможно создать резервные копии для виртуальных машин с подсоединенными физическими жесткими дисками, смонтированными ISO-образами или образами флоппи-дисков и т.д.

Подсоединение резервных копий к виртуальным средам

Для чтения содержимого резервной копии виртуальной машины или контейнера ее можно подсоединить к виртуальной среде в качестве виртуального жесткого диска.

Примечания:

1. Подсоединять можно только локальные резервные копии.

2. Подсоединенная резервная копия доступна для записи, чтобы файловая система могла обработать ее журнал при монтировании. Однако все изменения будут отменены при отсоединении резервной копии. Размер данных, который можно записать в подсоединенную резервную копию, ограничен до 256 МБ.

3. Подсоединенные резервные копии не сохраняются в процессе операций клонирования, резервного копирования и создания снапшотов.

Подсоединение резервных копий к виртуальным машинам Linux

1. Необходимо убедиться, что в виртуальной машине, к которой будет подсоединена резервная копия, установлены утилиты `prl_backup` и `kpartx`. Утилита `prl_backup` устанавливается вместе с гостевыми инструментами.
2. Узнайте ID и имя файла резервной копии, которую нужно подсоединить, с помощью команды `prlctl backup-list`. Например:

```
# prlctl backup-list vm2 -f
...
Backup_ID: {0fcd6696-c9dc-4827-9fbd-6ee3abe017fa}
...
Name: harddisk.hdd.qcow2c
```

3. Подсоедините резервную копию в качестве жесткого диска к нужной виртуальной машине Linux, используя команду `prlctl set --backup-add`. Например:

```
# prlctl set vm1 --backup-add {0fcd6696-c9dc-4827-9fbd-6ee3abe017fa} --disk
harddisk.hdd.qcow2c
Creating hdd1 (+) sata:2 real='backup:///{0fcd6696-c9dc-4827-9fbd-6ee3abe017fa}/ \
harddisk.hdd.qcow2c' backup='{0fcd6696-c9dc-4827-9fbd-6ee3abe017fa}'
disk='harddisk.hdd.qcow2c'
```

   Если резервная копия содержит несколько дисков и необходимо подсоединить их все, то параметр `--disk` не указывается.

4 Узнайте имя подключенного устройства, которое в данный момент находится в нерабочем состоянии, с помощью команды `prl_backup list`. Например:

```
# prlctl exec vm1 prl_backup list
...
List of disabled attached backups:
[1] /dev/sdc
```

5 Включите резервную копию, используя команду `prl_backup enable`. Например:

```
# prlctl exec vm1 prl_backup enable /dev/sdc
```

6 Дополнительно можно проверить, что резервная копия включена с помощью команды `prl_backup list -e`. Например:

```
# prlctl exec vm1 prl_backup list -e
List of enabled attached backups:
[1] /dev/sdc (/dev/mapper/backup1)
NAME                TYPE  SIZE   FSTYPE   UUID
MOUNTPOINT
backup1 (dm-3)      dm    64G
|-backup1p1 (dm-4)  part  500M   ext4     1ac82165-113d-40ee-8ae2-8a72f62d95bf
`-backup1p2 (dm-5)  part  63.5G  LVM2_mem Zw9QiY-BiU5-o8dn-ScTK-vOZx-KujW-wbgmS3
```

Теперь можно смонтировать нужную часть резервной копии как файловую систему.

Монтирование части `ext4` не требует дополнительных шагов. Например:

```
# prlctl exec vm1 mount /dev/mapper/backup1p1 /mnt/backup1p1
```

После выполнения данной команды содержимое части резервной копии доступно в `/mnt/backup1p1`.

Монтирование части `LVM2_member` включает следующие шаги:

1 Назначьте новой группе томов новое имя с помощью команды `vgimportclone`. Например:

```
# prlctl exec vm1 vgimportclone -n backup1p2 /dev/mapper/backup1p2
...
Volume group "VolGroup" successfully renamed to "backup1p2"
...
Found volume group "backup1p2" using metadata type lvm2
...
```

2 Выведите монтируемые логические тома в виде списка, используя команду `lvs`. Например:

```
# prlctl exec vm1 lvs | grep backup1p2
lv_home backup1p2 -wi-------11.54g
lv_root backup1p2 -wi------- 50.00g
lv_swap backup1p2 -wi-------1.97g
```

3 Активируйте нужный логический том командой `lvchange -ay`. Например:

```
# prlctl exec vm1 lvchange -ay /dev/backup1p2/lv_root
```

4 Смонтируйте логический том как файловую систему. Например:

```
# prlctl exec vm1 mount /dev/backup1p2/lv_root /mnt/backup1p2
```

После выполнения данных шагов содержимое части резервной копии доступно в `/mnt/backup1p2`.

Подсоединение резервных копий к виртуальным машинам Windows

1 Узнайте ID и имя файла резервной копии. Например:

```
# prlctl backup-list vm2 -f
...
Backup_ID: {cff742a9-f942-41c5-9ac2-ace3b4eba783}
...
Name: harddisk.hdd.qcow2c
```

2 Подсоедините резервную копию в качестве жесткого диска к нужной виртуальной машине Windows. Например:

```
# prlctl set vm1 --backup-add {cff742a9-f942-41c5-9ac2-ace3b4eba783} --disk
harddisk.hdd.qcow2c
Creating hdd1 (+) sata:2 real='backup:///{cff742a9-f942-41c5-9ac2-ace3b4eba783}/ \
harddisk.hdd.qcow2c' backup='{cff742a9-f942-41c5-9ac2-ace3b4eba783}'
disk='harddisk.hdd.qcow2c'
```

Подсоединенная резервная копия появится в виртуальной машине Windows как готовый к использованию диск.

Подсоединение резервных копий к контейнерам Linux

1 Узнайте ID и имя файла резервной копии с помощью команды `prlctl backup-list -f`. Например:

```
# prlctl backup-list 102 -f
...
Backup_ID: {d70441dd-f077-44a0-8191-27704d4d8fdb}
...
Name: root.hdd.qcow2c
...
```

2 Подсоедините резервную копию в качестве жесткого диска к нужному контейнеру Linux, используя команду `prlctl set --backup-add`. Например:

```
# prlctl set MyCT --backup-add {d70441dd-f077-44a0-8191-27704d4d8fdb} --disk
root.hdd.qcow2c
Creating hdd1 (+) sata:0 real='backup:///{d70441dd-f077-44a0-8191-27704d4d8fdb}/ \
root.hdd.qcow2c' backup='{d70441dd-f077-44a0-8191-27704d4d8fdb}' disk='root.hdd.qcow2c'
```

3 С помощью ID резервной копии идентифицируйте устройство ploop, соответствующее резервной копии. Например:

```
# ploop list | grep {d70441dd-f077-44a0-8191-27704d4d8fdb}
ploop28261 /buse/{8417a267-0919-4c8f-a31d-68671358d6a8}_{d70441dd-f077-44a0-8191-
27704d4d8fdb}_root.hdd.qcow2c/content
```

4 Смонтируйте логический том как файловую систему. Например:

```
# prlctl exec MyCT mount /dev/ploop28261p1 /mnt/backup1
```

После выполнения данных шагов содержимое резервной копии доступно в `/mnt/backup1`.

Отсоединение резервных копий от виртуальных сред

Примечание: Перед отсоединением резервной копии от запущенной виртуальной машины необходимо выполнить следующие действия:

1. (Для виртуальных машин Linux) Отключить резервное устройство, выполнив команду `prl_backup disable` в гостевой ОС.

2. (Для виртуальных машин Linux и Windows) Отсоединить соответствующий виртуальный диск, выполнив команду `prlctl set --device-set hdd<N> --disconnect` на сервере.

- Чтобы отсоединить виртуальные диски от всех резервных копий, подсоединенных к виртуальной среде, используйте команду `prlctl set --backup-del all`. Например:

```
# prlctl set vm1 --backup-del all
```

- Чтобы отсоединить виртуальные диски от конкретной резервной копии, подсоединенной к виртуальной среде, введите команду `prlctl set --backup-del` <*backup_ID*>. Например:

```
# prlctl set vm1 --backup-del {e13561bb-5676-49bd-a935-ae0145eb0229}
```

- Чтобы отсоединить определенный виртуальный диск от резервной копии, подсоединенной к виртуальной среде, удалите данный диск с помощью команды `prlctl set --device-del hdd<N>`. Например:

```
# prlctl set vm1 --device-del hdd1
```

Управление шаблонами

В ПК Р-Виртуализация шаблон представляет собой предварительно настроенную виртуальную среду, которую можно просто и быстро развернуть в полностью функционирующую виртуальную среду. Как и любая обычная виртуальная среда, шаблон содержит аппаратные ресурсы (виртуальные диски, периферийные устройства) и операционную систему. В нем также может быть установлено программное обеспечение. Фактически единственным основным отличием шаблона от виртуальной среды является то, что шаблон нельзя запустить.

С шаблонами можно выполнять следующие операции:

- создать новый шаблон,
- вывести список существующих шаблонов,
- создать виртуальную машину или контейнер по шаблону,
- мигрировать шаблоны между серверами ПК Р-Виртуализация (см. **Миграция шаблонов виртуальных сред** (стр. 52)).

Данные операции подробно описаны ниже.

Создание шаблонов

Шаблон можно создать с помощью команды `prlctl clone`. Шаблон может быть полезен для создания нескольких виртуальных сред с одинаковой конфигурацией. В данном случае требуется выполнить следующие шаги:

1 Создание виртуальной машины или контейнера с нужной конфигурацией.

2 Создание шаблона на базе созданной виртуальной среды.

3 Использование шаблона для создания любого числа виртуальных сред.

Для создания шаблона виртуальной машины MyVM можно выполнить команду:

```
# prlctl clone MyVM --name template1 --template
```

Данная команда клонирует виртуальную машину и сохраняет ее в виде шаблона `template1`. После создания шаблона его можно использовать для создания новых виртуальных машин.

Просмотр списка шаблонов

Иногда необходимо посмотреть список шаблонов, доступных на физическом сервере, например, если нужно создать виртуальную среду по какому-либо шаблону, но его точное имя неизвестно. В таком случае можно выполнить команду `prlctl list` для вывода полного списка шаблонов на физическом сервере:

```
# prlctl list -t
UUID                                    DIST         T   NAME
{017bfdf0-b546-4309-90d0-147ce55773f2}  centos       VM  centos_tmpl
{92cd331e-0572-46ac-8586-f19b8d029c4d}  centos       CT  ct201_tmp1
{fc40e38e-8da4-4b26-bb18-6098ec85f7b4}  Debian       VM  deb_tmpl
{0dea5912-b114-45a9-bd1a-dc065c1b8e9f}  Ubuntu       VM  ubuntu_tmp1
{479e66aa-332c-4e3e-975e-b8b6bfc9d2e0}  win-2012     VM  w12en_tmpl
```

В примере выше на сервере существует 5 шаблонов. Информация об этих шаблонах представлена в виде таблицы со следующими колонками (слева направо): ID шаблона, ОС шаблона, тип шаблона (контейнер или виртуальная машина) и имя шаблона.

Развертывание шаблонов

Чтобы развернуть виртуальную среду из шаблона, используйте параметр `--ostemplate` команды `prlctl create`. Например, чтобы развернуть виртуальную машину `MyVMtemplate1` из шаблона `template1`, можно выполнить следующую команду:

```
# prlctl create MyVMtemplate1 --ostemplate template1
```

Для проверки созданной виртуальной машины можно использовать команду `prlctl list -a`:

```
# prlctl list -a
STATUS       IP_ADDR         NAME
running      10.12.12.101    MyVM
stopped      10.12.12.34     MyVMtemplate1
```

Шаблон остается нетронутым и может быть использован для создания других виртуальных машин:

```
# prlctl list -t
{4ad11c28-9f0e-4086-84ea-9c0487644026} win-2008  template1
{64bd8fea-6047-45bb-a144-7d4bba49c849} rhel      template2
```

Хранение шаблонов в ПК Р-Хранилище

ПК Р-Виртуализация позволяет хранить шаблоны контейнеров и виртуальных машин в общих директориях в кластерах ПК Р-Хранилище. К данным шаблонам можно получить доступ с любого сервера в кластере.

Чтобы поместить шаблон в ПК Р-Хранилище, выполните следующие действия на том сервере кластера, где находится исходная виртуальная среда:

1 Создайте шаблон. Например:

```
# prlctl clone MyVM --name template1 --template
```

2 Переместите шаблон в директорию `vmtemplates` в ПК Р-Хранилище:

- для ПК Р-Хранилище с управлением при помощи командной строки полный путь директории - `/vstorage/<cluster_name>/vmtemplates`, например:

```
# prlctl move template1 --dst /vstorage/vstor1/vmtemplates
```

- для ПК Р-Хранилище с управлением при помощи графического интерфейса полный путь директории - `/mnt/vstorage/vmtemplates`, например:

```
# prlctl move template1 --dst /mnt/vstorage/vmtemplates
```

В течение пяти минут шаблон будет автоматически обнаружен утилитой `prlctl`. Доступность шаблона можно проверить выводом списка шаблонов на другом сервере ПК Р-Хранилище. Например:

```
# prlctl list -t | grep template1
{4ad11c28-9f0e-4086-84ea-9c0487644026} win-2008 template1
```

Когда шаблон стал доступен для всех серверов в ПК Р-Хранилище, можно приступить к созданию контейнеров или виртуальных машин на его основе на любом сервере кластера, как описано в разделе **Развертывание шаблонов** (стр. 43).

Управление снапшотами

В ПК Р-Виртуализация можно сохранить текущее состояние виртуальной машины или контейнера при помощи снапшотов. Затем можно продолжить работу в виртуальной среде и вернуться к сохраненному состоянию в любое время. Снапшоты могут быть полезны в следующих случаях:

- Настройка приложений с множеством параметров. Можно проверить работу настроек, перед тем как применять их к приложению. Поэтому перед началом экспериментов следует создать снапшот.
- Участие в крупных проектах по разработке. Можно отмечать этапы разработки созданием снапшота для каждого этапа. Если что-нибудь пойдет не так, можно будет легко вернуться к предыдущему этапу и продолжить разработку.

ПК Р-Виртуализация позволяет создавать, выводить в виде списка, удалять снапшоты и возвращаться к ним. Данные операции подробно описаны ниже.

Создание снапшотов

Для создания снапшота виртуальной машины или контейнера можно использовать команду `prlctl snapshot`.

Создание снапшотов виртуальных машин

Для создания снапшота виртуальной машины `MyVM` выполните следующую команду:

```
# prlctl snapshot MyVM
...
The snapshot with ID {12w32198-3e30-936e-a0bbc104bd20} has been successfully created.
```

Новый снапшот сохраняется в файл `/vz/vmprivate/`<*UUID*>`/Snapshots/`<*snapshot_ID*>`.pvs`, где <*UUID*> является UUID виртуальной машины и <*snapshot_ID*> - это уникальный ID снапшота. В примере выше снапшот с ID `{12w32198-3e30-936e-a0bbc104bd20}` сохраняется в файл `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/Snapshots/{12w32198-3e30-936e-a0bbc104bd20}.pvs`.

```
# ls /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/Snapshots/
{063615fa-f2a0-4c14-92d4-4c935df15840}.pvc
```

ID снапшотов нужны для переключения и удаления снапшотов.

При создании снапшота также можно указать его имя и добавить описание. Например:

```
# prlctl snapshot MyVM -n Clean_System -d "This snapshot was created right after
installing Windows XP."
```

Указанное имя и описание сохраняются в файле `/vz/vmprivate/`<*UUID*>`/Snapshots.xml`.

Создание снапшотов контейнеров

Для создания снапшота контейнера `MyCT` выполните следующую команду:

```
# prlctl snapshot MyCT
...
The snapshot with ID {08ddd014-7d57-4b19-9a82-15940f38e7f0} has been successfully
created.
```

Новый снапшот сохраняется в файл `/vz/private/`<*UUID*>`/dump/`<*snapshot_ID*>, где <*UUID*> является UUID контейнера и <*snapshot_ID*> - это уникальный ID снапшота. В примере выше снапшот с ID `{08ddd014-7d57-4b19-9a82-15940f38e7f0}` сохраняется в файл `/vz/private/26bc47f6-353f-444b-bc35-b634a88dbbcc/dump/{08ddd014-7d57-4b19-9a82-15940f38e7f0}`.

```
# ls /vz/private/26bc47f6-353f-444b-bc35-b634a88dbbcc/dump
{08ddd014-7d57-4b19-9a82-15940f38e7f0}
```

ID снапшотов нужны для переключения и удаления снапшотов.

При создании снапшота также можно указать его имя и добавить описание. Например:

```
# prlctl snapshot MyCT --n Clean_System --d "This snapshot was created right after
installing Windows XP."
```

Указанное имя и описание сохраняются в файле `/vz/private/<UUID>/Snapshots.xml`.

Ветвление снапшотов

Ветви снапшотов могут использоваться для тестирования или сравнения похожих конфигураций. Ветвь снапшотов создается при выполнении следующей последовательности действий:

1. Создание несколько снапшотов.
2. Возврат к одному из снапшотов.
3. Внесение изменения в виртуальную машину или контейнер.
4. Создание снапшота.

В этом случае созданный снапшот создаст новую ветвь на основе снапшота из **Шага 2**.

Ограничения и рекомендации

- Имена виртуальных машин и снапшотов, а также описания снапшотов, содержащие пробелы, должны быть заключены в кавычки (например, "Windows XP") при указании их в качестве аргументов для команды `prlctl`.
- Перед созданием снапшота рекомендуется завершить любые установки, загрузки и запись на внешние устройства. Также следует завершить или отменить транзакции, осуществляемые через виртуальную машину во внешние базы данных.
- Создание снапшотов не поддерживается для контейнеров с включенной функцией NFS-сервера.

Просмотр списка снапшотов

Для просмотра списка всех снапшотов конкретной виртуальной среды можно использовать команду `prlctl snapshot-list`. Например, чтобы проверить снапшоты виртуальной машины `MyVM`, выполните команду:

```
# prlctl snapshot-list MyVM
PARENT_SNAPSHOT_ID                          SNAPSHOT_ID
                                            {989f3415-3e30-4494-936e-a0bbc104bd20}
{989f3415-3e30-4494-936e-a0bbc104bd20}   *{063615fa-f2a0-4c14-92d4-4c935df15840}
```

Данная команда показывает, что виртуальная машина `MyVM` имеет два снапшота. Снапшот с ID `{063615fa-f2a0-4c14-92d4-4c935df15840}` основан на снапшоте с ID `{989f3415-3e30-4494-936e-a0bbc104bd20}`, т.е. первый снапшот является дочерним второго. Текущий снапшот помечается звездочкой (*).

Посмотреть отношение снапшотов друг к другу можно с помощью параметра `-t`:

```
# prlctl snapshot-list MyVM -t
_{989f3415-3e30-4494-936e-a0bbc104bd20}_{063615fa-f2a0-4c14-92d4-4c935df15840} \
*{712305b0-3742-4ecc-9ef1-9f1e345d0ab8}
```

Вывод команды показывает, что в текущий момент для виртуальной машины MyVM создано две ветви снапшотов. Снапшот с ID {989f3415-3e30-4494-936e-a0bbc104bd20} является основой данных ветвей.

Для получения подробной информации о снапшоте можно использовать параметр -i с ID снапшота:

```
# prlctl snapshot-list MyVM -i {063615fa-f2a0-4c14-92d4-4c935df15840}
ID: {063615fa-f2a0-4c14-92d4-4c935df15840}
Name: Clean_System
Date: 2012-07-22 22:39:06
Current: yes
State: poweroff
Description: <![CDATA[This snapshot was created right after installing Windows 7]]>
```

В выводе команды отображается следующая информация о снапшоте:

Поле	Описание
ID	ID, назначенный снапшоту.
Name	Имя, назначенное снапшоту.
Date	Дата и время создания снапшота.
Current	Обозначает, является ли данный снапшот текущим.
State	Состояние, в котором находилась виртуальная среда во время создания снапшота.
Description	Описание снапшота.

Возврат к снапшотам

Для возвращения к снапшоту можно использовать команду prlctl snapshot-switch. При возврате к снапшоту текущее состояние виртуальной среды сбрасывается, и все изменения системы, сделанные с последнего снапшота, отменяются. Поэтому перед возвратом к снапшоту рекомендуется сохранить текущее состояние, создав новый снапшот (см. **Создание снапшотов** (стр. 45)).

Для команды prlctl snapshot-switch необходимо указать имя виртуальной среды и ID снапшота, например:

```
# prlctl snapshot-switch MyVM --id {cedbc4eb-dee7-42e2-9674-89d1d7331a2d}
```

В примере выше выполняется возврат к снапшоту {cedbc4eb-dee7-42e2-9674-89d1d7331a2d} виртуальной машины MyVM.

Удаление снапшотов

Чтобы удалить ненужные снапшоты виртуальных сред, можно использовать команду prlctl snapshot-delete. Например:

```
# prlctl snapshot-delete MyVM --id {903c12ea-f6e6-437a-a2f0-a1d02eed4f7e}
```

При удалении родительского снапшота дочерние снапшоты не удаляются, и информация от первых добавляется к последним.

Миграция виртуальных сред

Для упрощения обновления аппаратного оборудования и распределения загрузки между несколькими серверами ПК Р-Виртуализация позволяет мигрировать виртуальные машины и контейнеры с помощью команды `prlctl migrate`.

Важно: Для осуществления миграции должно быть разрешено прямое SSH-соединение через порт 22 между исходным и целевым серверами.

Перед миграцией следует убедиться, что целевой сервер имеет:

- достаточно места на жестком диске для хранения мигрируемой виртуальной среды,
- достаточно памяти и мощности ЦП для запуска мигрируемой виртуальной среды,
- стабильное сетевое соединение с исходным сервером.

Примечание: Миграция виртуальных машин, имеющих снапшоты, не поддерживается (все существующие снапшоты должны быть удалены).

Виртуальные машины и контейнеры можно мигрировать как на удаленный сервер, так и с него. Например, чтобы мигрировать виртуальную машину на удаленный сервер, на локальном сервере можно ввести следующую команду:

```
# prlctl migrate MyVM root:passwd@remoteserver.com
```

Чтобы мигрировать виртуальную машину с удаленного сервера, на локальном сервере введите команду:

```
# prlctl migrate destserver.com/MyVM localhost
```

Если в команде не указать учетные данные целевого сервера, то это нужно будет сделать в процессе миграции.

Если вы хотите выбрать директорию на целевом сервере, куда будет помещена мигрируемая виртуальная среда, укажите полный путь до этой директории с помощью опции `--dst=<custom_path>`. Конечный путь к мигрированной виртуальной среде получится следующим: `<custom_path>/<VE_UUID>`.

По умолчанию по завершении миграции:

- виртуальная машина удаляется с исходного сервера,
- к именам собственной области и конфигурационного файла контейнера добавляется суффикс `.migrated`.

Если с командой `prlctl migrate` использовать параметр `--clone`, то виртуальная машина останется на исходном сервере, а к именам собственной области и конфигурационного файла контейнера не будет добавлен суффикс `.migrated`. Клонируемая копия будет иметь другой UUID, MAC адрес, SID (только для виртуальных машин Windows; если указан параметр `--changesid`), а также будет отключено автономное управление.

Миграция подразумевает перенос больших объемов данных с одного сервера на другой, что может занять значительно время. Для уменьшения объема переносимых данных в ПК Р-Виртуализация по умолчанию включена функция сжатия. На сжатие расходуются дополнительные ресурсы сервера, поэтому при необходимости данную функцию можно отключить с помощью параметра `--no-compression`.

Типы миграции

Между серверами ПК Р-Виртуализация можно выполнять два типа миграции:

- Офлайн миграция для остановленных и приостановленных контейнеров и виртуальных машин.
- Онлайн (живая) миграция для запущенных контейнеров и запущенных и поставленных на паузу виртуальных машин. Контейнеры и виртуальные машины могут находиться в ПК Р-Хранилище или локальном диске.

Оба типа миграции подробно описаны ниже.

Офлайн миграция виртуальных сред

Офлайн миграция подразумевает копирование по сети всех файлов виртуальной среды с одного сервера на другой.

Живая миграция виртуальных сред

Процесс живой миграции виртуальных сред является следующим:

1. В начале миграции ПК Р-Виртуализация проверяет соответствие целевого сервера необходимым требованиям и возможность миграции на него виртуальной среды.
2. Вся виртуальная память и диски виртуальной среды мигрируются на целевой сервер.
3. Виртуальная среда приостанавливается на исходном сервере.
4. Измененные страницы памяти и блоки виртуального диска, если таковые имеются, мигрируются на целевой сервер.
5. Работа виртуальной среды возобновляется на целевом сервере.

Виртуальная среда находится в запущенном состоянии во время шагов 1 и 2 и недоступна для пользователя в течение шагов 3-5. Но так как число страниц и блоков виртуального диска, измененных во время шага 2 невелико, то простой практически не заметен.

После миграции перемещенная виртуальная среда может быть недоступна по сети в течение нескольких минут из-за изменения конфигурации сетевого оборудования (например, когда маршрутизаторы обновляют таблицы динамического назначения сетей VLAN).

Примечание: Для повышения безопасности в процессе живой миграции ПК Р-Виртуализация обеспечивает туннелирование соединения между исходным и целевым серверами. Туннелирование увеличивает время миграции, поэтому, если вы хотите ускорить процесс и не

нуждаетесь в безопасном канале между серверами, то можете отключить туннелирование с помощью параметра `--no-tunnel`.

Требования и ограничения для живой миграции

При выполнении живой миграции следует принимать во внимание следующие требования и ограничения:

- Перед началом живой миграции рекомендуется синхронизировать системное время на исходном и целевом серверах, например, с помощью NTP (http://www.ntp.org). Это необходимо для определенных процессов, запущенных в виртуальных средах, которые зависят от статичности системного времени и могут повести себя непредсказуемо при возобновлении на целевом сервере, где системное время отличается.

  Примечание: В новой версии ПК P-Виртуализация синхронизация времени по NTP включена по умолчанию с помощью службы `chronyd`. Если вы хотите использовать `ntpdate` или `ntpd`, необходимо сначала остановить и отключить `chronyd`.

- Сеть должна поддерживать скорость передачи данных от 1 Гбит/с.
- Исходный и целевой серверы должны находиться в одной подсети.
- Процессоры на исходном и целевом серверах должны быть одного производителя, и возможности ЦП на целевом сервере не должны быть ниже возможностей ЦП на исходном сервере.
- Диски виртуальных сред могут находиться на локальных дисках, общих NFS и GFS2-хранилищах и raw-устройствах ISCSI.
- Живая миграция не поддерживается для виртуальных сред с открытыми сеансами `prlctl enter` и контейнеров с IPSec-соединениями.
- Контейнеры с NFS-клиентами можно мигрировать, только если выполняются следующие условия:
  - Файлы блочных и символьных устройств, совместно используемые через NFS, не открыты.
  - Удаленные файловые системы NFS, которые блокируют файлы и монтированы сверху, не используются одновременно.

  Примечание: Миграция локальных блокировок файлов поддерживается только для третьей версии NFS, так как у нее есть встроенная поддержка подобных блокировок.

- Живая миграция не поддерживается для контейнеров с включенной функцией NFS-сервера

Миграция виртуальных сред между серверами с новой версией ПК Р-Виртуализация

Между серверами с новой версией ПК Р-Виртуализация можно мигрировать виртуальные машины и контейнеры в любом режиме: онлайн или офлайн (для получения подробной информации см. **Типы миграции** (стр. 49)).

Виртуальные машины и контейнеры, хранящиеся в кластере ПК Р-Хранилище, мигрируются между серверами кластера без копирования данных, что значительно уменьшает время миграции.

Миграция виртуальных сред с ПК Р-Виртуализация старой версии на ПК Р-Виртуализация новой версии

С серверов со старой версией ПК Р-Виртуализация на серверы с новой версией ПК Р-Виртуализация можно мигрировать виртуальные машины (онлайн) и контейнеры (офлайн). В процессе миграции эти виртуальные среды будут конвертироваться в формат ПК Р-Виртуализация новой версии. В частности, устройства виртуальных машин будут заменены на устройства, поддерживаемые в ПК Р-Виртуализация новой версии (для просмотра списка оборудования виртуальных машин, поддерживаемого в ПК Р-Виртуализация новой версии, см. **Аппаратное оборудование виртуальной машины** (стр. 16)).

Примечание: Миграция с ПК Р-Виртуализация старой версии на ПК Р-Виртуализация новой версии подразумевает простой виртуальных сред, который зависит от пропускной способности сети, объема ОЗУ виртуальной машины и загрузки сервера. Для уменьшения времени простоя рекомендуется выполнять миграцию при минимальной загрузке сервера.

Перед миграцией виртуальных машин с сервера со старой версией ПК Р-Виртуализация необходимо убедиться, что выполняются следующие требования в дополнение к тем, что перечислены в подразделе **Требования и ограничения для живой миграции** (стр. 50):

- На физическом сервере должны быть установлены последние обновления для ПК Р-Виртуализация старой версии (или как минимум 6.0.11-3466).
- В виртуальной машине должна быть установлена гостевая операционная система.
- Виртуальная машина должна быть запущена.
- У виртуальной машины не должно быть снапшотов.

Например, чтобы мигрировать виртуальную машину `MyVM` с ПК Р-Виртуализация старой версии на ПК Р-Виртуализация новой версии, нужно выполнить следующую команду на сервере ПК Р-Виртуализация старой версии:

```
# prlctl migrate MyVM root:<passwd>@<RVZ7_host_IP_address_or_hostname>
```

В ходе миграции виртуальная машина копируется на целевой сервер и конвертируется в формат ПК Р-Виртуализация новой версии. После успешной миграции виртуальная

машина на сервере со старой версией ПК Р-Виртуализация лишается регистрации, и к имени ее директории добавляется суффикс `.migrated`.

Примечание: Чтобы избежать перезапуска мигрированных устаревших виртуальных машин в процессе конвертации, у подобные виртуальных машин для параметра `--on-crash` задано значение `paused` (значением по умолчанию является `restart`). После успешной миграции необходимо проверить данный параметр, например, командой `prlctl list -i <VM_UUID|VM_name> | grep crash`, и настроить его при необходимости командой `prlctl set <VM_UUID|VM_name> --on-crash restart`.

При неудачной конвертации мигрированная виртуальная машина удаляется с целевого сервера, и можно повторить попытку. Если вторая попытка также оказывается неуспешной, необходимо включить режим отладки для устаревших виртуальных машин на целевом сервере с новой версией ПК Р-Виртуализация (см. **Включение режима отладки для устаревших виртуальных машин** (стр. 171), предпринять еще одну попытку миграции, отправить отчет об ошибке и обратиться в службу технической поддержки.

В режиме отладки мигрированная виртуальная машина не будет удалена с целевого сервера с новой версией ПК Р-Виртуализация после неудачной конвертации. Она будет находиться на нем в остановленном или запущенном состоянии с отключенной сетью, чтобы команда технической поддержки имела возможность изучить дамп памяти и найти причину проблемы.

Миграция шаблонов виртуальных сред

Миграция шаблонов виртуальных машин и контейнеров между серверами ПК Р-Виртуализация аналогична офлайн-миграции виртуальных сред.

- Для миграции (перемещения) шаблона `template1` на удаленный сервер `remoteserver.com`, выполните следующую команду на локальном сервере:

```
# prlctl migrate template1 root:passwd@remoteserver.com
```

- Для миграции (перемещения) шаблона `template1` с удаленного сервера `remoteserver.com`, выполните следующую команду на локальном сервере:

```
# prlctl migrate root:passwd@remoteserver.com/template1 localhost
```

Если в команде не указаны учетные данные удаленного сервера, необходимо будет это сделать в ходе миграции.

По завершении миграции шаблон удаляется с исходного сервера (если не указан параметр `--clone`).

Миграция EZ-шаблонов

В отличие от миграции шаблонов виртуальных сред, которая по умолчанию перемещает их с одного сервера на другой, миграция EZ-шаблонов всегда копирует их на удаленный сервер. Если исходный EZ-шаблон не нужен после миграции, можно удалить его вручную с помощью команды `prlsrvctl cttemplate remove`.

Для миграции EZ-шаблонов ОС и приложений между серверами ПК Р-Виртуализация можно использовать команду `prlsrvctl cttemplate copy`. Например, чтобы копировать EZ-шаблон ОС `centos-7-x86-64` на удаленный сервер `remoteserver.com`, выполните следующую команду на локальном сервере:

```
# prlsrvctl cttemplate copy root:passwd@remoteserver.com centos-7-x86-64
```

Чтобы копировать EZ-шаблон приложения, необходимо дополнительно указать имя соответствующего EZ-шаблона ОС.

Примечание: Для успешной миграции указанный шаблон ОС должен существовать на целевом сервере.

Например, чтобы копировать EZ-шаблон приложения `mysql` шаблона ОС `centos-7-x86-64` на удаленный сервер `remoteserver.com`, выполните команду:

```
# prlsrvctl cttemplate copy root:passwd@remoteserver.com mysql centos-7-x86-64
```

Если в команде не указаны учетные данные удаленного сервера, необходимо будет это сделать в ходе миграции.

Чтобы пропустить все проверки правильности, нужно добавить к команде параметр `--force`.

По завершении миграции можно проверить, что мигрированные EZ-шаблоны находятся на целевом сервере, с помощью команды `prlsrvctl cttemplate list`:

```
# prlsrvctl cttemplate list
centos-7-x86_64     os     x86_64 yes    Centos 7 (for Intel EM64T) R-Platforma Template
...
mysql               app    x86_64 -      mysql for Centos 7 (for Intel EM64T) R-Platform
```

Выполнение операций для контейнеров

В данном разделе описываются операции для контейнеров.

Переустановка контейнеров

Переустановка контейнера может быть полезна в том случае, если какой-либо нужный файл был случайно изменен, перемещен или удален, и это привело к сбою системы контейнера. Переустановить контейнер можно при помощи команды `prlctl reinstall`, которая создает новую собственную область контейнера с нуля на основе его конфигурационного файла и соответствующих шаблонов ОС и приложений. Например:

```
# prlctl reinstall MyCT
```

Для сохранения личных данных из старого контейнера утилита также копирует содержимое старой собственной области в директорию `/vz/root/<UUID>/old` в новой собственной области (если не указан параметр `--skipbackup`). Данную директорию можно удалить после копирования нужных личных данных.

Команда `prlctl reinstall` сохраняет базу данных пользователей контейнера, если не указан параметр `--resetpwdb`.

Изменение параметров переустановки контейнеров

Стандартная переустановка, выполняемая командой `prlctl reinstall`, создает новую собственную область для неисправного контейнера, как если бы она создавалась с помощью команды `prlctl create`, и копирует собственную область неисправного контейнера в директорию `/old` в новой собственной области, чтобы не потерять ни одного файла. Также возможно удалить старую собственную область целиком без копирования или монтирования ее в новую собственную область контейнера при помощи параметра `--skipbackup`. Данный способ переустановки неисправных контейнеров может в некоторых случаях не отвечать необходимым требованиям. Одним из таких случаев является создание контейнера без использования команды `prlctl create`. Например, можно установить дополнительные лицензии на ПО в новые контейнеры или др. Тогда наилучшим вариантом будет выполнить переустановку таким способом, что неисправный контейнер возвратится к своему начальному состоянию, настроенному вручную, а не стандартными настройками команды `prlctl create`.

Для настройки переустановки необходимо написать собственные скрипты, указывающие, что должно быть сделано с контейнером при переустановке и что должно быть настроено внутри контейнера после завершения переустановки. Данные скрипты должны называться `vps.reinstall` и `vps.configure` соответственно и находиться на сервере в директории `/etc/vz/conf`. Чтобы облегчить задачу создания собственных скриптов, ПО контейнеров поставляется с образцами скриптов, которые можно использовать в качестве основы.

При выполнении команды `prlctl reinstall` *<UUID>* она ищет скрипты `vps.reinstall` и `vps.configure` и последовательно запускает их. Когда запускается скрипт `vps.reinstall`, ей передаются следующие параметры:

Параметр	Описание
`--veid`	UUID контейнера.
`--ve_private_tmp`	Путь к временной собственной области контейнера. Данный путь обозначает, куда временно создается новая собственная область контейнера. При успешном запуске скрипта данная собственная область монтируется в директорию исходной собственной области после выполнения скрипта.
`--ve_private`	Путь к исходной собственной области контейнера.

Данные параметры можно использовать в собственном скрипте `vps.reinstall`.

Если скрипт `vps.reinstall` выполняется успешно, то запускается контейнер и скрипт `vps.configure`. В этот момент старая собственная область монтируется в директорию `/old` в новой собственной области, независимо от параметра `--skipbackup`. Это делается для того, чтобы позволить использовать нужные файлы из старой собственной области в скрипте, который будет выполняться внутри запущенного контейнера. Например, если нужно скопировать некоторые файлы оттуда в обычные директории контейнера.

После выполнения скрипта `vps.configure` старая собственная область либо размонтируется и удалится, либо останется смонтированной, в зависимости от того, указан ли параметр `--skipbackup`.

Если вы не хотите запускать данные скрипты, а предпочитаете выполнить переустановку со стандартными настройками `prlctl reinstall`, можно выбрать один из двух вариантов:

- Удалить скрипты `vps.reinstall` и `vps.configure` из директории `/etc/vz/conf` или по крайней мере переименовать их.
- Изменить последнюю строку в скрипте `vps.reinstall` с `exit 0` на `exit 128`.

Прочитав код завершения `128`, утилита не будет запускать скрипты и выполнит переустановку со стандартными настройками.

Настройка VPN для контейнеров

Virtual Private Network (VPN)—это технология, которая позволяет установить безопасное сетевое соединение даже по небезопасной публичной сети. Настройка VPN для конкретного контейнера возможна через устройство TUN/TAP. Для того чтобы разрешить определенному контейнеру использовать данное устройство, нужно выполнить следующие действия:

1 Убедиться, что модуль `tun.o` уже загружен перед запуском ПК Р-Виртуализация:

```
# lsmod | grep 'tun'
```

2 Разрешить контейнеру использовать устройство TUN/TAP:

```
# vzctl set MyCT --devnodes net/tun:rw --save
```

Правильная настройка VPN является обычной задачей администрирования Linux и не рассматривается в данном руководстве. Некоторые часто используемые программы Linux для установки VPN с помощью драйвера TUN/TAP можно найти по ссылкам Virtual TUNnel и OpenVPN.

Установка NFS-сервера в контейнерах

Для установки NFS-сервера в контейнер необходимо выполнить следующие действия:

1 Убедиться, что в контейнере установлены службы `rpcbind`, `nfsd` и `nfslock`.

2 Включить функцию NFS-сервера для контейнера, выполнив команду `prlctl set --features nfsd:on` на сервере. Например:

```
# prlctl set MyCT --features nfsd:on
```

Если контейнер запущен, необходимо сначала остановить его. После включения функции перезапустите контейнер.

Примечание: Живая миграция и создание снапшотов не поддерживаются для контейнеров с включенной функцией NFS-сервера.

3 Запустить службу `rpcbind` в контейнере.

```
# service rpcbind start
Starting rpcbind:                                          [  OK  ]
```

4 Запустить службы `nfs` и `nfslock` в контейнере.

```
# service nfs start
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
# service nfslock start
Starting NFS statd:                                        [  OK  ]
```

После выполнения данных шагов можно приступить к установке общего NFS-каталога в настроенном контейнере.

Монтирование общего NFS-ресурса при запуске контейнера

Если общий NFS- ресурс настроен в файле `/etc/fstab` контейнера, созданного на базе CentOS или RHEL, и нужно, чтобы этот общий NFS-ресурс монтировался при запуске контейнера, то можно включить функцию автозапуска для службы `netfs` с помощью команды `chkconfig netfs on`.

Управление виртуальными дисками контейнеров

Примечание: Вы можете управлять виртуальными дисками как остановленных, так и запущенных контейнеров.

ПК Р-Виртуализация позволяет выполнять следующие действия с виртуальными дисками контейнеров:

- добавлять новые виртуальные диски,
- настраивать параметры виртуальных дисков,
- удалять виртуальные диски.

Добавление виртуальных дисков в контейнеры

Новые контейнеры создаются с одним виртуальным жестким диском, но при необходимости вы можете добавить новые диски следующим способом:

- подключить новый или существующий файл образа, эмулирующий дисковое устройство a, или
- подключить физический жесткий диск сервера.

Использование файлов образов

К контейнеру можно либо подключить существующий образ, либо создать новый и хранить его в выбранной директории, например, на локальном диске или в кластере ПК Р-Хранилище. Это позволяет создавать более гибкие контейнеры, в которых операционная

система может находиться на быстром SSD-диске, а данные пользователей - на жестком диске большой емкости или в ПК P-Хранилище.

Чтобы создать новый файл образа и добавить его в контейнер, можно использовать команду `prlctl set --device-add hdd`. Например:

```
# prlctl set MyCT --device-add hdd --size 100G --mnt /userdisk
```

Примечание: Если не указывать опцию `--mnt`, диск не будет подмонтирован.

Данная команда добавляет к конфигурации контейнера `MyCT` виртуальный жесткий диск со следующими параметрами:

- имя: `hdd`<*N*>, где <*N*> обозначает следующий доступный индекс диска,
- путь к образу по умолчанию: `/vz/private/<CT_UUID>/harddisk`<*N*>`.hdd`, где <*N*> обозначает следующий доступный индекс диска,
- объем: 102400 МБ,
- точка монтирования внутри контейнера: `/userdisk`. Соответствующая запись также добавляется в файл контейнера `/etc/fstab`.

Чтобы подключить существующий файл образа к контейнеру в виде виртуального жесткого диска, необходимо указать путь к файлу образа при помощи опции `--image`. Например:

```
# prlctl set MyCT --device-add hdd --image /hdd/MyCT.hdd --size 100G --mnt /userdisk
```

Подключение физических жестких дисков

К контейнеру можно подключить любое физическое блочное устройство, доступное на физическом сервере, например, локальный жеский диск или внешнее устройство, соединенное через Fibre Channel или iSCSI.

Примечание: Перед подключением к контейнеру необходимо убедиться, что физическое блочное устройство отформатировано и имеет только одну файловую систему.

Для подключения потребуется указать путь к устройству, который можно узнать с помощью команды `prlsrvctl info`. Например:

```
# prlsrvctl info
...
Hardware info:
     hdd  WDC WD1002FAEX-0 ATA (/dev/sda2)         '/dev/disk/by-id/lvm-pv-uuid-RDYrbU-
YZsH-uS8w-aH0t-EH9W-6dir-ea9lDL'
     hdd  WDC WD1002FAEX-0 ATA (/dev/sda)          '/dev/disk/by-id/wwn-
0x50014ee25a3df4dc'
   cdrom  PIONEER DVD-RW  DVR-220                  '/dev/sr0'
     net  eth0                                     'eth0'
  serial  /dev/ttyS0                               '/dev/ttyS0'
...
```

Когда путь к физическому блочному устройству известен, можно подключить его к контейнеру, используя команду `prlctl set --device-add hdd --device`. Например:

```
# prlctl set MyCT --device-add hdd --device '/dev/disk/by-id/wwn-0x50014ee25a3df4dc' \
--mnt /userdisk
```

Примечание: Если не указывать опцию `--mnt`, диск не будет подмонтирован.

Данная команда добавляет к конфигурации контейнера `MyCT` виртуальный жесткий диск со следующими параметрами:

- имя: `hdd<N>`, где <*N*> обозначает следующий доступный индекс диска,
- путь к устройству: `/dev/disk/by-id/wwn-0x50014ee25a3df4dc`, где `wwn-0x50014ee25a3df4dc` является уникальным идентификатором устройства хранения,
- точка монтирования внутри контейнера: `/userdisk`. Соответствующая запись также добавляется в файл контейнера `/etc/fstab`.

Примечания:

1. Перед миграцией контейнеров с внешними жесткими дисками необходимо убедиться, что соответствующие физические диски есть на целевом сервере и они доступны по тому же имени (по этой причине следует использовать постоянные имена, например, через `/dev/disk/by-id/`),

2. При резервном копировании контейнера подключенные к нему физические диски не будут скопированы.

2. При использовании многопутевого ввода-вывода от системы к устройству рекомендуется использовать функцию `user_friendly_names no`, чтобы данные устройства имели постоянные имена на всех серверах в кластере.

Настройка виртуальных дисков контейнеров

Чтобы настроить виртуальный диск, подключенный к контейнеру, используйте команду `prlctl set --device-set`.

Для настройки необходимо указать имя диска, которое можно узнать при помощи команды `prlctl list -i`. Например:

```
# prlctl list -i MyCT | grep "hdd"
hdd0 (+) scsi:0 image='/vz/private/9fd3eee7-70fe-43e3-9295-1ab29fe6dba5/root.hdd'
type='expanded' 10240Mb mnt=/ subtype=virtio-scsi
hdd1 (+) scsi:1 real='/dev/disk/by-id/wwn-0x50014ee25a3df4dc' mnt=/userdisk
subtype=virtio-scsi
```

Когда имя виртуального диска известно, можно настроить его параметры. Например, чтобы изменить тип виртуального диска `hdd0` в контейнере `MyCT` со SCSI на IDE, выполните команду:

```
# prlctl set MyCT --device-set hdd0 --iface ide
```

Проверить, что тип виртуального диска был изменен, можно с помощью команды `prlctl list -i`. Например:

```
# prlctl list -i MyCT | grep "hdd0"
```

```
hdd0 (+) ide:0 image='/vz/private/9fd3eee7-70fe-43e3-9295-1ab29fe6dba5/root.hdd'
type='expanded' 10240Mb mnt=/
```

Удаление виртуальных дисков из контейнеров

Чтобы удалить виртуальный диск из контейнера, используйте команду `prlctl set --device-del`.

Для удаления необходимо указать имя диска, которое можно узнать при помощи команды `prlctl list -i`. Например:

```
# prlctl list -i MyCT | grep "hdd"
hdd0 (+) scsi:0 image='/vz/private/9fd3eee7-70fe-43e3-9295-1ab29fe6dba5/root.hdd'
type='expanded' 10240Mb mnt=/ subtype=virtio-scsi
hdd1 (+) scsi:1 real='/dev/disk/by-id/wwn-0x50014ee25a3df4dc' mnt=/userdisk
subtype=virtio-scsi
```

Когда имя виртуального диска известно, можно удалить его из контейнера. Например, чтобы удалить виртуальный диск `hdd1` из контейнера `MyCT`, выполните команду:

```
# prlctl set MyCT --device-del hdd1
```

Перезапуск контейнеров

Изнутри контейнер можно перезапустить с помощью обычных команд Linux, например, `reboot` или `shutdown -r`. Перезапуск осуществляется демоном `vzeventd`.

При необходимости можно запретить перезапуск контейнеров изнутри следующими способами:

- Для отключения перезапуска для определенного контейнера необходимо добавить строку `ALLOWREBOOT="no"` в конфигурационный файл контейнера (`/etc/vz/conf/<UUID>.conf`).
- Для отключения перезапуска глобально для всех контейнеров на сервере нужно добавить строку `ALLOWREBOOT="no"` в глобальный конфигурационный файл (`/etc/vz/vz.conf`).
- Для отключения перезапуска глобально, за исключением определенных контейнеров, нужно добавить строки `ALLOWREBOOT="no"` в глобальный конфигурационный файл (`/etc/vz/vz.conf`) и `ALLOWREBOOT="yes"` в конфигурационные файлы соответствующих контейнеров.

Создание контейнеров на базе SimFS

В ПК Р-Виртуализация разметка simfs основана на создании ссылок на каталоги (bindmount). Когда контейнер, созданный на базе simfs, запускается, его собственная область монтируется в корневую область контейнера.

Чтобы создать контейнер на базе simfs необходимо выполнить следующие действия:

1 Задать параметр `VEFSTYPE=simfs` в файле `/etc/vz/vz.conf`.

2 Выполнить команду `prlctl create <CT_name>`.

В ПК Р-Виртуализация simfs 7 имеет следующие ограничения:

- Не поддерживается квотирование первого или второго уровня.
- Не поддерживается живая миграция контейнеров, созданных на базе simfs.

Выполнение операций для виртуальных машин

В данном разделе описываются операции для виртуальных машин.

Пауза виртуальных машин

Пауза запущенной виртуальной машины освобождает ресурсы памяти и ЦП, используемые данной виртуальной машиной. Освобожденные ресурсы затем могут использоваться физическим сервером или другими запущенными виртуальными средами.

Чтобы поставить виртуальную машину на паузу, можно использовать команду `prlctl pause`. Например, следующая команда ставит на паузу виртуальную машину `MyVM`:

```
# prlctl pause MyVM
```

Проверить, была ли виртуальная машина поставлена на паузу, можно с помощью команды `prlctl list -a`:

```
# prlctl list -a
STATUS       IP_ADDR       NAME
running      10.10.10.101  MyCT
paused       10.10.10.201  MyVM
```

Вывод команды показывает, что виртуальная машина `MyVM` в текущий момент поставлена на паузу. Для возобновления работы виртуальной машины нужно ввести команду:

```
# prlctl start MyVM
```

Управление устройствами виртуальных машин

ПК Р-Виртуализация позволяет управлять следующими устройствами виртуальных машин:

- жесткие диски
- CD/DVD-ROM устройства
- флоппи-диски
- сетевые адаптеры
- последовательные порты
- USB-контроллеры

Основными операциями с данными устройствами являются:

- добавление нового устройства в виртуальную машину
- настройка параметров устройства
- удаление устройства из виртуальной машины

Добавление новых устройств

Добавить новые виртуальные устройства в виртуальную машину можно с помощью команды `prlctl set`. Параметры для добавления определенных устройств перечислены в таблице ниже:

Параметр	Описание
`hdd`	Добавляет жесткий диск в виртуальную машину. Можно подключить существующий образ к виртуальной машине или создать новый. **Примечание:** Жесткие диски SCSI и VirtIO можно добавить как в остановленные, так и в запущенные виртуальные машины, IDE-диски можно добавить только в остановленные виртуальные машины.
`cdrom`	Добавляет CD/DVD-ROM диск в виртуальную машину.
`net`	Добавляет сетевой адаптер в виртуальную машину.
`fdd`	Добавляет флоппи-диск в виртуальную машину.
`serial`	Добавляет последовательный порт в виртуальную машину:
`usb`	Добавляет USB-контроллер в виртуальную машину.

Например, для добавления виртуального диска в виртуальную машину `MyVM` можно выполнить следующую команду:

```
# prlctl set MyVM --device-add hdd
Creating hdd1 (+) scsi:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk1.hdd' type='expanded' 65536Mb subtype=virtio-scsi
Created hdd1 (+) scsi:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk1.hdd' type='expanded' subtype=virtio-scsi
```

По умолчанию данная команда создает новый виртуальный диск со следующими параметрами:

- имя: `hdd1`
- тип диска: SCSI
- подтип диска: VirtIO SCSI
- имя и местоположение образа диска: `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk1.hdd`
- формат диска: расширяемый (`expanded`)
- емкость диска: 65536 МБ

Некоторые из данных параметров можно изменить с помощью определенных параметров для команды. Например, чтобы создать виртуальный диск IDE емкостью 84 ГБ, нужно выполнить следующую команду:

```
# prlctl set MyVM --device-add hdd --size 84000 --iface ide
Creating hdd2 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk2.hdd' type='expanded' 84000Mb
Created hdd2 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk2.hdd' type='expanded'
```

Виртуальный диск добавлен в виртуальную машину. Но перед его использованием необходимо его инициализировать. Подробная инструкция об инициализации диска представлена в следующем подразделе.

При управлении устройствами следует иметь в виду следующее:

- К виртуальной машине можно подключить до:
  - 4 IDE-устройств (виртуальных жестких дисков или CD/DVD-ROM дисков)
  - 15 SCSI-устройств (виртуальных жестких дисков или CD/DVD-ROM дисков)
  - 15 виртуальных жестких дисков VirtIO
- В виртуальной машине может быть до 16 виртуальных сетевых адаптеров.
- В виртуальной машине может быть до 4 последовательных портов.
- В виртуальной машине может быть только 1 USB-контроллер.
- В виртуальной машине может быть только 1 флоппи-диск.

Добавление устройств Hyper-V SCSI в виртуальные машины Windows

ПК Р-Виртуализация поддерживает эмуляцию паравиртуализованных с помощью Hyper-V жестких и CD/DVD-ROM дисков SCSI. Эмуляция позволяет использовать данные устройства с собственными драйверами Windows. Эмуляция поддерживается и рекомендуется для следующих операционных систем Windows:

- Windows 10 (x64),
- Windows Server 2016,
- Windows Server 2012 R2,
- Windows Server 2012,
- Windows Server 2008 R2 с SP2.

Чтобы добавить, например, CD/DVD-ROM и жесткий диск SCSI , эмулированный с помощью Hyper-V, в виртуальную машину, выполните следующие команды:

```
# prlctl set vm1 --device-add cdrom --image <path_to_image> --iface scsi --subtype
hyperv
# prlctl set vm1 --device-add hdd --iface scsi --subtype hyperv
```

Инициализация добавленного диска

После добавления нового пустого виртуального жесткого диска к конфигурации виртуальной машины он будет невидим для операционной системы, установленной в виртуальной машине, до его инициализации.

Инициализация нового виртуального жесткого диска в Windows

Для инициализации нового виртуального жесткого диска в гостевой ОС Windows необходимо использовать утилиту Disk Management. Например, в Windows Server 2012 можно запустить данную утилиту, щелкнув **Start** > **Control Panel** > **System and Security** > **Administrative Tools** > **Computer Management** > **Storage** > **Disk Management**.

При открытии окна утилиты Disk Management она автоматически определяет новый жесткий диск, добавленный к конфигурации, и запускает мастер **Initialize Disk**, в котором нужно выполнить следующие действия:

1. В секции **Select disks** выбрать добавленный диск.
2. Выбрать стиль раздела для выбранного диска: MBR (Master Boot Record) или GPD (GUID Partition Table).
3. Щелкнуть **OK**.

Добавленный диск появится в окне утилиты Disk Management в виде нового диска, но его пространство памяти будет нераспределенным. Чтобы распределить память диска, нужно щелкнуть правой кнопкой мыши по имени диска в окне Disk Management и выбрать **New Volume**. Появится окно **New Volume Wizard**. Далее нужно выполнить шаги в мастере и создать новый том в добавленном диске.

После выполнения всех шагов диск станет видимым в **My Computer**, и можно будет его использовать в качестве диска для хранения данных в виртуальной машине.

Инициализация нового виртуального жесткого диска в Linux

Инициализация нового виртуального жесткого диска в гостевой ОС состоит из двух шагов: (1) распределения пространства виртуального жесткого диска и (2) монтирования данного диска в гостевую ОС.

Для распределения пространства необходимо создать новый раздел на виртуальном диске с помощью утилиты `fdisk`, выполнив следующие действия:

Примечание: Для использования утилиты `fdisk` требуются привилегии пользователя `root`.

1. Запустить окно терминала.
2. Для получения списка всех IDE-дисков в виртуальной машине введите команду:

```
fdisk /dev/hd*
```

Примечание: Если к конфигурации виртуальной машины добавлен SCSI-диск, нужно использовать команду `fdisk /dev/sd*`.

3. По умолчанию второй виртуальный жесткий диск появляется в `/dev/hdc` в виртуальной машине Linux. Для работы с данным устройством введите команду:

```
fdisk /dev/hdc
```

Примечание: Для SCSI-диска следует использовать команду `fdisk /dev/sdc`.

4 Для получения подробной информации выполните:

```
p
```

5 Чтобы создать новый раздел, введите:

```
n
```

6 Для создания первичного раздела введите:

```
p
```

7 Укажите номер раздела. По умолчанию указывается номер 1.

8 Укажите первый цилиндр. Чтобы создать единственный раздел на жестком диске, можно использовать значение по умолчанию.

9 Укажите последний цилиндр. Чтобы создать единственный раздел на жестком диске, можно использовать значение по умолчанию.

10 Для создания раздела с заданными настройками введите:

```
w
```

После распределения пространства на добавленном виртуальном жестком диске необходимо его форматировать с помощью следующей команды, вводимой в терминале:

```
mkfs -t <FileSystem> /dev/hdc1
```

Примечание: <*FileSystem*> обозначает файловую систему, которая будет использоваться на данном диске. Рекомендуется использовать `ext4`.

После форматирования добавленного виртуального жесткого диска можно монтировать его в гостевую ОС.

1 Для создания точки монтирования для нового виртуального жесткого диска нужно выполнить команду:

```
mkdir /mnt/hdc1
```

Примечание: Можно указать другую точку монтирования.

2 Чтобы монтировать новый виртуальный жесткий диск в указанную точку монтирования введите:

```
mount /dev/hdc1 /mnt/hdc1
```

После монтирования виртуального жесткого диска можно использовать его пространство в виртуальной машине.

Настройка виртуальных устройств

В ПК Р-Виртуализация можно использовать параметр `--device-set` команды `prlctl set` для настройки параметров существующего виртуального устройства. Как правило, процесс настройки параметров устройства состоит из двух шагов:

1 Выяснение имени устройства, которое необходимо настроить.

2 Выполнение команды `prlctl set` для настройки нужных параметров устройства.

Выяснение имен устройств

Для настройки виртуального устройства необходимо указать его имя при вводе команды `prlctl set`. Если имя устройства неизвестно, можно использовать команду `prlctl list`, чтобы узнать его. Например, получить список виртуальных устройств в виртуальной машине `MyVM` можно с помощью команды:

```
# prlctl list --info MyVM
...
Hardware:
  cpu cpus=2 VT-x accl=high mode=32 ioprio=4 iolimit='0'
  memory 1024Mb
  video 32Mb 3d acceleration=off vertical sync=yes
  fdd0 (+) real='/dev/fd0' state=disconnected
  hdd0 (+) scsi:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk.hdd' type='expanded' subtype=virtio-scsi
  hdd1 (+) scsi:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk1.hdd' type='expanded' subtype=virtio-scsi
  cdrom0 (+) ide:1 real='Default CD/DVD-ROM'
  usb (+)
  net0 (+) dev='vme426f6594' network='Bridged' mac=001C426F6594 card=virtio
...
```

Все виртуальные устройства, доступные в данный момент в виртуальной машине, перечислены в секции `Hardware`. В примере выше виртуальная машина `MyVM` имеет следующие устройства: 2 ЦП, оперативную память, видеопамять, флоппи-диск, 2 жестких диска, CD/DVD-ROM диск, USB-контроллер и сетевую карту.

Настройка виртуальных устройств

После выяснения имени виртуального устройства можно настроить его параметры. Например, чтобы изменить тип виртуального диска `hdd1` со SCSI на IDE в виртуальной машине `MyVM`, нужно выполнить следующую команду:

```
# prlctl set MyVM --device-set hdd1 --iface ide
Configured hdd1 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk1.hdd' type='expanded'
```

Проверить, что тип виртуального диска был изменен, можно с помощью команды `prlctl list --info`:

```
# prlctl list --info MyVM
...
hdd1 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk1.hdd'
type='expanded'
...
```

Подключение и отключение виртуальных устройств

ПК Р-Виртуализация позволяет подключать и отключать определенные устройства, когда виртуальная машина запущена. Данные устройства включают:

- CD/DVD-ROM диски
- флоппи-диски
- сетевые адаптеры

- порты принтера
- последовательные порты

Обычно все виртуальные устройства подключаются автоматически к виртуальной машине в процессе их создании. Чтобы отключить устройство от виртуальной машины, можно использовать команду `prlctl set`. Например, следующая команда отключает CD/DVD-ROM диск `cdrom0` от виртуальной машины `MyVM`:

```
# prlctl set MyVM --device-disconnect cdrom0
Disconnect device: cdrom0
```

Чтобы повторно подключить CD/DVD-ROM диск, можно выполнить команду:

```
# prlctl set MyVM --device-connect cdrom0
Connect device: cdrom0
```

Удаление устройств

Удалить ненужное виртуальное устройство можно при помощи параметра `--device-del` команды `prlctl set`. Параметры для удаления определенных устройств перечислены в таблице ниже:

Параметр	Описание
`hdd`	Удаляет указанный жесткий диск из виртуальной машины. **Примечание:** Жесткие диски можно удалить только из остановленных виртуальных машин.
`cdrom`	Удаляет указанный CD/DVD-ROM диск из виртуальной машины.
`net`	Удаляет указанный сетевой адаптер из виртуальной машины.
`fdd`	Удаляет флоппи-диск из виртуальной машины.
`serial`	Удаляет указанный последовательный порт из виртуальной машины:
`usb`	Удаляет USB-контроллер из виртуальной машины.

Как правило, удаление виртуального устройства состоит из двух шагов:

1 Выяснение имени устройства, которое необходимо удалить.

2 Удаление устройства из виртуальной машины.

Выяснение имени устройства

Для удаления виртуального устройства необходимо указать его имя при вводе команды `prlctl set`. Если имя устройства неизвестно, можно использовать команду `prlctl list`, чтобы узнать его (для получения подробной информации см. **Выяснение имен устройств (стр. 65)**).

Удаление виртуального устройства

После выяснения имени виртуального устройства можно удалить его из виртуальной машины. Например, чтобы удалить виртуальный диск `hdd1` из виртуальной машины `MyVM`, можно выполнить следующую команду

```
# prlctl set MyVM --device-del hdd1
Remove the hdd1 device.
```

Если безвозвратно удалять виртуальное устройство не нужно, можно временно отключить его от виртуальной машины с помощью параметра `--disable`.

Создание скриншотов

Если виртуальная машина перестала отвечать на запросы, можно проверить ее состояние путем захвата изображения (или скриншота) ее экрана при помощи команды `prlctl capture`. Скриншот сохраняется в формате `.png`.

Примечание: Делать скриншоты можно только запущенных виртуальных машин.

Чтобы сделать скриншот экрана виртуальной машины `MyVM` и сохранить его в файл `/usr/screenshots/image1.png`:

1 Убедитесь, что виртуальная машина запущена:

```
# prlctl list
UUID                                    STATUS       IP_ADDR       T      NAME
{b2de86d9-6539-4ccc-9120-928b33ed31b9}  running      10.10.100.1   VM     MyVM
```

2 Сделайте скриншот виртуальной машины:

```
# prlctl capture MyVM --file /usr/screenshots/image1.png
```

Если не указан параметр `--file`, скриншот выводится на экран в выводе команды.

3 Проверьте, что файл `image1.png` создан:

```
# ls /usr/screenshots/
image1.png
```

Настройка диапазонов IP–адресов для сетей Host-Only

Все виртуальные машины, подключенные к сетям host-only, получают IP-адреса от DHCP-сервера. Данный DHCP-сервер настраивается в ходе установки ПК Р-Виртуализация и по умолчанию включает IP-адреса от 10.37.130.1 до 10.37.130.254. Этот диапазон IP-адресов можно перенастроить для сетей host-only, чтобы виртуальные машины могли получать IP-адреса из других диапазонов. Например, чтобы указать 10.10.11.1 и 10.10.11.254 для начального и конечного IP-адресов соответственно для сети `Host-Only` (данная сеть автоматически создается при установке ПК Р-Виртуализация), нужно выполнить команду:

```
# prlsrvctl net set Host-Only --ip-scope-start 10.10.11.1 --ip-scope-end 10.10.11.254
```

Также можно указать собственный диапазон IP-адресов непосредственно при создании сети host-only. Чтобы создать сеть с именем `Host-Only2` и определить диапазон IP-адресов для данной сети от 10.10.10.1 до 10.10.10.254, введите команду:

```
# prlsrvctl net add Host-Only2 -t host-only --ip-scope-start 10.10.10.1 --ip-scope-end
10.10.10.254
```

При работе с диапазонами IP-адресов следует иметь в виду следующее:

- Начальный и конечный IP-адреса должны относиться к одной подсети.
- Диапазоны IP-адресов можно определить для каждой сети host-only отдельно. Например, можно указать диапазон IP-адресов от 10.10.11.1 до 10.10.11.254 для сети `Host-Only` и от 10.10.10.1 до 10.10.10.254 для сети `Host-Only2`.

Настройка режима отказа виртуальных машин

В ПК Р-Виртуализация можно настроить поведение виртуальной машины после сбоя гостевой операционной системы. По умолчанию при отказе виртуальной машины создается аварийный дамп, отправляется отчет об ошибке в службу поддержки, и виртуальная машина перезапускается с неизмененной конфигурацией.

Для того чтобы самостоятельно разобраться с проблемой, можно переключить режим отказа виртуальной машины на `pause` с помощью команды `prlctl set`. Например:

```
# prlctl set MyVM --on-crash pause
```

Ресурсы виртуальной машины будут сохранены для анализа, и аварийный дамп будет отправлен в отчете об ошибке.

Так как дампы могут занимать значительное место на диске и привести к неисправности всего сервера, режим отказа автоматически переключается на `pause`, если сбои виртуальной машины происходят более трех раз в течение 24 часов с последнего отказа.

Если вы хотите пропустить создание аварийного дампа и отправку отчета об ошибке, добавьте `:no-report` после команды. Например:

```
# prlctl set MyVM --on-crash restart:no-report
```

Глава 3

Управление ресурсами

Главной целью управления ресурсами в ПК Р-Виртуализация является обеспечение Управления Качеством Услуг (Service Level Management) или Качества обслуживания (Quality of Service) для виртуальных машин и контейнеров. Правильно настроенные параметры ресурсов могут предотвратить серьезные последствия, вызванные чрезмерным потреблением какого-либо ресурса (случайным или злоумышленным) виртуальной машины или контейнера, на остальные виртуальные среды. Использование параметров ресурсов для управления ресурсами также позволяет добиться справедливого использования ресурсов среди виртуальных сред и при необходимости лучшего качества обслуживания для выбранных виртуальных сред. Все параметры ресурсов можно указать при помощи утилит командной строки.

Управление ресурсами ЦП

Для виртуальных машин и контейнеров можно управлять следующими параметрами ресурсов ЦП:

- единицы ЦП для виртуальных машин и контейнеров
- привязка к ЦП виртуальных машин и контейнеров
- лимиты ЦП для виртуальных машин и контейнеров
- узлы NUMA для контейнеров
- горячее подключение ЦП для виртуальных машин
- топология ЦП для виртуальных машин

Данные параметры подробно описаны ниже.

Настройка единиц ЦП

Единицы ЦП определяют, сколько процессорного времени одна виртуальная среда может получить по сравнению с другими виртуальными средами на физическом сервере, если все ЦП сервера используются полностью. Например, если для контейнера `MyCT` и виртуальной машины `MyVM` указано значение 1000 единиц ЦП для каждого, а для контейнера `MyCT2`—2000 единиц ЦП, то контейнер `MyCT2` получит в два раза больше процессорного времени, чем контейнер `MyCT` или виртуальная машина `MyVM` при полной загрузке всех ЦП на сервере.

По умолчанию каждая виртуальная среда на сервере получает 1000 единиц ЦП. Изменить стандартную настройку можно с помощью команды `prlctl set`. Например, чтобы

выделить 2000 единиц ЦП контейнеру MyCT и виртуальной машине MyVM, нужно выполнить следующие команды:

```
# prlctl set MyCT --cpuunits 2000
# prlctl set MyVM --cpuunits 2000
```

Настройка привязки виртуальных сред к ЦП

Если на физическом сервере установлено несколько ЦП, можно привязать виртуальную среду к определенному ЦП, чтобы только данные ЦП использовались для обработки процессов в виртуальной среде. Функция привязывания определенных процессов к определенным ЦП называется *привязкой ЦП*.

По умолчанию любая новая виртуальная среда может использовать процессорное время всех ЦП, установленных на физическом сервере. Чтобы привязать виртуальную среду к определенным ЦП, можно использовать параметр `--cpumask` команды `prlctl set`. Если на физическом сервере установлено 8 ЦП, можно настроить виртуальную машину MyVM и контейнер MyCT таким образом, чтобы их процессы выполнялись на ЦП 0, 1, 3, 4, 5 и 6 с помощью следующих команд:

```
# prlctl set MyVM --cpumask 0,1,3,4-6
# prlctl set MyCT --cpumask 0,1,3,4-6
```

Также можно указать маску привязки ЦП–то есть, процессоры, которые нужно привязать к виртуальным средам—в виде отдельных номеров ЦП (0,1,3) или диапазонов ЦП (4-6). При настройке маски привязки ЦП для запущенных виртуальных сред изменения применяются “на лету”.

Чтобы отменить изменения, сделанные для виртуальной машины MyVM и контейнера MyCT и настроить их процессы для выполнения на всех доступных ЦП, необходимо выполнить команды:

```
# prlctl set MyVM --cpumask all
# prlctl set MyCT --cpumask all
```

Настройка лимитов ЦП

Лимит ЦП указывает максимальную мощность ЦП, которую может получить виртуальная среда для своих процессов. Контейнеру не позволяется превышать указанный лимит, даже если сервер имеет достаточно свободной мощности ЦП. По умолчанию параметр лимита ЦП отключен для всех новых виртуальных сред. Это означает, что любое приложение в любой виртуальной среде может использовать всю свободную мощность ЦП на сервере.

Примечание: Можно настроить какие потоки виртуальной машины — вспомогательные и рабочие или только рабочие — ограничиваются параметрами, описанными ниже. Для данного изменения необходимо ввести команду `prlsrvctl set --vm-cpulimit-type <full|guest>` и перезапустить виртуальные машины.

Чтобы задать лимит ЦП для виртуальной среды, можно использовать один из двух параметров: `--cpulimit`, `--cpus`. Оба параметра подробно описаны ниже.

Использование --cpulimit для установки лимита ЦП

Как правило, лимит ЦП задается для виртуальной среды с помощью команды `prlctl set --cpulimit`. В примере ниже контейнер `MyCT` настраивается для получения не более 25% процессорного времени сервера даже при неполной загрузке ЦП на сервере:

```
# prlctl set MyCT --cpulimit 25
```

Данная команда устанавливает лимит ЦП для контейнера `MyCT` в 25% от общей мощности ЦП сервера. Общая мощность ЦП сервера в процентах рассчитывается путем умножения количества логических ядер ЦП, установленных на сервере, на 100%. Таким образом, если сервер имеет 2 логических ядра ЦП, 2 ГГц каждое, то общая мощность ЦП будет равна 200%, и контейнер `MyCT` будет иметь лимит в 500 МГц.

Например, на физическом сервере с 2 логическими ядрами ЦП, 3 ГГц каждое, контейнер `MyCT` сможет получить 25% от 6 ГГц, то есть, 750 МГц. Чтобы обеспечить всегда одинаковый лимит ЦП на всех серверах для контейнера `MyCT`, независимо от общей мощности ЦП данных серверов, можно задать лимит ЦП в мегагерцах (МГц). Например, чтобы контейнер `MyCT` потреблял не более 500 МГц на любом физическом сервере, нужно выполнить следующую команду:

```
# prlctl set MyCT --cpulimit 500m
```

Примечание: Для получения более подробной информации об установке лимитов ЦП для виртуальных машин и контейнеров см. также **Особенности лимитов ЦП** (стр. 72).

Использование --cpus для установки лимита ЦП

Другим способом задания лимита ЦП для виртуальной среды является использование команды `prlctl set --cpus`. В данном случае можно указать, сколько логических ядер ЦП на один сокет ЦП сможет использовать виртуальная среда. Например, так как контейнеры имеют только один сокет ЦП (см. **Настройка топологии ЦП для виртуальных машин** (стр. 74)), то можно позволить контейнеру `MyCT` использовать только 2 ядра с помощью следующей команды:

```
# prlctl set MyCT --cpus 2
```

Проверить, что лимит ЦП был установлен, можно в `/proc/cpuinfo` контейнера. Например:

```
# prlctl exec MyCT cat /proc/cpuinfo | grep "cpu cores"
cpu cores       : 2
```

Одновременное использование --cpulimit и --cpus

При одновременном использовании обоих параметров `--cpulimit` и `--cpus` для задания лимита ЦП для виртуальной среды, будет применен наименьший лимит. Например, выполнение данных команд на сервере с 4 ЦП, каждый в 2 ГГц, установит лимит для контейнера `MyCT` в 2 ГГц:

```
# prlctl set MyCT --cpus 2
# prlctl set MyCT --cpulimit 2000m
```

Особенности лимитов ЦП

ПК Р-Виртуализация устанавливает внутри системы лимиты ЦП для виртуальных сред в процентах. В многоядерных системах считается, что каждое логическое ядро ЦП имеет 100% мощность ЦП. Таким образом, если на сервере есть 4 ядра ЦП, то общая мощность ЦП сервера равна 400%.

Также можно задать лимит ЦП в мегагерцах (МГц). Если лимит указан в МГц, то для преобразования мощности ЦП сервера из МГц в проценты ПК Р-Виртуализация использует следующую формулу: `CPULIMIT_% = 100% * CPULIMIT_MHz / CPUFREQ`, где

- `CPULIMIT_%` обозначает общую мощность ЦП сервера в процентах,
- `CPULIMIT_MHz` является общей мощностью ЦП сервера в мегагерцах,
- `CPUFREQ`—это частота одного ядра ЦП на сервере.

При установке лимитов ЦП следует иметь в виду следующее:

- Лимит ЦП, который будет задан виртуальной среде, не должен превышать общую мощность ЦП сервера. Таким образом, если сервер имеет 4 ЦП, 1000 МГц каждый, нельзя установить лимит ЦП более чем в 4000 МГц.
- Процессы, запущенные в виртуальной среде, будут выполняться на всех ЦП сервера в равных долях. Например, если сервер имеет 4 ЦП, 1000 МГц каждый, и установлен лимит ЦП для виртуальной машины в 2000 МГц, то виртуальная среда будет использовать 500 МГц с каждого ЦП.
- Все запущенные виртуальные среды на сервере не могут одновременно использовать больше мощности ЦП, чем физически доступно на сервере. Другими словами, если общая мощность ЦП сервера равна 4000 МГц, запущенные виртуальные среды на данном сервере не смогут использовать более 4000 МГц, независимо от их лимитов ЦП. Однако общий лимит ЦП всех виртуальных сред может превышать общую мощность ЦП сервера, так как большую часть времени виртуальные среды используют не всю назначенную им мощность ЦП.

Привязка ЦП к узлам NUMA

В системах с архитектурой с неравномерной памятью (NUMA) можно настроить, чтобы контейнеры использовали ЦП только из определенных узлов NUMA.

Можно рассмотреть следующую ситуацию:

- На физическом сервере установлено 8 ЦП.
- ЦП разделены на 2 узла NUMA: узел NUMA 0 и узел NUMA 1. В каждом узле NUMA есть 4 ЦП.
- Необходимо, чтобы процессы в контейнере `MyCT` выполнялись на процессорах из узла NUMA 1.

Чтобы настроить контейнер MyCT использовать процессоры из узла NUMA 1, нужно выполнить следующую команду:

```
# prlctl set MyCT --nodemask 1
```

Проверить, что контейнер MyCT теперь привязал к узлу NUMA 1, можно с помощью команды:

```
# prlctl list -i MyCT | grep nodemask
  cpu cpus=unlimited VT-x hotplug accl=high mode=32 cpuunits=1000 ioprio=4 nodemask=1
```

Отменить привязку контейнера MyCT от узла NUMA 1 можно с помощью команды:

```
# prlctl set MyCT --nodemask all
```

Теперь контейнер MyCT может снова использовать все ЦП, доступные на сервере.

Примечание: Получить более подробную информацию по NUMA можно по ссылке http://lse.sourceforge.net/numa.

Включение горячего подключения ЦП для виртуальных машин

Если гостевая операционная система поддерживает функцию горячего подключения ЦП, ее можно включить для виртуальной машины. Функция горячего подключения ЦП позволяет увеличить число ЦП, доступных для виртуальных машин, даже если они запущены.

На текущий момент поддержка горячего подключения ЦП осуществляется для следующих систем:

- ОС Linux на базе ядра RHEL 5 и более поздних версий (Red Hat Linux Enterprise 5, CentOS 5 и др.)
- x64 версия Windows Server 2008 R2 (Datacenter Edition)
- x64 версия Windows Server 2012 (Standard и Datacenter Edition)
- x64 версия Windows Server 2008 (Standard Edition)
- x64 версия Windows Server 2008 (Enterprise Edition)
- x64 версия Windows Server 2008 (Datacenter Edition)

По умолчанию поддержка горячего подключения ЦП отключена для всех новых виртуальных машин. Для включения данной функции можно использовать параметр `--cpu-hotplug` команды `prlctl set`. Например, чтобы включить поддержку горячего подключения ЦП в виртуальной машине MyVM , в которой запущена одна из поддерживаемых операционных систем, нужно остановить виртуальную машину MyVM и выполнить следующую команду:

```
# prlctl set MyVM --cpu-hotplug on
set cpu hotplug: 1
```

После включения функции можно увеличить число ЦП в виртуальной машине MyVM, даже если она запущена. Если предположить, что на физическом сервере установлено 4 ЦП и

процессы в виртуальной машине MyVM настроены на выполнение на двух ЦП, то назначить 3 ЦП виртуальной машине можно с помощью следующей команды:

```
# prlctl set MyVM --cpus 3
set cpus(4): 3
```

Отключить поддержку горячего подключения ЦП в виртуальной машине MyVM можно командой:

```
# prlctl set MyVM --cpu-hotplug off
set cpu hotplug: 0
```

Изменения будут применены к виртуальной машине при ее следующем запуске.

Настройка топологии ЦП для виртуальных машин

В текущей версии ПК Р-Виртуализация вы можете задать топологию ЦП для виртуальных машин, т.е. число сокетов ЦП и ядер ЦП на один сокет. Это может понадобиться, например, если вы используете гостевую операционную систему, которая поддерживает определенное число сокетов ЦП (например, когда оно ограничено лицензией). В этом случае можно настроить конфигурацию виртуальной машины таким образом, чтобы она использовала максимальное доступное число ядер ЦП, и тем самым обеспечить высокую производительность гостевой ОС. Общее число ядер ЦП, которое доступно для виртуальной машины, вычисляется путем умножения числа сокетов ЦП на число ядер ЦП на один сокет. Оно не может превышать число ядер ЦП на физическом сервере.

По умолчанию виртуальная машина создается с одним сокетом ЦП и двумя ядрами ЦП.

Примечание: Контейнер имеет только один сокет ЦП, и данный параметр нельзя изменить.

Для виртуальной машины можно изменить число сокетов ЦП и ядер ЦП на один сокет при помощи параметров `--cpu-sockets` и `--cpus` для команды `prlctl set`. Например, если на физическом сервере есть 4 ядра ЦП, вы можете позволить виртуальной машине MyVM использовать 2 сокета ЦП и 2 ядра ЦП на один сокет, выполнив следующую команду:

```
# prlctl set MyVM --cpu-sockets 2 --cpus 2
```

Если виртуальная машина запущена, ее конфигурация изменится после перезапуска.

Чтобы проверить текущую топологию ЦП виртуальной машины, выполните следующую команду:

```
# prlctl list MyVM -i | grep "cpu"
cpu sockets=2 cpus=2 cores=2 <...>
```

Управление дисковыми квотами

Ограничить дисковое пространство, которое могут использовать отдельные пользователи и группы в контейнере, можно при помощи стандартных инструментов Linux из пакета `quota`.

Перед настройкой дисковых квот в контейнере необходимо включить их для данного контейнера следующим образом:

1 Указать значение параметра `QUOTAUGIDLIMIT` равное `1` в конфигурационном файле контейнера (`/etc/vz/conf/<UUID>.conf`) или выполнить команду `prlctl set <UUID> --quotaugidlimit 1`.

2 Перезапустить контейнер.

Управление виртуальными дисками

В ПК Р-Виртуализация можно управлять виртуальными дисками следующим образом:

- изменить емкость,
- выполнить сжатие (уменьшить размер, который виртуальные диски занимают на физическом жестком диске),
- изменить интерфейс.

Данные операции описаны ниже.

Изменение емкости диска

Важно: Используйте инструмент `prl_disk_tool` только для дисков остановленных виртуальных машин.

Емкость виртуальных жестких дисков виртуальных машин или контейнеров можно изменить с помощью команды `prl_disk_tool resize --size`. Например:

```
# prl_disk_tool resize --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/
harddisk.hdd --size 30G
```

При увеличении дискового пространства следует иметь в виду следующее:

- У виртуальной машины, емкость виртуального диска которой нужно изменить, не должно быть снапшотов.
- Емкость расширяемого виртуального диска, показанная изнутри виртуальной среды, и размер, который занимает виртуальный диск на физическом сервере, могут отличаться.
- При увеличении емкости виртуального диска дополнительное дисковое пространство добавляется нераспределенным. Распределить добавленное дисковое пространство можно с помощью стандартных средств гостевой ОС.
- Невозможно уменьшить файловые системы XFS (стандартный выбор у CentOS 7 и Red Hat Enterprise Linux 7).

Проверка минимальной емкости диска

Перед уменьшением емкости диска можно узнать минимальное значение, до которого возможно уменьшение, с помощью команды `prl_disk_tool resize --info`. Например, если диск `hdd0` виртуальной машины `MyVM` эмулирован образом `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk.hdd`, нужно выполнить команду:

```
# prl_disk_tool resize --info --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/
harddisk.hdd
Disk information:
...
Minimum: 2338M
...
```

Сжатие дисков

Важно: Используйте инструмент `prl_disk_tool` только для дисков остановленных виртуальных машин.

В ПК Р-Виртуализация можно уменьшить место, которое виртуальные среды занимают на физическом диске, путем сжатия их виртуальных дисков. Данный способ позволяет сэкономить дисковое пространство и разместить большее число виртуальных сред на сервере.

Для сжатия виртуального диска можно использовать команду `prl_disk_tool compact`. Например, чтобы сжать диск `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk.hdd`, нужно выполнить следующую команду:

```
# prl_disk_tool compact --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/
harddisk.hdd/
```

Проверить, сколько места освободилось при сжатии виртуального диска, можно с помощью стандартных утилит Linux (например, `df`).

Управление интерфейсами дисков виртуальных машин

По умолчанию любая виртуальная машина создается с виртуальным жестким диском SCSI. При необходимости можно изменить тип интерфейса диска со SCSI на IDE или VirtIO. Например, чтобы изменить тип интерфейса стандартного диска (`hdd0`) в виртуальной машине `MyVM` со SCSI на IDE, необходимо выполнить следующую команду:

```
# prlctl set MyVM --device-set hdd0 --iface ide
```

Проверить, что тип интерфейса был успешно изменен, можно с помощью команды:

```
# prlctl list -i MyVM | grep hdd0
Boot order: hdd0 cdrom0 fdd0 net0
hdd0 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk.hdd'
```

Из вывода команды видно, что тип интерфейса у диска `hdd0` теперь IDE.

Для виртуальной машины MyVM можно создать дополнительные диски. Например, чтобы добавить новый диск типа IDE в виртуальную машину, введите команду:

```
# prlctl set MyVM --device-add hdd --iface ide
Creating hdd1 (+) ide:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-
8fef6178bc5b/harddisk1.hdd' 65536Mb
Create the expanding image file, 65536Mb...
```

Чтобы создать диск VirtIO следует указать параметр `--iface virtio` вместо `--iface ide`. Если параметр `--iface` не указан, то по умолчанию будет создан диск SCSI.

Максимальное количество устройств, которое можно добавить в виртуальную машину, указано ниже:

- 4 IDE-устройств (виртуальных жестких дисков или CD/DVD-ROM дисков)
- 15 SCSI-устройств (виртуальных жестких дисков или CD/DVD-ROM дисков)
- 15 виртуальных жестких дисков VirtIO

В любое время можно удалить диск hdd1 из виртуальной машины MyVM командой:

```
# prlctl set MyVM --device-del hdd1
Remove the hdd1 device.
```

Примечания:

1. Виртуальные диски IDE и SCSI можно добавлять в виртуальные машины или удалять из них, только если виртуальные машины остановлены.

2. Добавленный новый диск необходимо инициализировать перед использованием. Для инициализации диска можно использовать стандартные средства, предоставляемые гостевой операционной системой.

Управление ресурсами и пропускной способностью сети

В данном разделе объясняется, как выполнить следующие задачи в ПК Р-Виртуализация:

- настройка сетевых классов
- просмотр статистики сетевого трафика
- включение и выключение управления пропускной способностью сети
- настройка лимитов пропускной способности сети

Параметры сетевого трафика

В таблице ниже приведены параметры сетевого трафика, которые можно настроить в ПК Р-Виртуализация.

Параметр	Описание
`traffic_shaping`	Если указано значение `yes`, то установлены ограничения на исходящий трафик для виртуальных сред. Значение по умолчанию `no`.
`bandwidth`	Данный параметр выводит список всех сетевых адаптеров, установленных на физическом сервере, и их пропускную способность.
`totalrate`	Данный параметр определяет пропускную способность, которую можно выделить для каждого сетевого класса. Он активен, если включено управление трафиком.
`rate`	Если включено управление трафиком, данный параметр указывает гарантированную пропускную способность для виртуальных сред.
`ratebound`	Если для данного параметра указано значение `yes`, гарантированная пропускная способность (глобальный параметр скорости передачи данных) является также лимитом для виртуальной среды, и виртуальная среда не может брать пропускную способность из пула пропускной способности `totalrate`:

Настройка сетевых классов

ПК Р-Виртуализация позволяет отслеживать входящий и исходящий сетевой трафик, а также управлять исходящим сетевым трафиком для виртуальных сред. Чтобы иметь возможность различать типы трафика, например, местный и международный, вводится понятие сетевых классов. Сетевой класс представляет собой диапазон IP-адресов, для которых ПК Р-Виртуализация ведет учет и управление трафиком.

Классы указаны в файле `/etc/vz/conf/networks_classes`. Файл имеет формат ASCII, и все пустые строки в нем, а также строки, начинающиеся со знака #, игнорируются. Остальные строки имеют следующий формат:

```
<class_id> <IP_address>/<prefix_length>
```

где <*class_id*> указывает ID сетевого класса, а пара <*IP_address*>/<*prefix_length*> определяет диапазон IP-адресов для данного класса. Для каждого класса может быть несколько строк.

Классы 0 и 1 имеют особые значения:

- Класс 0 определяет диапазон IP-адресов, для которых не выполняется учет. Обычно он соответствует подсети физического сервера (самому серверу и виртуальным средам на нем). Настройка класса 0 не требуется; однако его правильная настройка повышает производительность.
- Класс 1 определяется ПК Р-Виртуализация, чтобы соответствовать любому IP-адресу. Он должен всегда присутствовать в файле сетевых классов. Таким образом, не рекомендуется изменять строку по умолчанию в файле `networks_classes`:

```
1 0.0.0.0/0
```

  Если виртуальные среды используют IPv6-адреса, можно также добавить в файл следующую строку:

```
1 ::/0
```

После класса 1 необходимо определить остальные классы. Они представляют собой исключения из класса 1, который соответствует любому IP-адресу. В примере ниже

показана возможная конфигурация файла сетевых классов, содержащая правила для IPv4 и IPv6-адресов:

```
# Hardware node networks
0 192.168.0.0/16
0 fe80::/64
# any IP address (all traffic)
1 0.0.0.0/0
1 ::/0
# class 2 - addresses for the "foreign" traffic
2 10.0.0.0/8
2 2001:db88::/64
# inside "foreign" network there
# is a hole belonging to "local" traffic
1 10.10.16.0/24
1 2001:db88:3333::/64
```

В данном примере IPv4-адреса в диапазоне от `192.168.0.0` до `192.168.255.255` и IPv6-адреса в диапазоне от `fe80::` до `fe80::ffff:ffff:ffff:ffff` считаются классом 0, и не выполняется учет для трафика от виртуальных сред, которым назначены данные адреса.

Класс 2 соответствует следующим IP-адресам:

- IPv4-адресам от `10.0.0.0` до `10.255.255.255`, за исключением адресов в поддиапазоне от `10.10.16.0` до `10.10.16.255`, которые считаются классом 1.
- IPv6-адресам от `2001:db88::` до `2001:db88::ffff:ffff:ffff:ffff`, за исключением адресов в поддиапазоне от `2001:db88:3333::` до `2001:db88:3333::ffff:ffff:ffff:ffff`, которые также считаются классом 1.

Все остальные IP-адреса (IPv4 и IPv6) относятся к классу 1.

Чтобы применить изменения, внесенные в файл `/etc/vz/conf/networks_classes`, необходимо перезапустить либо виртуальные среды, для которых были сделаны данные изменения, либо физический сервер, если изменения глобальные.

Просмотр статистики сетевого трафика

В ПК Р-Виртуализация можно посмотреть текущую статистику сетевого трафика для виртуальных сред с помощью утилиты `vznetstat`. Например:

```
# vznetstat
UUID         Net.Class     Input(bytes)  Input(pkts)  Output(bytes)  Output(pkts)
0            0             566093064     2800575      3120481        41736
47406484...  0             67489         155          8033           110
fbb30afa-... 0             9369          78           12692          71
```

Следует иметь в виду, что утилита `vznetstat` отображает статистику только для виртуальных сред, которые были хотя бы один раз запущены.

Утилита `vznetstat` показывает следующую информацию:

Колонка	Описание
UUID	UUID, назначенный виртуальной среде.
Net.Class	ID сетевого класса, для которого рассчитывается статистика.
Input(bytes)	Размер входящего трафика, в байтах.
Input(pkts)	Размер входящего трафика, в пакетах.
Output(bytes)	Размер исходящего трафика, в байтах.
Output(pkts)	Размер исходящего трафика, в пакетах.

Например, из вывода команды выше видно, что примерно 9 МБ данных было получено виртуальной средой с UUID `fbb30afa-e770-4081-9d9e-6b9c262eb091`, примерно 12 МБ было отправлено из данной среды, и обмен трафиком произошел между серверами из сетей класса 0.

При необходимости можно посмотреть статистику сетевого трафика для отдельной виртуальной среды, указав ее UUID после параметра `-v`, например:

```
# vznetstat -v fbb30afa-e770-4081-9d9e-6b9c262eb091
UUID         Net.Class  Input(bytes)  Input(pkts)  Output(bytes)  Output(pkts)
fbb30afa-... 0          9369          78           12692          71
```

Данная команда выводит статистику только для контейнера `MyCT`.

Настройка управления трафиком

Управление трафиком (также известное как управление пропускной способностью сети) позволяет контролировать пропускную способность сети, которую виртуальная среда может использовать для исходящего трафика. По умолчанию данная функция отключена.

Примечания:

1. Текущая версия ПК Р-Виртуализация не позволяет управлять трафиком внутри сервера, что включает трафик между виртуальными средами одного сервера, а также трафик между виртуальными средами и самим сервером.

2. Текущая версия ПК Р-Виртуализация не позволяет управлять входящим трафиком для виртуальных сред.

Управление трафиком в ПК Р-Виртуализация контролируется следующими параметрами:

- `TRAFFIC_SHAPING`, включает и отключает управление трафиком.
- `BANDWIDTH`, устанавливает пропускную способность для определенных сетевых адаптеров.
- `TOTALRATE`, устанавливает размер пула пропускной способности, разделенный между виртуальными средами на сервере.
- `RATEMPU`, ограничивает скорость передачи пакетов в дополнение к скорости передачи данных.

- RATE, устанавливает гарантированную пропускную способность для виртуальных сред.
- RATEBOUND, устанавливает параметр RATE в качестве лимита.

В ПК Р-Виртуализация управление трафиком работает следующим образом. Пул пропускной способности для данного сетевого класса (задается параметром TOTALRATE) разделяется между виртуальными средами, которые передают данные в соответствии со своими настройками RATE. Если сумма значений RATE всех виртуальных сред, передающих данные, не превышает значение TOTALRATE, каждая виртуальная среда получает пропускную способность, равную или больше своего значения RATE (если параметр RATEBOUND не включен для данной виртуальной среды). Если сумма значений RATE всех виртуальных сред, передающих данные, превышает значение TOTALRATE, то каждая виртуальная среда может получить меньше своего значения RATE.

Чтобы включить и настроить управление трафиком, необходимо выполнить следующие действия:

1. Указать значение yes для параметра TRAFFIC_SHAPING в глобальном конфигурационном файле /etc/vz/vz.conf.
2. Задать параметры BANDWIDTH, TOTALRATE в /etc/vz/vz.conf.
3. При необходимости задать необязательные параметры RATEMPU, RATE, RATEBOUND в /etc/vz/vz.conf.
4. При необходимости задать параметры RATE и RATEBOUND для определенных виртуальных сред с помощью команд prlctl set --rate и prlctl set --ratebound.
5. Для применения изменений нужно перезапустить либо виртуальные среды, для которых были сделаны данные изменения, либо физический сервер, если изменения глобальные.

В разделах ниже приводится дополнительная информация о параметрах управления трафиком.

Настройка параметра BANDWIDTH

Параметр BANDWIDTH используется для управлением трафиком отдельных сетевых адаптеров. Например, для двух Fast Ethernet-адаптеров типичная настройка может иметь вид enp0s5 enp0s6:100000, где enp0s5 и enp0s6 являются именами сетевых адаптеров. По умолчанию параметр имеет значение 100000, что соответствует Fast Ethernet-адаптеру в 100 Мбит/с.

Настройка параметра TOTALRATE

Параметр TOTALRATE указывает размер пула пропускной способности для определенного сетевого класса на сервере. Виртуальные среды могут занимать пропускную способность из данного пула для обмена данными с серверами из соответствующего сетевого класса.

Таким образом, параметр ограничивает общий доступный исходящий трафик для сетевого класса, который могут использовать виртуальные среды.

Параметр имеет формат: <*NIC*>:<*network_class*>:<*bandwidth_in_Kbps*>. Например, чтобы задать размер пула в 4 Мбит/с для сетевого класса 1 на Ethernet-адаптере `enp0s5`, нужно указать `enp0s5:1:4000` для параметра `TOTALRATE`. Несколько записей можно отделить пробелами, например, `enp0s5:1:4000 enp0s6:2:8000`.

Настройка параметра RATEMPU

Необязательный параметр `RATEMPU` (где "MPU" обозначает "minimum packet unit") ограничивает скорость передачи пакетов, обязывая пакеты меньшего размера, чем MPU, использовать маркеры HTB. С помощью данного параметра пакеты маленького размера могут считаться крупными и ограничиваться параметрами `TOTALRATE` и `RATE`. Максимальная скорость передачи пакетов в секунду можно приблизительно вычислить как `TOTALRATE` / `RATEMPU`.

Параметр имеет формат: <*NIC*>:<*network_class*>`[`:<*MPU_in_bytes_per_packet*>`]`. Если часть <*MPU_in_bytes_per_packet*> не указывается, то используется значение по умолчанию равное 1000 байтам. Несколько записей можно отделить пробелами, например, `enp0s5:1:2000 enp0s6:2:4000`. Чтобы задать параметр `RATEMPU` для всех известных Ethernet-устройств, нужно указать звездочку (`*`) для <*NIC*>. Например, чтобы задать минимальный размер пакетов в 2 КБ для сетевого класса 1 на всех Ethernet-адаптерах сервера, необходимо изменить значение параметра на `*:1:2000`.

Настройка параметров RATE и RATEBOUND

Необязательный параметр `RATE` позволяет гарантировано предоставлять виртуальным средам исходящую пропускную способность для адресатов в определенном сетевом классе на конкретном Ethernet-устройстве. Гарантированная пропускная способность не является лимитом (если для параметра `RATEBOUND` не указано значение `on`, см. ниже). Виртуальная среда может дополнительно получать неиспользуемую пропускную способность из пула пропускной способности, определенного параметром `TOTALRATE`.

Гарантированную пропускную способность можно задать двумя способами:

1. Для всех виртуальных сред на сервере путем настройки параметра `RATE` в глобальном конфигурационном файле `/etc/vz/vz.conf`.

   Данный параметр имеет формат: <*NIC*>:<*network_class*>:<*bandwidth_in_Kbps*>. Например, чтобы все виртуальные среды на сервере гарантировано получали пропускную способность от 8 Кбит/с для исходящего трафика в сетевом классе 1 на Ethernet-устройстве `enp0s5`, нужно указать `enp0s5:1:8` для параметра `RATE`.

2. Для конкретных виртуальных сред с помощью команды `prlctl set --rate`.

   Например, чтобы контейнер `MyCT` гарантировано получал пропускную способность от 16 Кбит/с для исходящего трафика в сетевом классе 1, нужно выполнить команду:

```
# prlctl set MyCT --rate 1:16
```

Данная команда задает пропускную способность только для сетевого адаптера по умолчанию. Если нужно задать пропускную способность для других сетевых адаптеров, следует указать значение параметра RATE в `/etc/vz/vz.conf`.

Примечание: Рекомендуется увеличивать значение параметра RATE с шагом в 8 Кбит/с и установить его как минимум 8 Кбит/с.

Необязательный параметр RATEBOUND указывает, является ли гарантированная пропускная способность, заданная параметром RATE, также лимитом. По умолчанию данная функция отключена для всех новых виртуальных сред, чтобы они могли дополнительно получать неиспользуемую пропускную способность из пула пропускной способности, определенного параметром TOTALRATE.

Ограничить пропускную способность виртуальных сред до гарантированного значения можно следующим образом:

1. Для всех виртуальных сред на сервере путем настройки параметра RATEBOUND в глобальном конфигурационном файле `/etc/vz/vz.conf` (по умолчанию не указывается).
2. Для конкретных виртуальных сред с помощью команды `prlctl set --ratebound`. Например:

```
# prlctl set MyCT --ratebound yes
```

Если заданы значения параметров RATE и RATEBOUND для конкретных виртуальных сред, то они применяются вместо глобальных значений в `/etc/vz/vz.conf`.

Пример управления трафиком

Пример ниже иллюстрирует сценарий, в котором для контейнеров MyCT1 и MyCT2 задано значение no для параметра RATEBOUND, а для виртуальной машины MyVM параметр RATEBOUND имеет значение yes. Со значением по умолчанию параметра TOTALRATE равным 4096 Кбит/с и параметра RATE равным 8 Кбит/с пул пропускной способности будет раздаваться следующим образом:

MyCT1	MyCT2	MyVM	Используемая пропускная способность
передает данные	бездействует	бездействует	MyCT1: 4096 Кбит/с
бездействует	бездействует	передает данные	MyVM: 8 Кбит/с
передает данные	передает данные	бездействует	MyCT1: 2048 Кбит/с MyCT2: 2048 Кбит/с
передает данные	бездействует	передает данные	MyCT1: 4032 Кбит/с MyVM: 8 Кбит/с
передает данные	передает данные	передает данные	MyCT1: 2016 Кбит/с MyCT2: 2016 Кбит/с MyVM: 8 Кбит/с

Управление параметрами дискового ввода-вывода

В данном разделе объясняется, как управлять параметрами дискового ввода-вывода (I/O) в системах ПК Р-Виртуализация.

Настройка уровней приоритета

ПК Р-Виртуализация позволяет настроить уровень приоритета дискового ввода-вывода для виртуальных сред. Чем выше уровень приоритета ввода-вывода, тем больше времени получит виртуальная среда на дисковые операции ввода-вывода по сравнению с другими виртуальными средами на физическом сервере. По умолчанию для любой виртуальной среды на сервере уровень приоритета ввода-вывода равен 4. Однако можно изменить текущий уровень приоритета ввода-вывода с помощью параметра `--ioprio` команды `prlctl set`. Например, чтобы задать приоритет ввода-вывода равный 6 для контейнера `MyCT` и виртуальной машины `MyVM`, нужно выполнить следующие команды:

```
# prlctl set MyCT --ioprio 6
# prlctl set MyVM --ioprio 6
```

Проверить уровень приоритета ввода-вывода, примененный к контейнеру `MyCT` и виртуальной машине `MyVM`, можно с помощью команд:

- для контейнера `MyCT`:

```
# grep IOPRIO /etc/vz/conf/fbb30afa-e770-4081-9d9e-6b9c262eb091.conf
IOPRIO="6"
```

- для виртуальной машины `MyVM`:

```
# prlctl list MyVM --info | grep ioprio
  cpu cpus=2 VT-x accl=high mode=32 ioprio=6 iolimit='0'
```

Настройка пропускной способности дискового ввода-вывода

В ПК Р-Виртуализация можно настроить пропускную способность, которую разрешается использовать виртуальным средам для операций дискового ввода-вывода. При помощи ограничения пропускной способности дискового ввода-вывода можно предотвратить ситуации, когда высокая активность диска в одной виртуальной среде (вызванная, например, переносом большого объема данных с данной виртуальной среды или на нее) может снизить производительность других виртуальных сред на физическом сервере.

По умолчанию ограничение пропускной способности ввода-вывода для всех новых виртуальных сред равно 0, это значит, что для виртуальных сред не установлено ограничения. Чтобы ограничить пропускную способность дискового ввода-вывода для виртуальной среды можно использовать параметр `--iolimit` команды `prlctl set`. Например, следующая команда ограничивает пропускную способность ввода-вывода для контейнера `MyCT` до 10 мегабайт в секунду (МБ/с):

```
# prlctl set MyCT --iolimit 10
```

По умолчанию ограничение указывается в мегабайтах в секунду. Однако можно использовать следующие суффиксы для обозначения других единиц измерения:

- `G` устанавливает ограничение в гигабайтах в секунду (`1G`).
- `K` устанавливает ограничение в килобайтах в секунду (`10K`).
- `B` устанавливает ограничение в байтах в секунду (`10B`).

Примечание: В текущей версии ПК Р-Виртуализация максимальное ограничение пропускной способности ввода-вывода, которое можно установить для виртуальной среды, равно 2 ГБ/с.

Проверить, что ограничение скорости ввода-вывода было установлено для контейнера `MyCT`, можно с помощью команды `prlctl list`:

```
# prlctl list MyCT -o iolimit
IOLIMIT
10485760
```

В любое время можно убрать ограничение пропускной способности ввода-вывода, установленное для контейнера `MyCT`, выполнив следующую команду:

```
# prlctl set MyCT --iolimit 0
```

Настройка количества операций ввода-вывода в секунду

В ПК Р-Виртуализация можно ограничить максимальное количество операций дискового ввода-вывода в секунду, которое разрешено выполнять виртуальным средам (также известное, как лимит IOPS). Рекомендуется настроить лимит IOPS для виртуальных сред с высокой активностью диска, чтобы они не влияли на производительность других виртуальных сред на физическом сервере.

Примечание: По умолчанию все операции ввода-вывода внутри контейнеров кэшируются, и при открытии файлов флаг прямого доступа (`O_DIRECT`) игнорируется, что значительно снижает количество IOPS, необходимых для нагрузки контейнера, и помогает избежать проблем с операциями ввода-вывода на сервере. Для получения инструкций о настройке учета флага `O_DIRECT` внутри контейнеров см. **Установка флага прямого доступа внутри контейнеров** ниже.

По умолчанию количество IOPS не ограничено для новых виртуальных сред. Для установки лимита IOPS можно использовать параметр `--iopslimit` команды `prlctl set`. Например, чтобы контейнер `MyCT` и виртуальная машина `MyVM` не могли выполнять более 100 операций дискового ввода-вывода в секунду, введите следующие команды:

```
# prlctl set MyCT --iopslimit 100
# prlctl set MyVM --iopslimit 100
```

Чтобы убедиться, что лимит IOPS был применен, используйте команду `prlctl list -i`. Например:

```
# prlctl list MyCT -i | grep iopslimit
<...> iopslimit=100
```

В любое время можно убрать установленные лимиты IOPS командами:

```
# prlctl set MyCT --iopslimit 0
# prlctl set MyVM --iopslimit 0
```

Установка флага прямого доступа внутри контейнеров

Можно настроить, чтобы флаг `O_DIRECT` учитывался внутри контейнеров с параметром `sysctl fs.odirect_enable`:

- Чтобы игнорировать флаг `O_DIRECT` внутри контейнера, нужно указать значение `0` для `fs.odirect_enable` в данном контейнере.
- Чтобы учитывать флаг `O_DIRECT` внутри контейнера, нужно указать значение `1` для `fs.odirect_enable` в данном контейнере.
- Чтобы контейнер использовал настройку физического сервера, нужно указать значение `2` для `fs.odirect_enable` в данном контейнере (значение по умолчанию). На физическом сервере `fs.odirect_enable` имеет значение `0` по умолчанию.

Примечание: Параметр `fs.odirect_enable` на сервере влияет только на учет флага `O_DIRECT` в контейнерах, а не на самом сервере, где флаг `O_DIRECT` всегда учитывается.

Просмотр статистики дискового ввода-вывода

В ПК Р-Виртуализация можно посмотреть статистику дискового ввода-вывода для всех процессов на сервере:

1. Запустите утилиту `vztop`.
2. Нажмите **F2** или **S** для переключения в меню **Setup**.
3. В колонке **Setup** выберите **Columns**.
4. В **Available Columns** выберите из параметры из перечисленных ниже для добавления к выводу (**Active Columns**):

Параметр	Описание	Колонка
`RBYTES`	Количество прочитанных байтов для процесса.	IO_RBYTES
`WBYTES`	Количество записанных байтов для процесса.	IO_WBYTES
`IO_READ_RATE`	Скорость чтения для процесса, в байтах в секунду.	DISK READ
`IO_WRITE_RATE`	Скорость записи для процесса, в байтах в секунду.	DISK WRITE
`IO_RATE`	Общая скорость операций ввода-вывода для процесса, в байтах в секунду.	DISK R/W
`IO_PRIORITY`	Приоритет ввода-вывода для процесса.	IO

   Чтобы добавить параметр, выделите его и нажмите **F5** или **Enter**. Чтобы удалить параметр из **Active Columns**, выделите его и нажмите **F9**.
5. После окончания настройки колонок нажмите **F10** для сохранения изменений и просмотра вывода.

Настройка лимитов ввода-вывода для операций резервного копирования и миграции

Операции резервного копирования и миграции с контейнерами и виртуальными машинами могут создать высокую загрузку ввода-вывода на сервере, что приведет к снижению производительности виртуальных сред или самого сервера. Подобной ситуации можно избежать, если для данных операций задать лимиты ввода-вывода.

Чтобы задать лимит ввода-вывода, выполните следующие действия:

1 В глобальном конфигурационном файле `/etc/vz/vz.conf` найдите раздел:

```
# VZ Tools limits
<...>
# Uncomment next line to specify required disk IO bandwidth in Bps (10485760 - 10MBps)
# VZ_TOOLS_IOLIMIT=10485760
```

2 Раскомментируйте параметр `VZ_TOOLS_IOLIMIT` и задайте лимит ввода-вывода для операций резервного копирования и миграции в байтах в секунду.

3 Сохраните файл.

При настройке лимитов ввода-вывода следует иметь в виду следующее:

- `VZ_TOOLS_IOLIMIT` является глобальным параметром, который задается для всех виртуальных сред на сервере.
- Параметр `VZ_TOOLS_IOLIMIT` контролирует загрузку ввода-вывода только для операций резервного копирования и миграции. Он не влияет на операции по восстановлению виртуальных сред из резервных копий.
- Заданный лимит не делится между несколькими одновременными операциями, а применяется отдельно для каждой операции.
- Для миграции применяется только лимит, заданный на исходном сервере, а лимит, установленный на целевом сервере, не учитывается.

Управление параметрами памяти контейнеров

В данном разделе описана система управления памятью VSwap, а в частности:

- настройка основных параметров VSwap для контейнеров,
- настройка лимита на выделение памяти в контейнерах,
- настройка OOM killer,
- повышение функциональности VSwap.

Настройка основных параметров VSwap

ПК Р-Виртуализация использует схему VSwap для управления в контейнерах параметрами, связанными с памятью. Как и многие другие схемы управления памятью, используемые на автономных компьютерах Linux, данная схема основана на двух главных параметрах:

- RAM определяет общий размер ОЗУ, который можно использовать процессам контейнера,
- swap определяет общий размер области подкачки, который можно использовать контейнеру для откачки памяти после превышения размера ОЗУ.

Схема управления памятью работает следующим образом:

1. Контейнеру задается определенные размеры ОЗУ и области подкачки, которые можно использовать процессам, запущенным в контейнере.
2. Когда контейнер превышает установленный для него лимит ОЗУ, начинается процесс подкачки. Процесс подкачки для контейнеров немного отличается от подкачки на автономном компьютере. Файл подкачки контейнера является виртуальным и, при наличии возможности, находится в ОЗУ сервера. Другими словами, когда начинается откачка для контейнера и на сервере освобождается достаточно памяти для хранения файла подкачки, файл подкачки помещается в ОЗУ сервера и хранится там, а не на жестком диске.
3. После превышения лимита на подкачку система запускает OOM Killer для данного контейнера.
4. OOM Killer выбирает один или несколько процессов, запущенных в данном контейнере, и принудительно завершает их.

По умолчанию данная схема управления памятью используется любым новым контейнером. Чтобы узнать размеры ОЗУ и области подкачки, заданные для контейнера, можно проверить значения параметров PHYSPAGES и SWAPPAGES в конфигурационном файле контейнера, например:

```
# grep PHYSPAGES /etc/vz/conf/26bc47f6-353f-444b-bc35-b634a88dbbcc.conf
PHYSPAGES="65536:65536"
# grep SWAPPAGES /etc/vz/conf/26bc47f6-353f-444b-bc35-b634a88dbbcc.conf
SWAPPAGES="65536"
```

В примере выше параметр PHYSPAGES для контейнера MyCT имеет значение 65536. Параметр PHYSPAGES показывает размер ОЗУ в 4-килобайтных страницах, таким образом, общий размер ОЗУ для контейнера MyCT составляет 256 МБ. Значение параметра SWAPPAGES также равно 256 МБ.

Для настройки размеров ОЗУ и области подкачки для контейнера MyCT, можно использовать параметры `--memsize` и `--swappages` команды `prlctl set`. Например, Чтобы задать для контейнера MyCT размеры ОЗУ и области подкачки в 1 ГБ и 512 МБ соответственно, можно выполнить следующую команду:

```
# prlctl set MyCT --memsize 1G --swappages 512M
```

Настройка гарантированной памяти контейнеров

Гарантированная память представляет собой процентное соотношение памяти контейнера, которое гарантированно доступно данному контейнеру.

Важно: Общая гарантированная память всех запущенных виртуальных сред на сервере не должна превышать размер физического ОЗУ сервера. Если запуск виртуальной среды с гарантированной памятью приведет к тому, что общая гарантированная память превысит размер физического ОЗУ на сервере, то данную виртуальную среду не получится запустить. Если установка гарантированной памяти для запущенной виртуальной среды приведет к тому, что общая гарантированная память превысит размер физического ОЗУ на сервере, то данную гарантированную память не получится установить.

По умолчанию для гарантированной памяти контейнеров установлено значение 0%. Для изменения значения можно использовать команду `prlctl set --memguarantee`. Например:

```
# prlctl set MyCT --memguarantee 80
```

Чтобы вернуть стандартную настройку введите команду:

```
# prlctl set MyCT --memguarantee auto
```

Настройка ограничения выделенной памяти для контейнеров

При запуске приложения в контейнере ему выделяется некоторый размер памяти для его нужд. Обычно выделенная память намного превышает память, требующуюся приложению для выполнения. Это может привести к ситуации, когда невозможно запустить приложение в контейнере, даже если он имеет достаточно свободной памяти. Для разрешения подобных ситуаций схема управления памятью VSwap включает новый параметр `vm_overcommit`. С использованием данного параметра можно настроить размер памяти, который можно выделить приложениям в контейнере, независимо от размеров ОЗУ и области подкачки, заданных контейнеру.

Размер памяти, который можно выделить приложениям контейнера, рассчитывается как сумма размеров ОЗУ и области подкачки, установленных для контейнера, умноженная на фактор оверкоммита по памяти. В стандартном (базовом) конфигурационном файле контейнере значение данного фактора равно 1.5. Например, если контейнер создан по стандартному конфигурационному файлу и ему задано 1 ГБ ОЗУ и 512 МБ подкачки, то ограничение выделенной памяти для контейнера будет 2304 МБ. Данное ограничение можно настроить и задать, например, значение 3 ГБ с помощью следующей команды:

```
# vzctl set MyCT --vm_overcommit 2 --save
```

Данная команды использует фактор 2 для увеличения ограничения выделенной памяти до 3 ГБ: (1 ГБ ОЗУ + 512 МБ подкачки) * 2 = 3 ГБ.

Теперь приложениям в контейнере `MyCT` можно выделить при необходимости до 3 ГБ памяти.

Настройка поведения OOM Killer для контейнеров

OOM killer выбирает процесс контейнера (или процессы) для завершения, основываясь на негодности (badness) процесса из `/proc/<pid>/oom_score`. Негодность вычисляется с помощью памяти процесса, общей памяти и корректировки негодности, а затем вставляется в диапазон от 0 до 1000. Каждое очко негодности представляет собой одну тысячную часть от всей памяти контейнера. Процесс, который будет завершен, имеет наибольшее значение негодности.

OOM killer можно настроить для процессов контейнера при помощи файла `/etc/vz/oom-groups.conf`, в котором перечислены все модели, на базе которых выбирается корректировка негодности для каждого запущенного процесса. Каждая модель занимает отдельную строку и содержит следующие колонки:

- <*command*>, маска для названия команды задачи;
- <*parent*>, маска для названия родительской задачи;
- <*oom_uid*>, фильтр идентификатора пользователя (UID) задачи:
  - если <`oom_uid`> равен -1, модель будет применяться к задачам с любыми UID,
  - если <*oom_uid*> равен или больше 0, модель будет применяться к задачам с UID, равными значению <*oom_uid*>,
  - если <*oom_uid*> меньше -1, модель будет применяться к задачам с UID, меньшими, чем негативное значение <*oom_uid*>);
- <*oom_score_adj*>, корректировка негодности. Как и очко негодности, каждое очко корректировки представляет собой одну тысячную часть от всей памяти контейнера. Отрицательные значения корректировки уменьшают значение негодности процесса. В случае нехватки памяти корректировка гарантирует, что процесс сможет занимать от <*oom_score_adj*> тысячных от памяти контейнера, если в запущенном контейнере есть процессы с большим значением негодности.

Примечание: Маски <*command*> и <*parent*> поддерживают метасимвольные суффиксы: звездочка соответствует любому суффиксу. Например, "foo" соответствует только "foo", "foo*" соответствует "foo" и "foobar".

Например, модель

```
sshd     init     -500     -100
```

означает, что в случае нехватки памяти процессу `sshd`, дочернему для `init`, будет гарантировано от 100 тысячных (т.е. от 10%) памяти контейнера, если его UID меньше -(-500) или просто 500, например, 499. В соответствии с соглашениями RHEL, UID от 1 до 499 обычно резервируются для использования системой, поэтому такое разграничение может быть удобным для приоритизации и сохранения системных процессов.

При вычислении негодности процесса OOM killer ищет в `/proc/vz/oom_score_adj` подходящую модель, основываясь на маски и фильтр UID задач. Поиск начинается с первой строки и заканчивается нахождением первой подходящей модели. Затем

применяется соответствующее значение корректировки для получения итоговой негодности процесса.

Данные из `/etc/vz/oom-groups.conf` перезагружаются и передаются в ядро при перезапуске. Чтобы перезагрузить и передать данные из конфигурационного файла вручную, можно использовать следующую команду:

```
# cat /etc/vz/oom-groups.conf > /proc/vz/oom_score_adj
```

Настройка VSwap

Схему управления VSwap можно расширить с помощью параметров UBC. Например, можно задать параметр `numproc` для настройки максимального количества процессов и потоков, которые может создать контейнер, или параметр `numfile`, чтобы указать количество файлов, которые можно открыть всеми процессами в контейнере.

Управление параметрами памяти виртуальных машин

В данном разделе описано, как настроить параметры памяти, доступные для виртуальных машин:

- размер памяти,
- размер видеопамяти,
- горячее подключение памяти,
- гарантированная память.

Настройка размера памяти виртуальных машин

Для увеличения или уменьшения размера памяти, который будет доступен для виртуальной машины, можно использовать параметр `--memsize` команды `prlctl set`. В примере ниже показано, как можно увеличить ОЗУ виртуальной машины `MyVM` с 1 ГБ до 2 ГБ и проверить, что новое значение успешно задано:

```
# prlctl list -i MyVM | grep memory
memory 1024Mb
# prlctl set MyVM --memsize 2048
Set the memsize parameter to 2048Mb
The VM has been successfully configured.
# prlctl list -i MyVM | grep memory
memory 2048Mb
```

Изменения сохраняются в конфигурационном файле виртуальной машины и применяются при запуске виртуальной машины. Если виртуальная машина запущена, ее необходимо перезапустить. Чтобы увеличить или уменьшить размер ОЗУ виртуальной машины без ее перезапуска, нужно включить функцию горячего подключения, как описано в подразделе **Включение горячего подключения памяти для виртуальных машин** (стр. 92).

Примечание: Значение, заданное с помощью `prlctl --memsize`, не сообщается внутри виртуальной машины как размер физической или другой памяти. Пользователю, вошедшему в гостевую ОС, будет отображаться размер физической памяти, который можно получить путем полного “сдувания” balloon (см. **MaxNumaSize** в подразделе **Включение горячего подключения памяти для виртуальных машин** (стр. 92)). Размер balloon также не сообщается внутри виртуальной машины. Однако, если balloon не полностью “сдут”, то часть сообщенной физической памяти всегда будет оказываться занятой (тем, что фактически является balloon).

Настройка размера видеопамяти виртуальных машин

Для настройки размера видеопамяти, доступной для видеокарты виртуальной машины, можно использовать параметр `--videosize` команды `prlctl set`. Чтобы увеличить размер видеопамяти с 32 МБ до 64 МБ для виртуальной машины `MyVM`, выполните следующую команду:

```
# prlctl set MyVM --videosize 64
```

Проверить, задано ли новое значение, можно с помощью команды:

```
# prlctl list -i MyVM | grep video
video 64Mb
```

Включение горячего подключения памяти для виртуальных машин

Горячее подключение памяти позволяет увеличивать или уменьшать размер ОЗУ виртуальной машины “на лету” без необходимости перезапуска виртуальной машины. Горячее подключение памяти реализовано в виде комбинации механизма ballooning и добавления/удаления виртуальных слотов DIMM.

Алгоритм работает следующим образом. При выполнении команды для увеличения размера памяти виртуальной машины до `RAM_size` (как описано в подразделе **Настройка размера памяти виртуальных машин** (стр. 91)) память сначала расширяется путем “сдувания” balloon виртуальной машины. Лимит на “сдувание” balloon `MaxNumaSize` автоматически вычисляется согласно следующей формуле:

```
MaxNumaSize = (RAM_size + 4GB) rounded up to a multiple of 4GB
```

Если полного сдувания balloon не достаточно для получения `RAM_size` (то есть, `RAM_size` превышает `MaxNumaSize`), тогда память еще расширяется с помощью добавления виртуальных слотов DIMM (вплоть до двойного размера `MaxNumaSize`) и `MaxNumaSize` устанавливается равным `RAM_size` (то есть, максимальный размер balloon также увеличивается). При выполнении команды для уменьшения размера памяти виртуальной машины память сжимается путем “надувания” balloon виртуальной машины. Добавленные виртуальные слоты DIMM остаются до перезапуска виртуальной машины. После перезапуска размер памяти виртуальной машины равен `RAM_size`.

Данная функция поддерживается только для виртуальных машин с ОЗУ от 1 ГБ и по умолчанию отключена. Чтобы включить ее для виртуальной машины (например, `MyVM`), необходимо выполнить следующие действия:

1 Убедиться, что виртуальная машина остановлена.

2 Выполнить команду:

```
# prlctl set MyVM --mem-hotplug on
```

3 Запустить виртуальную машину.

Теперь размер ОЗУ виртуальной машины можно увеличивать и уменьшать с помощью команды `prlctl set --memsize` без необходимости перезапуска виртуальной машины.

Настройка гарантированной памяти виртуальных машин

Гарантированная память представляет собой процентное соотношение памяти виртуальной машины, которое гарантированно доступно данной виртуальной машине.

Важно: Общая гарантированная память всех запущенных виртуальных сред на сервере не должна превышать размер физического ОЗУ сервера. Если запуск виртуальной среды с гарантированной памятью приведет к тому, что общая гарантированная память превысит размер физического ОЗУ на сервере, то данную виртуальную среду не получится запустить. Если установка гарантированной памяти для запущенной виртуальной среды приведет к тому, что общая гарантированная память превысит размер физического ОЗУ на сервере, то данная гарантированную память не получится установить.

По умолчанию для гарантированной памяти виртуальных машин установлено значение 40%. Для изменения значения используйте команду `prlctl set --memguarantee`. Например:

```
# prlctl set MyVM --memguarantee 80
```

Чтобы вернуть стандартную настройку нужно ввести команду:

```
# prlctl set MyVM --memguarantee auto
```

Примечание: Виртуальные машины с гарантированной памятью можно запустить только с помощью `prlctl start`. Запуск таких виртуальных машин другим способом (например, используя `virsh`) приведет к тому, что гарантированная память не будет применена.

Управление конфигурацией ресурсов контейнеров

Любой контейнер можно настроить при помощи его конфигурационного файла. Конфигурациями контейнера можно управлять различными способами:

1 Использовать файлы образцов конфигураций, которые поставляются вместе с ПК Р-Виртуализация. Данные файлы используются при создании нового контейнера (для получения дополнительной информации см. **Контейнеры** (стр. 11)). На текущий момент предоставляются следующие файлы образцов конфигурации:

- `basic` для создания стандартных контейнеров;

- `confixx` для создания контейнеров с панелью управления Confixx;
- `vswap.plesk` для создания контейнеров с панелью управления Plesk;
- `vswap.256MB` для создания контейнеров с 256 МБ оперативной памяти;
- `vswap.512Mb` для создания контейнеров с 512 МБ оперативной памяти;
- `vswap.1024Mb` для создания контейнеров с 1024 МБ оперативной памяти;
- `vswap.2048Mb` для создания контейнеров с 2048 МБ оперативной памяти.

Примечание: Имена файлов образцов конфигурации не могут содержать пробелы.

Любой файл образца конфигурации можно также применить к существующему контейнеру, например, для обновления или возврата к предыдущей версии конфигурации всех ресурсов определенного контейнера:

```
# prlctl set MyCT --applyconfig basic
```

Данная команда применяет все параметры из файла `ve-basic.conf-sample` к контейнеру `MyCT`. В процессе установки ПК Р-Виртуализация на физический сервер стандартные образцы конфигурации помещаются в директорию `/etc/vz/conf`. Они имеют следующий формат: `ve-`*<name>*`.conf-sample` (например, `ve-basic.conf-sample`).

2 Использовать специальные утилиты для подготовки конфигурационных файлов в их совокупности. Задачи, которые выполняют эти утилиты, описаны в подразделах данного раздела.

3 Напрямую создавать и редактировать конфигурационный файл соответствующего контейнера (`/etc/vz/conf/`*<UUID>*`.conf`). Данные способ осуществляется с помощью любого текстового редактора. Инструкции по редактированию конфигурационных файлов контейнеров представлены в четырех предыдущих разделах. В данном случае придется настраивать все параметры конфигурации отдельно.

Разделение сервера на равные части

При помощи команды `vzsplit` можно создать конфигурации для контейнеров, которые будут занимать определенную долю ресурсов физического сервера. Например, чтобы создать конфигурацию `myconf` для как максимум 20 контейнеров, выполните команду:

```
# vzsplit -n 20 -f myconf
Config /etc/vz/conf/ve-myconf.conf-sample was created
```

Конфигурация вычисляется на основе ресурсов физического сервера. Теперь можно использовать параметр `--config myconf` команды `prlctl create`, чтобы создать контейнер на базе данной конфигурации.

Применение новых образцов конфигурации к контейнерам

ПК Р-Виртуализация позволяет изменить файл образца конфигурации, на базе которого создан контейнер и тем самым одновременно изменить все ресурсы, которые может

потреблять контейнер. Например, если контейнер MyCT создан на базе образца конфигурации basic и в нем будет запущено приложение Plesk, то можно применить к нему образец vswap.plesk вместо basic, таким образом, автоматически настроив необходимые параметры ресурсов для запуска приложения Plesk в контейнере MyCT. Чтобы выполнить данное действие, введите на физическом сервере следующую команду:

```
# prlctl set MyCT --applyconfig vswap.plesk
```

Данная команда считывает параметры ресурсов из файла ve-vswap.plesk.conf-sample, который находится в директории /etc/vz/conf, и применяет их по очереди к контейнеру MyCT.

При применении новых образцов конфигурации к контейнерам следует иметь в виду следующее:

- Все файлы образцов конфигурации находятся в директории /etc/vz/conf на физическом сервере и их имена имеют следующий формат: ve-<*name*>.conf-sample. Необходимо указать только часть <*name*> соответствующего имени образца после параметра --applyconfig (vswap.plesk в примере выше).
- Параметр --applyconfig применяет все параметры из указанного файла образца к данному контейнеру, за исключением параметров OSTEMPLATE, TEMPLATES, VE_ROOT, VE_PRIVATE, HOSTNAME, IP_ADDRESS, TEMPLATE, NETIF (если они указаны в файле образца).

В зависимости от того, могут ли изменения для выбранных параметров применяться “на лету” или нет, необходимо будет перезапустить контейнер. Если некоторые параметры невозможно настроить в ходе работы, появится соответствующее сообщение об этом.

Управление конфигурацией ресурсов виртуальных машин

Конфигурация виртуальной машины определяется конфигурационным файлом config.pvs. Данный файл в формате XML автоматически генерируется при создании новой виртуальной машины и содержит все ее параметры: память, ЦП, дисковое пространство и др.

После создания виртуальной машины можно вручную настроить ее параметры с помощью утилиты prlctl. Однако если нужно настроить большое число параметров для нескольких виртуальных машин, это может оказаться трудоемкой задачей. Для облегчения работы можно создать образцы виртуальных машин и использовать их для быстрого и простого изменения конфигурации виртуальных машин. Можно еще больше упростить процесс конфигурации с помощью создания шаблона виртуальной машины и нескольких файлов образцов. В таком случае удастся быстро создать новую виртуальную машину на базе данного шаблона и применить к ней нужный файл конфигурации.

Создание образца конфигурации

Перед использованием образцов конфигурации для виртуальных машин необходимо создать по крайней мере один образец конфигурации. Самый простой способ создания образца конфигурации описан ниже:

1 Создайте конфигурацию виртуальной машины, например:

```
# prlctl create VmConfiguration
```

2 Задайте желаемые параметры для конфигурации виртуальной машины. Например, можно использовать команду `prlctl set`, чтобы задать нужный размер памяти и дискового пространства. Все параметры сохраняются в файл `config.pvs` виртуальной машины `VmConfiguration`.

   Для просмотра списка параметров, которые можно применить из образцов конфигурации, см. **Параметры, применяемые из образцов конфигурации** ниже.

3 Скопируйте файл `config.pvs` в директорию `/etc/Parallels/samples`. Если данная директория не существует, нужно ее создать:

```
# mkdir /etc/Parallels/samples
# cp /vz/vmprivate/<VM_UUID>/config.pvs /etc/parallels/samples/configMySQL.pvs
```

   Последняя команда копирует файл `config.pvs` в файл `configMyDB.pvs`.

Применение образцов конфигурации к виртуальным машинам

После создания образца конфигурации можно применить его к любой виртуальной машине, используя параметр `--applyconfig` команды `prlctl set` и указав имя образца без расширения `.pvs`. Например, чтобы применить образец `configMySQL` к виртуальной машине `VM1`, можно выполнить следующую команду:

```
# prlctl set VM1 --applyconfig configMySQL
```

Образцы конфигурации можно применять только к остановленным виртуальным машинам.

Параметры, применяемые из образцов конфигурации

Из нового образца конфигурации к виртуальной машине применяются следующие параметры:

- Все параметры, связанные с памятью (ОЗУ и видеопамятью). Для просмотра данных параметров в файле образца, нужно найти элементы `<Memory>` и `<Video>`.
- Все параметры, связанные с ЦП. Для просмотра данных параметров в файле образца, нужно найти элемент `<Cpu>`.
- Параметры ввода-вывода и IOPS. Для просмотра данных параметров в файле образца, нужно найти элементы `<IoLimit>` и `<IopsLimit>` соответственно.
- Параметры диска. Для просмотра данных параметров в файле образца, нужно найти элемент `<Size>`, заключенный в элемент `<Hdd>`:

```
<Hdd id=0" dyn_lists="Partition 0">
<Index>0</Index>
<Size>65536</Size>
</Hdd>
```

Виртуальный диск, к которому применяется значение элемента <Size>, определяется индексным номером в элементе <Index>. Например, в примере выше параметр диска (65536 МБ) применяется к виртуальному диску с индексным номером 0. Если в виртуальной машине нет виртуального диска с указанным индексом, то параметр игнорируется.

Мониторинг ресурсов

В ПК Р-Виртуализация можно использовать утилиту vztop для мониторинга системных ресурсов в реальном времени. Утилита отображает информацию о процессоре, использовании подкачки и памяти, а также количество задач, среднюю загрузку и время работы в верхней части экрана. Можно изменить стандартные счетчики, нажав **F2** или **S**. Например, чтобы посмотреть текущие системные ресурсы, можно выполнить следующую команду на физическом сервере:

```
# vztop
1  [                                          0.0%]  Tasks: 77, 65 thr; 1 running
2  [|||                                       2.6%]  Load average: 0.02 0.03 0.05
3  [||||                                      4.6%]  Uptime: 06:46:48
4  [|                                         0.7%]
Mem[||||||||||||||||||||                 344M/3.68G]
Swp[                                      0K/3.87G]
```

Цифры в левой части представляют собой количество ЦП/ядер в системе. Шкала прогресса показывает их загрузку и может состоять из нескольких цветов. По умолчанию шкала ЦП отображается в четырех цветах:

- синий – процессы с низким приоритетом,
- зеленый – процессы (пользователя) с нормальным приоритетом,
- красный – процессы ядра,
- бирюзовый – время виртуализации.

Шкала памяти состоит из трех цветов:

- зеленый – используемые страницы памяти,
- синий – страницы буфера,
- желтый/оранжевый – страницы кэша.

Шкала подкачки содержит только один цвет—красный—который обозначает используемую область подкачки.

Вывод команды обновляется через временные интервалы, которые можно задать с помощью параметра -d в десятых долях секунды. Если параметр -d не указан, то интервал по умолчанию равен 1 секунде (т.е. -d 10).

Глава 4

Управление службами и процессами

В данной главе объясняется, что представляют собой службы и процессы, как они влияют на работу и производительность системы и какие задачи они выполняют в системе.

В этой главе можно узнать, как использовать утилиты командной строки для управления службами и процессами в ПК Р-Виртуализация. В частности, можно узнать, как посмотреть активные процессы в системе, изменить режим `xinetd`-зависимых служб, идентифицировать UUID контейнера, в котором запущен процесс, по ID процесса, как запускать, останавливать или перезапускать службы и процессы, а также как редактировать уровни запуска служб.

Что представляют собой службы и процессы

Процессами называются экземпляры программ, в текущий момент запущенные в системе. Процесс может рассматриваться как виртуальное адресное пространство и данные по управлению, необходимые для выполнения программы. Типичным примером процесса является приложение *vi*, запущенное на сервере или в контейнерах на базе Linux. Наряду с обычными процессами, существует большое число процессов, которые предоставляют интерфейс для вызова других процессов. Такие процессы называются службами. Службы выступают в роли “мозга” для многих ключевых системных процессов. Обычно большую часть времени они проводят в ожидании события или периода, когда у них запланирована какая-либо задача. Многие службы обеспечивают другим серверам в сети возможность соединения с данным сервером, используя различные сетевые протоколы. Например, служба `nfs` обеспечивает NFS-сервер функциональностью, позволяющей общий доступ к файлам в TCP/IP-сетях.

В связи с процессами и службами также часто можно встретить термин "демон". Данный термин относится к программному продукту, который используется для выполнения определенной функции в системе, а также используется в качестве синонима для службы. Демона можно легко идентифицировать, так как он имеет d на конце своего имени. Например, `httpd` (HTTP-демон) обозначает программу, запущенную на фоне системы и ожидающую входящих запросов к веб-серверу. Демон автоматически отвечает на запросы и обслуживает гипертекстовые и мультимедийные документы в сети Интернет, используя HTTP.

При работе со службами следует иметь в виду, что в течение своего жизненного цикла служба потребляет много системных ресурсов. Она использует ЦП системы для запуска инструкций и физическую память системы для себя и своих данных. Она открывает и использует файлы внутри файловых систем и может использовать некоторые физические устройства в системе. Таким образом, для лучшей производительности системы на

сервере должны быть запущены только те службы, которые необходимы в данный момент.

Также следует помнить, что запускать службы в ОС сервера намного опасней, чем в виртуальных машинах и контейнерах. Если через запущенную службу злоумышленники получат доступ к одной из виртуальных сред, они смогут навредить только той виртуальной среде, на которой запущена служба, а остальные виртуальные машины и контейнеры на сервере будут в безопасности. Физический сервер также останется нетронутым. Если служба будет запущена на сервере, то вред будет причинен физическому серверу и всем виртуальным средам, находящимся на нем. Поэтому для повышения безопасности системы рекомендуется запускать на физическом сервере только те службы, которые необходимы для его исправной работы, а дополнительные службы запускать в отдельных виртуальных средах.

Основные операции с процессами и службами

Возможность мониторинга и управления процессами и службами в системе очень важна, так как они оказывают сильное влияние на работу и производительность всей системы. Чем больше информации доступно о каждом процессе или службе, тем легче будет обнаружить и устранить проблемы при их возникновении.

Наиболее употребительными задачами, связанными с управлением службами, являются запуск, остановка, включение и отключение службы. Например, необходимо запустить службу, чтобы использовать определенные серверные приложения, или остановить либо поставить на паузу службу для выполнения тестирования или обнаружения проблемы.

Вместо запуска и остановки `xinetd`-зависимые службы включают и отключают. Службы, включенные таким образом, запускаются и останавливаются в соответствии с состоянием `xinetd`-демона. Отключенные службы не запускаются, независимо от состояния `xinetd`.

В ПК Р-Виртуализация можно управлять службами на физическом сервере или внутри контейнеров при помощи специальных утилит командной строки Linux локально или с любого сервера, подключенного к сети.

Для процессов в ПК Р-Виртуализация есть такие утилиты, как `vzps`, `vztop`, `vzpid`, которые позволяют посмотреть, что делает процесс, и управлять им. Использование данных утилит может помочь в обнаружении проблем, приводящих к медлительности и нестабильности системы. Также следует упомянуть, что в ПК Р-Виртуализация можно выполнять все операции с процессами, которые доступны в нормальной системе, например, завершить процесс с помощью сигнала завершения.

Управление процессами и службами

В ПК Р-Виртуализация службами и процессами можно управлять посредством следующих утилит командной строки:

- `vzps`
- `vzpid`
- `vztop`

С их помощью можно выполнять такие задачи, как:

- выведение информации об активных процессах на физическом сервере,
- просмотр активности процессов в реальном времени,
- изменение режима служб (службы могут быть `xinetd`-зависимые или автономные),
- идентификация UUID контейнера, в котором запущен процесс, по ID процесса.

Примечание: Максимальное число процессов для каждого контейнера ограничено до 131,072.

Просмотр активных процессов и служб

Утилита `vzps` обеспечивает дополнительную функциональность, связанную с мониторингом отдельных контейнеров, запущенных на физическом сервере. Например, можно использовать параметр `-E` для:

- отображения UUID контейнеров, в которых запущены процессы
- просмотра процессов, запущенных в определенном контейнере

Утилита `vzps` выводит информацию об активных процессах на физическом сервере. Если параметры не указаны, то `vzps` показывает в виде списка только те процессы, которые запущены в данном терминале. Ниже приводится пример вывода утилиты `vzps`:

```
# vzps
  PID TTY          TIME CMD
 4684 pts/1    00:00:00 bash
27107 pts/1    00:00:00 vzps
```

На текущий момент единственные процессы единственными процессами для данного пользователя/терминала являются `bash`-оболочка и сама команда `vzps`. В выводе информация о процессах представлена в виде таблицы с колонками: **PID**, **TTY**, **TIME** и **CMD**. **PID** обозначает ID процесса, **TTY**—терминал, в котором запущен процесс, **TIME** показывает, сколько времени ЦП использовал процесс, а **CMD**—имя команды, которая запустила процесс.

Примечание: ID процессов, которые запущены в контейнерах и отображаются в выводе команды `vzps`, выполненной на физическом сервере, не совпадают с ID тех же процессов, которые показаны в выводе команды `ps`, выполненной внутри контейнеров.

Как можно видеть, стандартная команда `vzps` отображает только основную информацию. Чтобы получить дополнительные подробности о процессах, запущенных на сервере, к команде `vzps` нужно добавить некоторые аргументы командной строки. Например, использование аргументов `aux` с командой показывает процессы, запущенные другими пользователями (a), процессы без терминала или других терминалов (x), пользователя, запустившего процесс, и время запуска (u).

```
# vzps aux
USER  PID %CPU %MEM   VSZ  RSS TTY   STAT START  TIME COMMAND
root    1  0.0  0.0  1516  128 ?     S   Jul14   0:37 init
root    5  0.0  0.0     0    0 ?     S   Jul14   0:03 [ubstatd]
root    6  0.0  0.0     0    0 ?     S   Jul14   3:20 [kswapd]
#27     7  0.0  0.0     0    0 ?     S   Jul14   0:00 [bdflush]
root    9  0.0  0.0     0    0 ?     S   Jul14   0:00 [kinoded]
root 1574  0.0  0.1   218  140 pts/4 S   09:30   0:00 -bash
```

Вывод содержит теперь больше информации о процессах. В таблицу добавлены колонки **USER**, **%CPU**, **%MEM**, **VSZ**, **RSS**, **STAT** и **START**. Информация, которую они содержат, описана ниже.

Колонка **USER** показывает пользователя, который запустил команду. Если процесс был запущен при загрузке системы, то в колонке в качестве пользователя отобразится `root` или некоторая системная учетная запись.

В колонках **%CPU**, **%MEM**, **VSZ** и **RSS** показана информация об использовании системных ресурсов. В колонках **%CPU** и **%MEM** можно посмотреть, какое процентное соотношение ресурсов ЦП и памяти соответственно использует процесс в данный момент. В колонке **VSZ** (virtual memory size) показан объем памяти, который заняла бы программа, если бы находилась полностью в памяти. А в колонке **RSS** (resident set size) показан действительный объем, находящийся в данный момент в памяти. Информация о текущем потреблении ресурсов процессом помогает определить, является ли поведение процесса нормальным или он вышел из-под контроля.

В колонке **TTY** можно увидеть много вопросительных знаков. Причиной этого является запуск данных программ при загрузке системы и/или при помощи скриптов инициализации. Для данных процессов не существует управляющего терминала; поэтому в выводе отображается вопросительный знак. С другой стороны, команда `bash` имеет значение **TTY** `pts/4`. Это команда, запущенная с удаленного соединения, и имеет связанный с ней терминал. Данная информация может быть полезна, если имеется больше одного открытого соединения к машине и нужно определить, в каком окне была запущена команда.

В колонке **STAT** показан текущий статус процесса. В примере выше все процессы спят, что обозначается буквой `S`. Спящие процессы означают, что они находятся в ожидании чего-либо, например, ввода пользователя или доступности системных ресурсов. Другим распространенным статусом является `R`, обозначающем, что процесс запущен.

Команду `vzps` также можно использовать для просмотра процессов внутри запущенного контейнера. В примере ниже показаны все активные процессы контейнера `MyCT` с UUID `26bc47f6-353f-444b-bc35-b634a88dbbcc`:

```
# vzps -E 26bc47f6-353f-444b-bc35-b634a88dbbcc
                                  CTID     PID TTY          TIME CMD
26bc47f6-353f-444b-bc35-b634a88dbbcc   14663 ?        00:00:00 init
26bc47f6-353f-444b-bc35-b634a88dbbcc   14675 ?        00:00:00 kthreadd/26bc47
26bc47f6-353f-444b-bc35-b634a88dbbcc   14676 ?        00:00:00 khelper
26bc47f6-353f-444b-bc35-b634a88dbbcc   14797 ?        00:00:00 udevd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15048 ?        00:00:00 rsyslogd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15080 ?        00:00:00 sshd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15088 ?        00:00:00 xinetd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15097 ?        00:00:00 saslauthd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15098 ?        00:00:00 saslauthd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15116 ?        00:00:00 sendmail
26bc47f6-353f-444b-bc35-b634a88dbbcc   15125 ?        00:00:00 sendmail
26bc47f6-353f-444b-bc35-b634a88dbbcc   15134 ?        00:00:00 httpd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15139 ?        00:00:00 httpd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15144 ?        00:00:00 crond
26bc47f6-353f-444b-bc35-b634a88dbbcc   15151 ?        00:00:00 mingetty
26bc47f6-353f-444b-bc35-b634a88dbbcc   15152 ?        00:00:00 mingetty
```

Мониторинг процессов в реальном времени

Утилита `vztop` похожа на `vzps`, но ее вывод занимает весь экран, а информация о процессах постоянно обновляется. Данная утилита может пригодиться для программ, которые нечасто вызывают проблемы и которые сложно обнаружить с помощью `vzps`.

Утилиту `vztop` можно использовать так же, как стандартную утилиту Linux `htop`. Она показывает динамический список всех процессов, запущенных в системе вместе с их полными командными строками. В выводе `vztop` также отображается информация по всей системе.

По умолчанию в верхней части экрана показана информация об использовании ресурсов процессора, подкачки и памяти, количество задач, среднее значение загрузки и время работы системы. Можно изменить стандартные счетчики, а также параметры отображения, цветовые схемы и колонки на экране настроек (**S** или **F2**).

`vztop` можно использовать интерактивно для отправки сигналов процессам. Например, можно завершить процессы с неизвестными PID путем их выделения и нажатия **F9**. Также можно менять приоритет процессов клавишами **F7** (повышение; может выполняться только пользователем `root`) и **F8** (снижение).

Вывод утилиты `vztop` имеет следующий вид:

```
# vztop
1  [                                          0.0%]  Tasks: 77, 65 thr; 1 running
2  [|||                                       2.6%]  Load average: 0.02 0.03 0.05
3  [||||                                      4.6%]  Uptime: 06:46:48
4  [|                                         0.7%]
Mem[||||||||||||||||||||                 344M/3.68G]
Swp[                                      0K/3.87G]

PID CTID USER  PRI NI VIRT   RES   SHR S CPU% MEM%  TIME+  Command
1     0  root  20  0 41620  4132  2368 S 0.0  0.1  0:05.91 /usr/lib/systemd/systemd
3164  0  root  20  0 19980  1380  1160 S 0.0  0.0  0:00.32 /usr/1ib/systemd/systemd-
3163  0  root  21  1 1402M 56992 10204 S 0.0  1.5  4:12.41 /usr/libexec/qemu-kvm -na
3186  0  root  20  0 1402M 56992 10204 S 0.0  1.5  0:00.09 /usr/libexec/qemu-kvm -na
```

```
3185  0  root  20  0 1402M 56992 10204 S 0.7  1.5  2:16.83 /usr/libexec/qemu-kvm -na
3180  0  root  20  0 1402M 56992 10204 S 0.0  1.5  0:00.00 /usr/libexec/qemu-kvm -na
3084  0  smmsp 20  0 85712  2036   516 S 0.0  0.1  0:00.19 sendmail: Queue runner@01
3064  0  root  20  0   98M  2380   572 S 0.0  0.1  0:01.43 sendmail: accepting conne
3036  0  root  20  0  291M  4788  3580 S 0.0  0.1  0:00.00 /usr/sbin/virt1ogd
3037  0  root  20  0  291M  4788  3580 S 0.0  0.1  0:00.00 /usr/sbin/virt1ogd
2787  0  nobody20  0 15548   896   704 S 0.0  0.0  0:00.14 /sbin/dnsmasq --conf-file
2788  0  root  20  0 15520   184     0 S 0.0  0.0  0:00.00 /sbin/dnsmasq --conf-file
2479  0  root  20  0 1962M 33344 24160 S 0.7  0.9  3:13.12 /usr/sbin/pr1_disp_servic
9022  0  root  20  0 1962M 33344 24160 S 0.0  0.9  0:10.74 /usr/sbin/pr1_disp_servic
```

В колонке **CTID** отображается UUID контейнера, в котором запущен процесс (значение `0` указывает, что процесс запущен на сервере). Под **PRI** (**PRIORITY**) показан внутренний приоритет ядра для процесса, а под **NI** (**NICE**)—приоритет nice (чем больше значение, тем больше процесс позволяет другим процессам иметь приоритета).

Посмотреть взаимосвязь процессов можно, переключив отображение на иерархическую структуру клавишей **F5**.

Определение UUID контейнеров по ID процессов

Каждый процесс идентифицируется уникальным PID (идентификатором процесса), который является записью этого процесса в таблице процессов ядра. Например, при запуске Apache назначается ID процесса. Данный PID потом используется для мониторинга и управления программой. Значение PID всегда является положительным целым числом. В ПК Р-Виртуализация можно использовать утилиту `vzpid` для вывода UUID контейнера, к которому принадлежит процесс с данным идентификатором. В качестве аргументов можно указать несколько ID процессов. В этом случае утилита выведет UUID контейнера для каждого процесса.

Вывод утилиты `vzpid` имеет следующий вид:

```
# vzpid 14663
Pid               VEID    Name
14663     26bc47f6-...    init
```

В примере выше процесс с идентификатором `14663` имеет имя `init` и запущен в контейнере с UUID `26bc47f6-{skipped}`.

Примечание: UUID контейнера, в котором запущен процесс, также можно узнать при помощи утилиты `vzps`.

Глава 5

Управление сетью

В данной главе дается представление о структуре сети в ПК Р-Виртуализация, перечисляются сетевые компоненты, а также объясняется, как управлять этими компонентами в рабочих средах. В частности, данная глава содержит информацию по:

- управлению сетевыми адаптерами на физическом сервере,
- виртуальным сетям и управлению ими на физическом сервере,
- созданию виртуальных сетевых адаптеров внутри виртуальных сред и настройке их параметров,
- подключению виртуальных сред к различным сетям.

Управление сетевыми адаптерами на физическом сервере

Сетевые адаптеры, установленные на физическом сервере, используются для обеспечения виртуальных машин и контейнеров доступом друг к другу и к внешним сетям. В процессе установки ПК Р-Виртуализация регистрирует все физические сетевые адаптеры, доступные на сервере. После установки ПК Р-Виртуализация можно управлять сетевыми адаптерами на физическом сервере при помощи собственных утилит RHEL7. Вы также можете создавать агрегации сетевых интерфейсов и интерфейсы VLAN (например, с помощью инструмента `nmtui`) и использовать их вместо физических адапторов.

Важно:

1. Каждый сетевой адаптер должен иметь только один конфигурационный файл в директории `/etc/sysconfig/network-scripts/`.

2. Чтобы применить изменения для сетевых настроек, следует использовать `systemctl restart NetworkManager.service` вместо `systemctl restart network`. При использовании второй команды могут возникнуть проблемы с сетью.

Сетевые режимы в ПК Р-Виртуализация

В данном разделе описаны сетевые режимы, доступные в ПК Р-Виртуализация.

В ПК Р-Виртуализация любая виртуальная среда может работать в одном из двух сетевых режимов: host-routed (режим маршрутизатора) или bridged (режим моста).

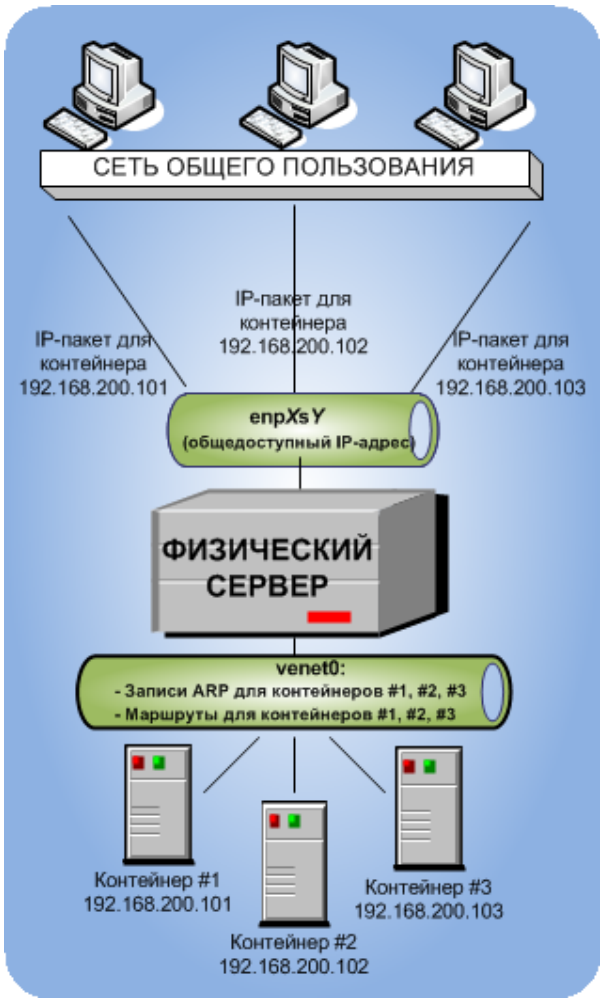
Сетевые режимы контейнеров

В данном разделе описаны сетевые режимы bridged и host-routed для контейнеров.

Примечание: В контейнерах поддерживаются IPSec-соединения.

Режим Host-Routed для контейнеров

По умолчанию новый контейнер настроен в режиме host-routed. В данном режиме контейнер использует специальный сетевой адаптер `venet0` для обмена данными с сервером, на котором он находится, с остальными контейнерами на сервере и с компьютерами из внешних сетей. На схеме ниже изображен пример конфигурации сети, в которой все контейнеры настроены в режиме host-routed.



В данной конфигурации:

- Контейнеры #1, #2 и #3 используют адаптер `venet0` в качестве шлюза по умолчанию для обмена данными с другими сетями. Они также используют данный адаптер для обмена трафиком между собой.
- При запуске контейнеров #1, #2 и #3 сервер для них создает записи ARP и маршруты в ARP-таблице и таблице маршрутизации. Посмотреть текущие записи ARP и маршруты на сервере можно с помощью команд `arp -n` и `route -n`. Например:

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask      Iface
10.30.0.4                ether   00:1a:e2:c7:17:c1   C               enp0s5
10.30.23.162             ether   70:71:bc:42:f6:a0   C               enp0s5
192.168.200.101          *       *                   MP              enp0s5
192.168.200.102          *       *                   MP              enp0s5
192.168.200.103          *       *                   MP              enp0s5
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.200.101 *               255.255.255.255 UH    1000   0        0 venet0
192.168.200.102 *               255.255.255.255 UH    1000   0        0 venet0
192.168.200.103 *               255.255.255.255 UH    1000   0        0 venet0
10.30.0.0       *               255.255.0.0     U     0      0        0 enp0s5
default         rosplatforma.ru 0.0.0.0         UG    0      0        0 enp0s5
```

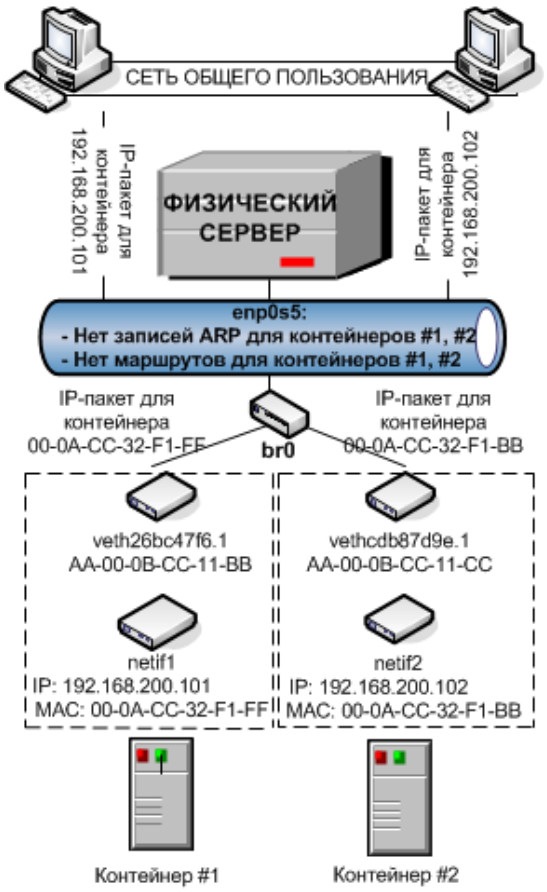
  Из вывода команд видно, что ARP-таблица и таблица маршрутизации содержат записи об IP-адресах 192.168.200.101, 192.168.200.102 и 192.168.200.103, принадлежащим контейнерам #1, #2 и 3# соответственно.

- Весь исходящий сетевой трафик контейнеров идет на адаптер `venet0` и передается через физический адаптер `enp0s5` адресату в соответствии с таблицей маршрутизации сервера.
- Весь входящий сетевой трафик контейнеров также обрабатывается адаптером `venet0`. Можно рассмотреть следующую ситуацию:

  1. **Компьютеру X** в локальной сети нужно отправить пакет данных контейнеру #1 с IP-адресом 192.168.200.101, поэтому он рассылает ARP-запрос, какой компьютер имеет данный IP-адрес.
  2. Сервер, на котором находится контейнер #1, отправляет ответ на запрос со своим MAC адресом.
  3. **Компьютер X** отправляет пакет данных на указанный MAC адрес.
  4. Сервер получает пакет и передает его на `venet0`, который перенаправляет его контейнеру #1.

Режим Bridged для контейнеров

Стандартный сетевой адаптер контейнера может работать только в режиме host-routed. Однако в контейнерах можно создать дополнительные виртуальные адаптеры и настроить их в режиме bridged. На схеме ниже изображен пример сетевой конфигурации, где контейнеры #1 и #2 работают в режиме bridged.



В данной конфигурации:

- Контейнеры #1 и #2 имеют отдельные виртуальные адаптеры, которые состоят из двух сетевых интерфейсов:
  - Интерфейс `netif<X>` в контейнере (**netif1** и **netif2** на рисунке). Данный интерфейс представляет собой копию физического сетевого адаптера, установленного на автономном сервере. Как и любой физический адаптер, он имеет MAC адрес, ему можно назначить один или несколько IP-адресов, включить его в разные сети и т.д.
  - Интерфейс `veth` на физическом сервере (**veth26bc47f6.1** и **vethcdb87d9e.1** на рисунке). Данный интерфейс используется в основном для поддержания обмена данными между физическим сервером и интерфейсами Ethernet в контейнерах.

  Примечание: Для упрощения виртуальные адаптеры, работающие в режиме bridged, называются адаптерами `veth`, хотя это и не совсем правильно, с технической точки зрения.

  Оба интерфейса тесно связаны друг с другом, таким образом, пакет данных, получаемый одним интерфейсом, всегда отправляется другим интерфейсом.
- Контейнеры #1 и #2 имеют свои ARP-таблицы и таблицы маршрутизации, в соответствии с которыми они отправляют и получают данные.
- Адаптеры `veth` обоих контейнеров соединены через мост `br0` с физическим сетевым адаптером `enp0s5`.

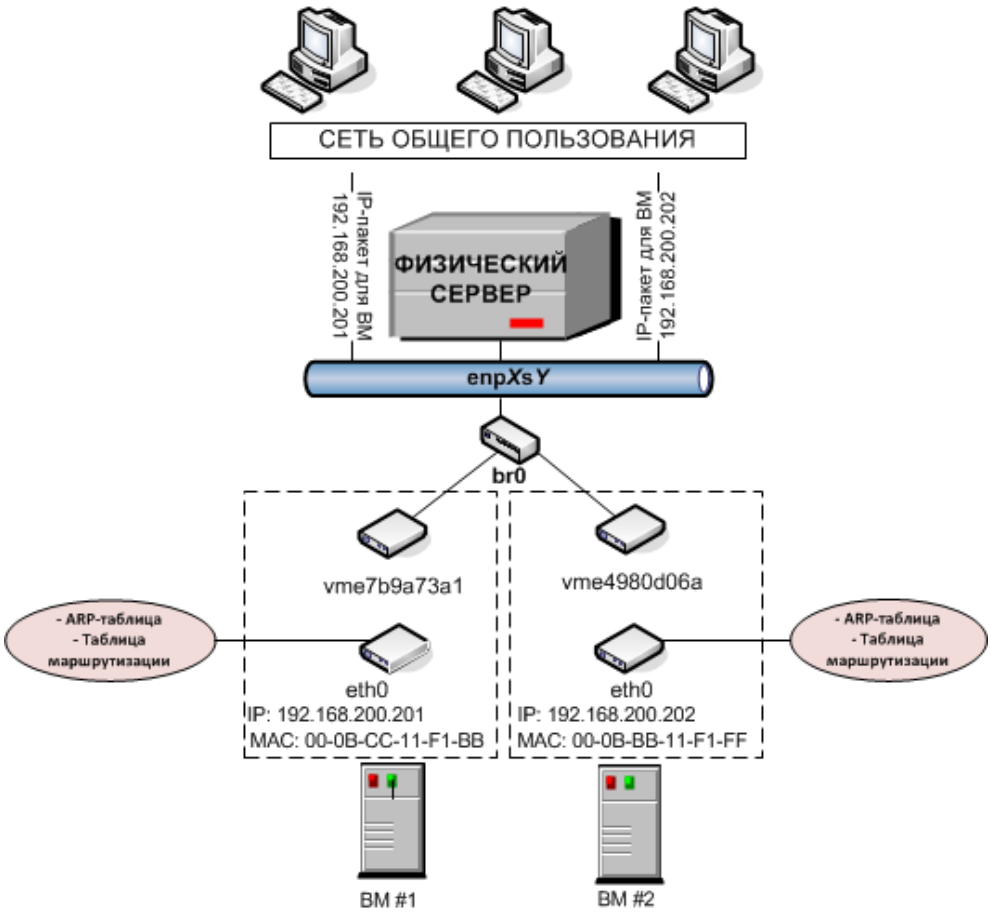
- Весь исходящий сетевой трафик контейнеров идет через адаптеры `veth` на мост и затем передается через физический адаптер `enp0s5` адресату в соответствии с таблицами маршрутизации, хранящимися в контейнерах.
- Все входящие пакеты данных для контейнеров #1 и #2 получаются сначала физическим адаптером `enp0s5`, а затем отправляются через мост на адаптер `veth` контейнера-адресата.

Сетевые режимы виртуальных машин

В данном разделе описаны сетевые режимы bridged и host-routed для виртуальных машин.

Режим Bridged для виртуальных машин

По умолчанию новая виртуальная машина создается с сетевым адаптером, работающем в режиме bridged. На схеме ниже изображен пример конфигурации сети, в которой две виртуальные машины ВМ #1 и ВМ #2 настроены в режиме bridged.



В данной конфигурации:

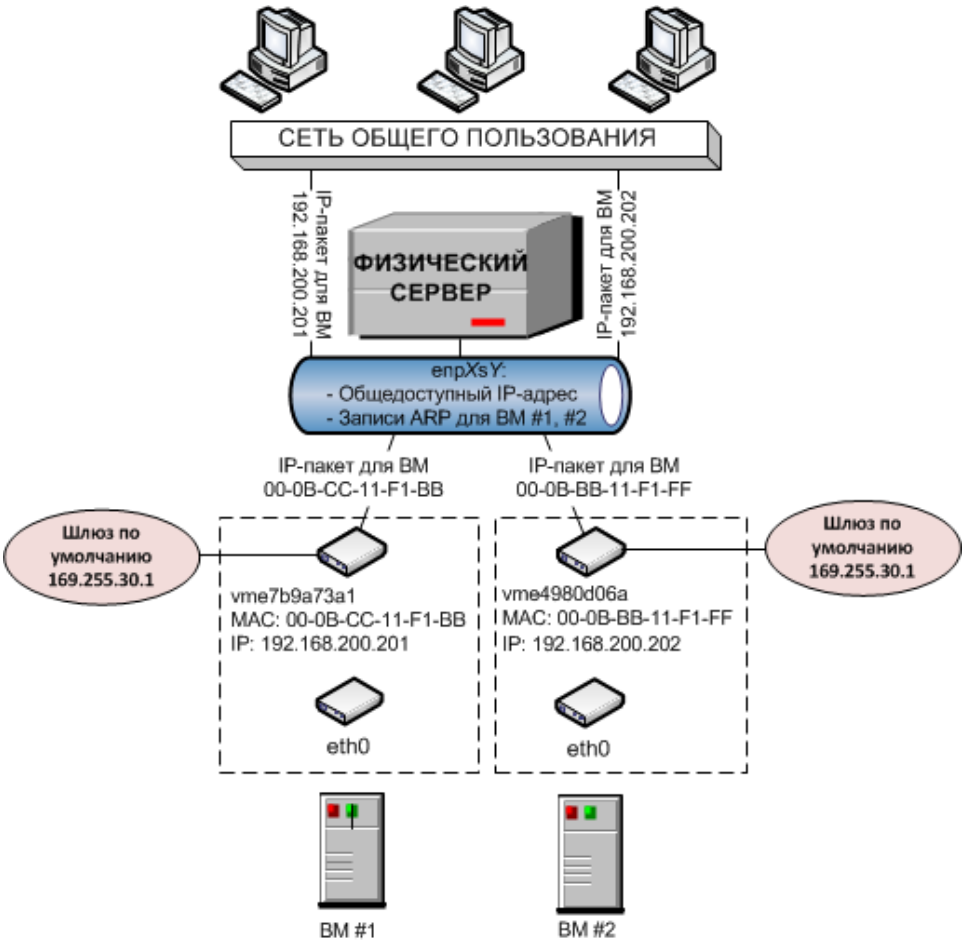
- Каждая виртуальная машина имеет отдельный виртуальный адаптер с двумя интерфейсами: интерфейс `ethX` в виртуальной машине (`eth0` на рисунке) и интерфейс `vme` на сервере (`vme7b9a73a1` и `vme4980d06a` на рисунке). Оба интерфейса тесно связаны друг с другом, таким образом, IP-пакет данных, получаемый одним интерфейсом, всегда отправляется другим интерфейсом. Адаптер `eth` имеет MAC адрес, ему можно назначить один или более IP-адресов, включить его в разные сетевые среды и т.д.

  Примечание: Для упрощения виртуальные адаптеры, работающие в режиме bridged, называются адаптерами `vme`, хотя это и не совсем правильно, с технической точки зрения.

- Виртуальные машины #1 и #2 имеют свои ARP-таблицы и таблицы маршрутизации, в соответствии с которыми они отправляют и получают данные.
- Виртуальные адаптеры обеих виртуальных машин соединены через мост `br0` с физическим сетевым адаптером `enp0s5`.
- Все исходящие пакеты данных отправляются из виртуальных машин через мост и физический адаптер `enp0s5` адресату в соответствии со своими таблицами маршрутизации.
- Все входящие пакеты данных для ВМ #1 и ВМ #2 получаются сначала физическим адаптером `enp0s5`, а затем отправляются через мост на адаптер `vme` ВМ-адресата.

Режим Host-Routed для виртуальных машин

Виртуальная машина также может работать в режиме host-routed. На схеме ниже изображен пример конфигурации сети, где две виртуальные машины ВМ #1 и ВМ #2 настроены в режиме host-routed.



В данной конфигурации:

- Каждая виртуальная машина имеет собственный виртуальный адаптер с двумя интерфейсами: интерфейс `ethX` в виртуальной машине (`eth0` на рисунке) и интерфейс `vme` на сервере (`vme7b9a73a1` и `vme4980d06a` на рисунке).
- В отличие от режима bridged, записи ARP для виртуальных машин #1 и #2 хранятся на сервере, а не в самих виртуальных машинах. При запуске ВМ #1 и ВМ #2 сервер создает данные записи ARP и сохраняет их в свою ARP-таблицу. Посмотреть текущие записи ARP на сервере можно с помощью команды `arp -n`, например:

```
# arp -n
Address              HWtype HWaddress           Flags Mask   Iface
10.30.0.4            ether  00:1a:e2:c7:17:c1   C            eth0
10.30.23.162         ether  70:71:bc:42:f6:a0   C            eth0
192.168.200.201      *      *                   MP           eth0
192.168.200.202      *      *                   MP           eth0
```

- Вместе с записями ARP, сервер также создает маршруты для обеих виртуальных машин. Таким образом, когда сервер получает пакет данных, отправленный на IP-адрес 192.168.200.201, он знает, что пакет нужно передать на интерфейс `vme7b9a73a1` виртуальной машины #1.

- Сервер обрабатывает весь входящий трафик для обеих виртуальных машин. Можно рассмотреть следующую ситуацию:
  1. **Компьютеру X** в сети нужно отправить пакет данных для ВМ #1 с IP-адресом 192.168.200.201, поэтому он рассылает ARP-запрос, какой компьютер имеет данный IP-адрес.
  2. Сервер, на котором находится ВМ #1, отправляет ответ на запрос со своим MAC адресом.
  3. **Компьютер X** отправляет пакет данных на указанный MAC адрес.
  4. Физический адаптер `enp0s5` получает пакет и передает его на интерфейс `vme7b9a73a1` виртуальной машины #1.
- Весь исходящий сетевой трафик от ВМ #1 и ВМ #2 направляется через шлюз по умолчанию на физический адаптер `enp0s5`. Шлюзу для виртуальных машин в режиме host-routed автоматически назначается IP-адрес 169.255.30.1. Данный специальный IP-адрес берется из диапазона автоматического назначения частных IP-адресов (APIPA) и используется только для отправки пакетов данных от виртуальных машин на сервер.

Различия между режимами Host-Routed и Bridged

В сравнении с режимом host-routed, режим bridged имеет ряд отличий:

- Каждый виртуальный адаптер `veth` или `vme` имеет назначенный ему MAC адрес, в то время как адаптер host-routed его не имеет. Благодаря MAC-адресу:
  - Любая виртуальная среда может видеть все широковещательные и многоадресные пакеты, получаемые от или отправляемые с выбранного сетевого адаптера на физическом сервере.
  - Можно размещать серверы DHCP или Samba в виртуальных средах.
- Больше нет необходимости настройки всех сетевых параметров (IP-адресов, маски подсети, шлюза и др.) для виртуальных сред на сервере. Все сетевые параметры можно задать внутри виртуальных сред.
- Адаптеры `veth` и `vme` могут соединяться через мост между собой и с другими устройствами. Если несколько адаптеров `veth` объединены в сетевой мост, то данный мост можно использовать для обработки сетевого трафика для всех виртуальных сред, у которых адаптеры `veth` и `vme` включены в мост.
- Из-за того, что адаптеры `veth` и `vme` выступают в роли полноправных участников в сети (а не "спрятаны" за виртуальными сетевыми адаптерами на сервере), они более подвержены уязвимостям системы безопасности: прослушиванию сети, конфликтам IP-адресов и т.п. Таким образом, адаптеры `veth` и `vme` рекомендуется использовать только в надежных сетях.

Настройка виртуальных машин и контейнеров в режиме Host-Routed

У сетевых адаптеров, работающих в режиме host-routed, можно настроить следующие параметры:

- IP-адреса и маски сети
- DNS-серверы
- Домены поиска DNS

Настройка IP-адресов

В сеансе ниже показано, как можно назначить IP-адреса для виртуальной машины `MyVM` и контейнера `MyCT`

```
# prlctl set MyVM --device-set net0 --ipadd 10.0.186.100/24
# prlctl set MyVM --device-set net0 --ipadd 1fe80::20c:29ff:fe01:fb07
# prlctl set MyCT --ipadd 10.0.186.101/24
# prlctl set MyCT --ipadd fe80::20c:29ff:fe01:fb08
```

`net0` в командах обозначает сетевую карту в виртуальной машине `MyVM`, которой назначается IP-адрес. Посмотреть список всех сетевых карт виртуальной машины можно с помощью команды `prlctl list <VM_name> -i`. Для контейнера `MyCT` не нужно указывать имя сетевой карты; `prlctl set` автоматически выполняет операцию со стандартным адаптером, работающим в режиме host-routed.

Настройка адресов DNS-сервера

Чтобы задать DNS-сервер для виртуальной машины `MyVM` и контейнера `MyCT`, можно использовать следующие команды:

```
# prlctl set MyVM --device-set net0 --nameserver 192.168.1.165
# prlctl set MyCT --nameserver 192.168.1.165
```

Настройка доменов поиска DNS

Чтобы задать домен поиска DNS для виртуальной машины `MyVM` и контейнера `MyCT`, необходимо ввести следующие команды:

```
# prlctl set MyVM --device-set net0 --searchdomain 192.168.10.10
# prlctl set MyCT --searchdomain 192.168.10.10
```

Примечания:

1. Настройка сетевых параметров доступна только для виртуальных машин с установленными гостевыми инструментами.

2. Сетевые адаптеры, работающие в режиме host-routed, должны иметь по крайней мере один статический IP-адрес.

3. Для назначения маски сети контейнерам, работающим в сетевом режиме `venet0`, необходимо указать значение `yes` для параметра USE_VENET_MASK в конфигурационном файле `/etc/vz/vz.conf`.

4. Контейнеры могут иметь только один сетевой адаптер, работающий в режиме host-routed. Данный адаптер автоматически создается при создании контейнера.

5. Серверы имен и область поиска можно задать в файле `/etc/vz/vz.conf` с помощью параметров `NAMESERVER` и `SEARCHDOMAIN`. Если задано `inherit`, то значения будут скопированы из файла `/etc/resolv.conf` на хосте.

Переключение адаптеров виртуальных машин в режим Host-Routed

По умолчанию виртуальный адаптер в новой виртуальной машине настроен в режиме bridged (для получения подробной информации см. **Подключение виртуальных сред к виртуальным сетям** (стр. 122)). Чтобы изменить текущий сетевой режим на host-routed, нужно ввести следующую команду:

```
# prlctl set <VM_name> --device-set Net_ID --type routed
```

Например, чтобы настроить адаптер `net0` в виртуальной машине `MyVM` для работы в режиме host-routed, выполните команду:

```
# prlctl set MyVM --device-set net0 --type routed
```

Настройка виртуальных машин и контейнеров в режиме Bridged

Внимание: Для создания виртуальных сетей и сетевых мостов и управления ими необходимо использовать либо интерфейс командной строки, либо ПК Р-Управление, но не оба варианта вместе.

В данном разделе перечислены все операции по настройке виртуальных сред, работающих в режиме bridged.

Управление виртуальными сетями

Виртуальная сеть выступает в роли связывающего интерфейса между виртуальным сетевым адаптером виртуальной среды и соответствующим сетевым адаптером физического сервера. Используя виртуальные сети, можно включать виртуальные машины и контейнеры в различные сети. ПК Р-Виртуализация позволяет управлять виртуальными сетями следующим образом:

- создавать виртуальные сети,
- настраивать параметры виртуальных сетей,
- выводить список виртуальных сетей,
- удалять виртуальные сети.

Данные операции подробно описаны ниже.

Создание виртуальных сетей

По умолчанию ПК Р-Виртуализация создает на сервере следующие виртуальные сети:

- Виртуальная сеть **Bridged**, которая подсоединена к одному из физических адаптеров на физическом сервере (как правило, `enp0s5`) и обеспечивает виртуальным средам, включенным в данную сеть, доступ к сети через физический адаптер.
- Виртуальная сеть **Host-only**, которая подсоединена к специальному виртуальному адаптеру на сервере и позволяет виртуальным средам, объединенным в данную сеть, доступ только к серверу и виртуальным средам данной сети.

Создать собственные виртуальные сети можно при помощи команды `prlsrvctl net add`. Например, для создания новой виртуальной сети `network1` выполните следующую команду:

```
# prlsrvctl net add network1
```

По умолчанию команда создает виртуальную сеть host-only, но при необходимости тип сети можно изменить (см. **Настройка параметров виртуальных сетей** (стр. 118)).

Создание сетевых мостов для сетевых адаптеров

Для соединения сетевого адаптера с виртуальной сетью bridged сначала необходимо создать сетевой мост. Сетевой адаптер может быть физическим (`enp<X>s<Y>`) или логическим: адаптер VLAN (`enp<X>s<Y>.<N>`) или агрегация (`bond<N>`).

Например, чтобы создать сетевой мост для адаптера VLAN, вы можете использовать текстовый пользовательский интерфейс NetworkManager, `nmtui`, следующим образом:

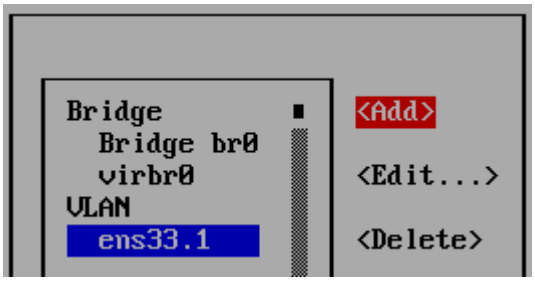
1 На сервере запустите инструмент `nmtui`:

```
# nmtui
```

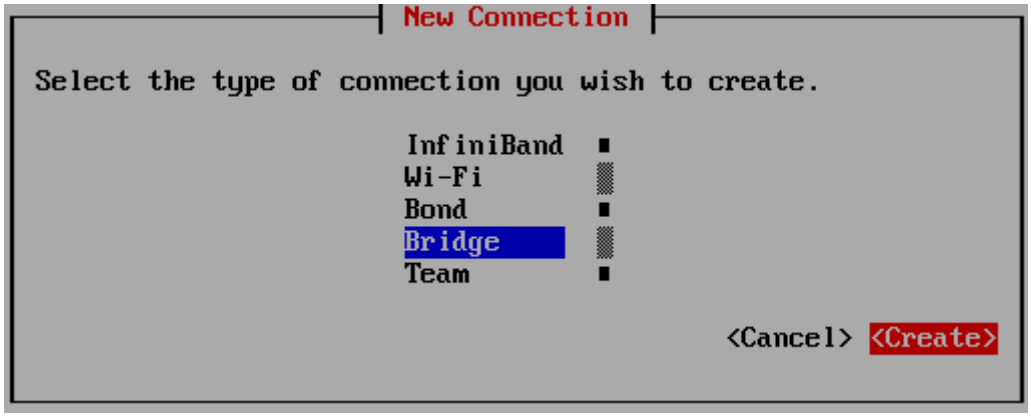
Для работы с интерфейсом используйте клавиши со стрелками и **Tab** для навигации между пунктами, **Enter** для выбора пункта и **Space**, чтобы ставить и снимать флажки.

2 На экране NetworkManager TUI выберите Edit a connection в меню.

3 На следующем экране выберите **Add**, чтобы добавить новое соединение.

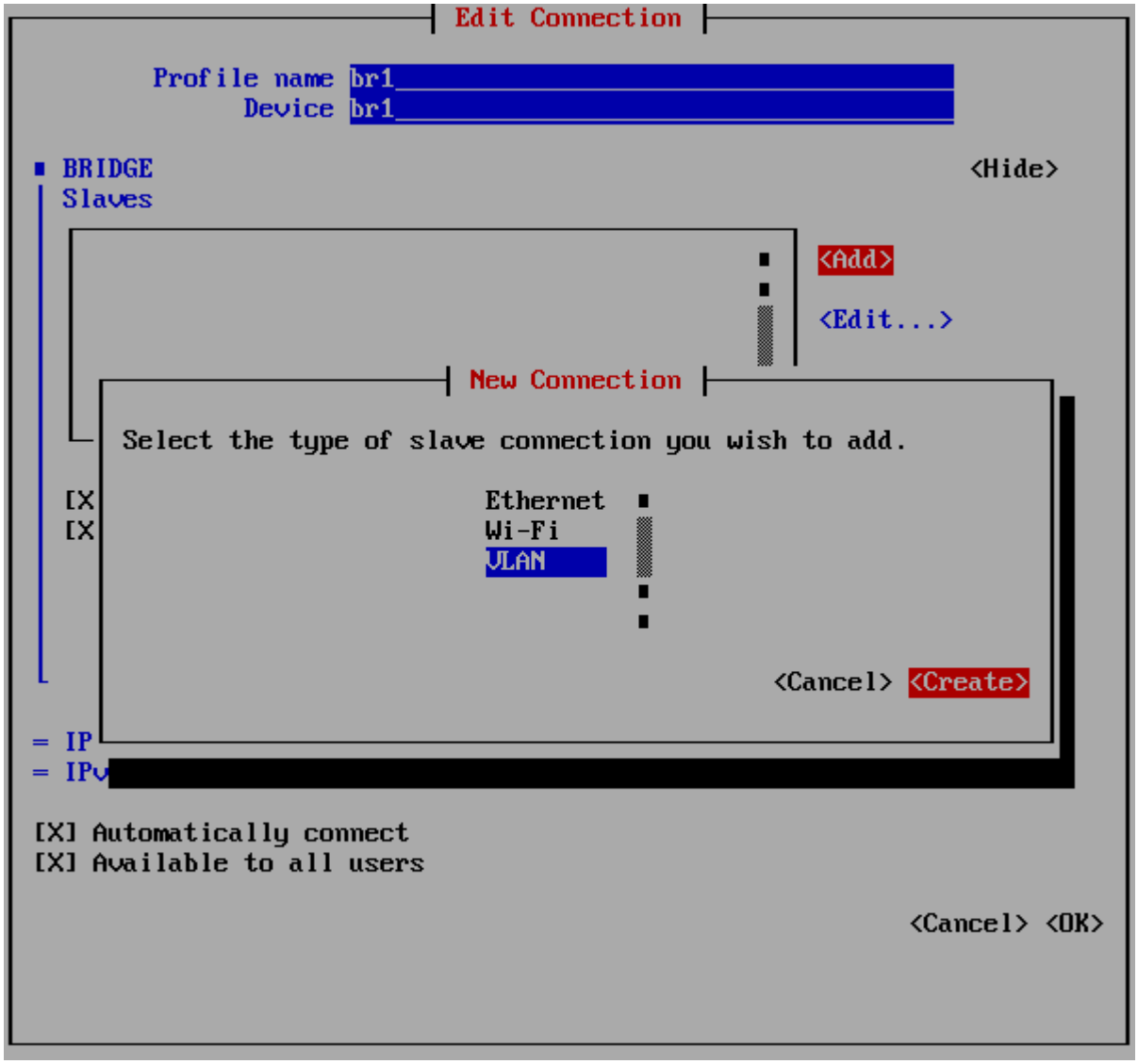


4 Для того чтобы добавить сетевой мост выберите **Bridge** из ниспадающего списка в окне **New connection** и нажмите **Create**.



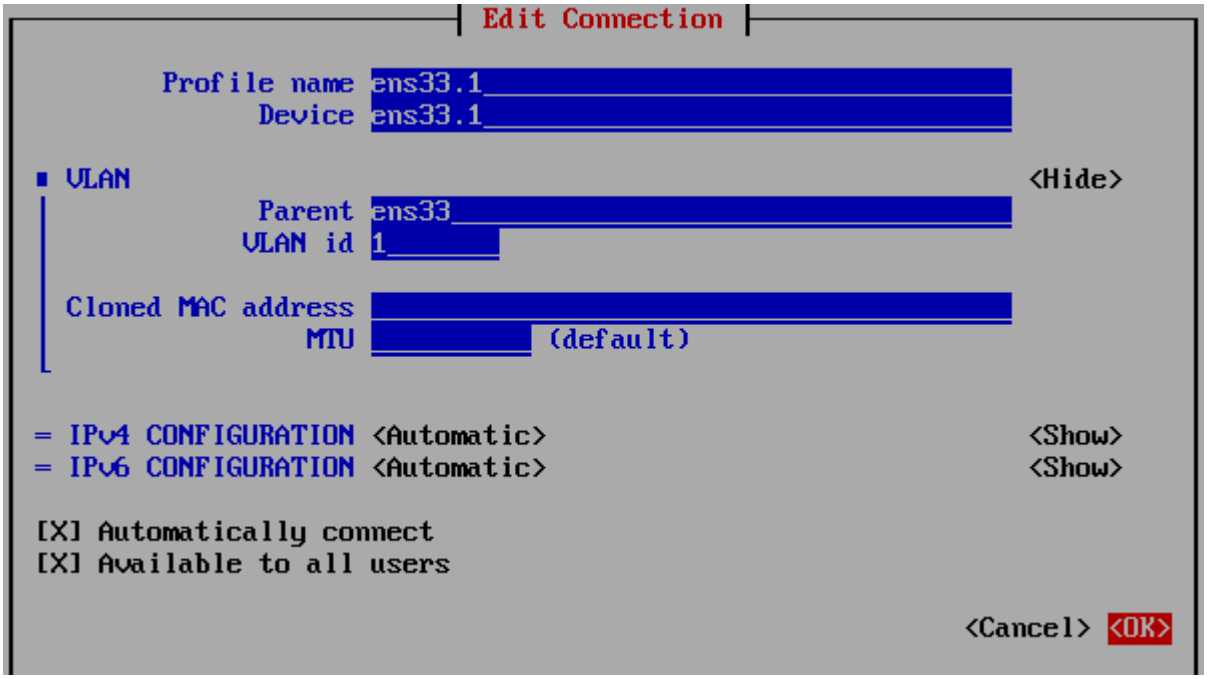
5 На экране Edit connection:

1. В поле **Profile name** введите имя профиля соединения. Данное имя вместе с префиксом `ifcfg-` будет использовано при создании конфигурационного файла интерфейса в директории `/etc/sysconfig/network-scripts/`.
2. В поле **Device** укажите имя устройства нового сетевого моста.
3. Нажмите **Add**, чтобы задать подчиненный сетевой интерфейс.
4. В окне **New connection** выберите **VLAN** из ниспадающего списка и нажмите **Create**.



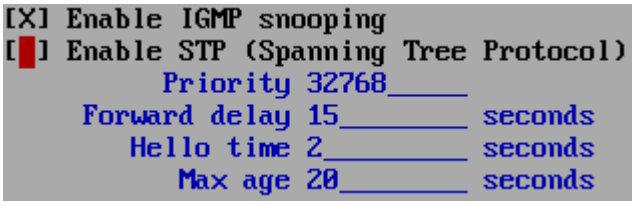
5. В окне **Edit connection** укажите имя профиля и имя устройства для адаптера VLAN в полях **Profile name** и **Device** соответственно и нажмите **OK**.

   Поля **Parent** и **VLAN ID** будут заполнены автоматически.

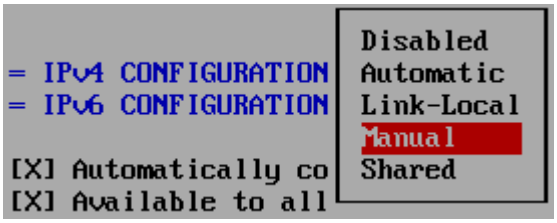


Выбранный адаптер VLAN появится в секции **Slaves**.

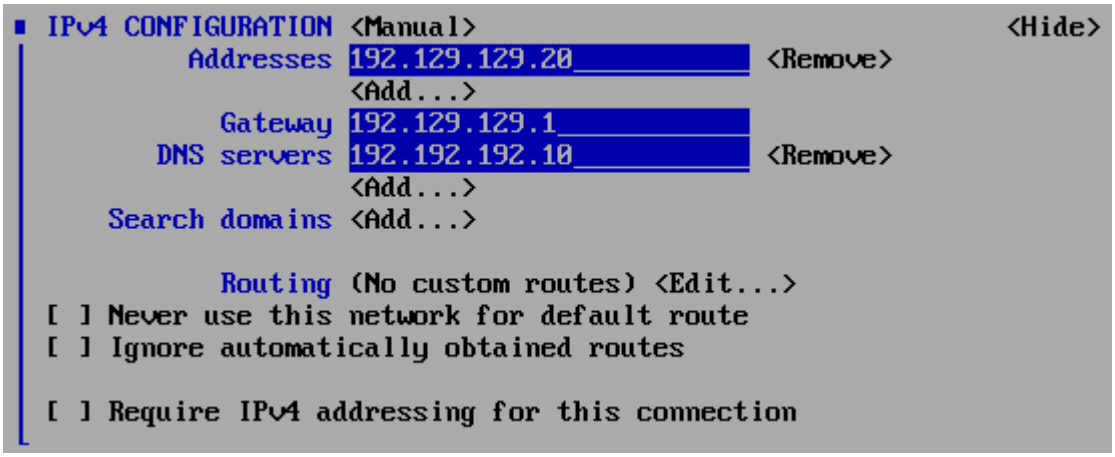
6. Снимите галочку напротив Enable STP (Spanning Tree Protocol).



7. Настройте статические IP-параметры сетевого моста:
   1. В секции **IPv4 CONFIGURATION** нажмите **Automatic** и выберите **Manual** из ниспадающего списка.

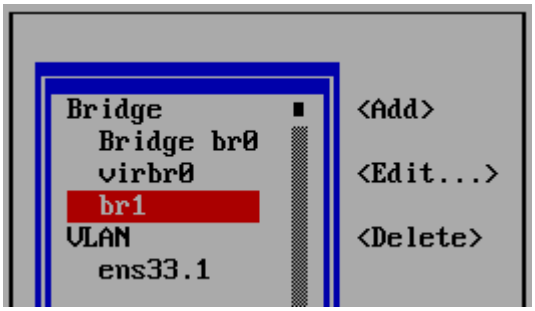


   2. Нажмите **Show**, чтобы отобразить секцию полностью.
   3. Назначьте статический IP-адрес, задайте шлюз по умолчанию и DNS-сервер для сетевого моста в соответствующих полях.



8. При необходимости настройте остальные сетевые параметры и нажмите **OK**.

Сетевой мост для адаптера VLAN появится в списке существующих соединений.



6 Для выхода из `nmtui` нажмите **Back** и затем **Quit**.

После создания сетевого моста вы можете проверить его конфигурационный файл, который хранится в директории `/etc/sysconfig/network-scripts/`. Например:

```
# cat /etc/sysconfig/network-scripts/ifcfg-br1
```

Настройка параметров виртуальных сетей

ПК Р-Виртуализация позволяет настроить следующие параметры для виртуальной сети:

- Сетевой режим, в котором работает виртуальная сеть.

  Примечание: Перед тем, как изменить тип виртуальной сети на bridged, необходимо создать сетевой мост для виртуальной сети. Для получения подробной информации см. **Создание сетевых мостов для сетевых адаптеров** (стр. 114).

- Описание виртуальной сети.

Данные операции можно выполнить при помощи утилиты `prlsrvctl`. Чтобы настроить виртуальную сеть `network1`, которая в данный момент настроена как host-only и имеет описание `This is a host-only virtual network`, необходимо выполнить следующую команду:

```
# prlsrvctl net set network1 -t bridged --ifname enp0s6 -d "This is now a bridged \
virtual network"
```

Данная команда настраивает виртуальную сеть `network1` следующим образом:

1. Изменяет тип виртуальной сети на bridged.
2. Изменяет описание виртуальной сети на следующее: `This is now a bridged virtual network`.

Просмотр списка виртуальных сетей

Для того, чтобы вывести полный список виртуальных сетей, существующих на физическом сервере, можно использовать команду `prlsrvctl net list`:

```
# prlsrvctl net list
Network ID        Type      Bound To       Bridge
Host-Only         host-only                virbr1
Bridged           bridged   enp0s5         br0
```

Информация о виртуальных сетях в выводе команды представлена в виде таблицы со следующими колонками:

Колонка	Описание
Network ID	Имя, назначенное виртуальной сети.
Type	Режим, в котором работает виртуальная сеть.
Bound To	Адаптер на физическом сервере, подключенный к виртуальной сети, если он есть.

Подключение виртуальных сетей к адаптерам

Подключив адаптер на физическом сервере к виртуальной сети, можно включить все виртуальные среды данной виртуальной сети к сети, к которой подключен данный адаптер.

Можно рассмотреть следующую ситуацию:

- На сервере существует физический адаптер `enp0s6` и виртуальная сеть `network1`. Для получения подробной информации о создании виртуальных сетей см. **Создание виртуальных сетей** (стр. 114).
- Физический адаптер `enp0s6` подключен к локальной сети.
- Для физического адаптера `enp0s6` создан сетевой мост `br1`. Для получения подробной информации о создании сетевых мостов см. **Создание сетевых мостов для сетевых адаптеров** (стр. 114).
- Контейнер `MyCT` подключен к виртуальной сети `network1`. Для получения подробной информации о подключении виртуальных сред к виртуальным сетям см. **Подключение виртуальных сред к виртуальным сетям** (стр. 122).

Чтобы подключить адаптер `enp0s6` к виртуальной сети `network1` и, таким образом, включить контейнер `MyCT` в сеть за адаптером `enp0s6`, выполните команду на сервере:

```
# prlsrvctl net set network1 -i enp0s6
```

Чтобы проверить, что физический адаптер enp0s6 добавлен к виртуальной сети network1, введите следующую команду:

```
# prlsrvctl net list
Network ID        Type      Bound To       Bridge
Host-Only         host-only                virbr1
Bridged           bridged   enp0s5         br0
network1          bridged   enp0s6         br1
```

Из вывода команды видно, что адаптер enp0s6 теперь включен в виртуальную сеть network1. Это означает, что контейнер MyCT, виртуальный сетевой адаптер которого подключен к network1, имеет доступ к локальной сети за адаптером enp0s6.

Удаление виртуальных сетей

В любое время можно удалить ненужную виртуальную сеть с физического сервера посредством утилиты prlsrvctl. Например, виртуальную сеть network1 можно удалить, выполнив команду:

```
# prlsrvctl net del network1
```

Чтобы проверить, что виртуальная сеть network1 была удалена, введите команду:

```
# prlsrvctl net list
Network ID     Type        Bound To
Host-Only      host-only
Bridged        bridged     enp0s5
```

Управление виртуальными сетевыми адаптерами в виртуальных средах

ПК Р-Виртуализация позволяет настраивать виртуальные сетевые адаптеры в виртуальных средах и включать их в различные сетевые среды. В данном разделе показано, как можно выполнить следующие операции:

- как создать виртуальные сетевые адаптеры и удалить существующие,
- как настроить параметры существующего виртуального сетевого адаптера,
- как подключить виртуальные сетевые адаптеры к виртуальным сетям.

Данные операции подробно описаны ниже.

Создание и удаление виртуальных адаптеров vme

Виртуальная среда может иметь до 15 виртуальных сетевых адаптеров. Каждый адаптер можно подключить к разным сетям. Чтобы создать виртуальный адаптер для виртуальной машины MyVM, нужно выполнить следующую команду:

```
# prlctl set MyVM --device-add net
```

Проверить, что сетевой адаптер (net1) был добавлен в виртуальную машину, можно с помощью команды:

```
# prlctl list --info MyVM
```

```
ID: {f3b3d134-f512-324b-b0b1-dbd642f5220b}
Name: MyVM
...
net1 (+) dev='vme4208fa77' network='Bridged' mac=001C4208FA77 card=virtio
```

В любое время можно удалить созданный сетевой адаптер (net1), выполнив команду:

```
# prlctl set MyVM --device-del net1
```

Настройка параметров виртуальных адаптеров

ПК Р-Виртуализация позволяет настраивать следующие параметры виртуальных адаптеров:

Настройка MAC адресов

Если по какой-либо причине необходимо повторно сгенерировать MAC адрес сетевого адаптера, можно использовать следующую команду:

```
# prlctl set MyVM --device-set net1 --mac 00:1C:42:2D:74:00
```

Данная команда назначает MAC адрес 00:1C:42:2D:74:00 для адаптера net1 в виртуальной машине MyVM. При сомнении в том, какой MAC адрес назначить виртуальному адаптеру, можно автоматически сгенерировать новый MAC адрес с помощью следующей команды:

```
# prlctl set MyVM --device-set net1 --mac auto
```

Настройка IP-параметров

Как и на любом автономном сервере, для работы в сети в каждой виртуальной среде должны быть правильно настроены параметры TCP/IP. Данные параметры включают:

- IP-адрес
- шлюз по умолчанию
- DNS-сервер

Примечание: Настраивать сетевые параметры можно только для виртуальных машин с установленными гостевыми инструментами.

Обычно данные параметры назначаются при создании виртуальной среды. Однако если параметры еще не настроены или необходимо их изменить, можно использовать команду prlctl set. Например, чтобы назначить IP-адрес 192.129.129.20 адаптеру net1, шлюз по умолчанию 192.129.129.1 и DNS-сервер 192.192.192.10 в виртуальной машине MyVM, нужно выполнить следующую команду:

```
# prlctl set MyVM --device-set net1 --ipadd 192.129.129.20 --gw 192.129.129.1 \
--nameserver 192.192.192.10
```

Наряду со статическим назначением сетевых параметров виртуальному адаптеру, можно настроить адаптер, чтобы он получал настройки TCP/IP автоматически, при помощи протокола динамической настройки узла (DHCP). Например, чтобы адаптер net1 в виртуальной машине MyVM получал свои настройки IP через DHCP, нужно ввести команду:

```
# prlctl set MyVM --device-set net1 --dhcp yes
```

Подключение виртуальных сред к виртуальным сетям

В ПК Р-Виртуализация можно подключать виртуальные среды к виртуальным сетям следующих типов:

- Виртуальная сеть **Bridged** позволяет виртуальной среде использовать один из физических сетевых адаптеров и появляться в виде отдельного компьютера в сети, к которой относится соответствующий адаптер.
- Виртуальная сеть **Host-only** позволяет виртуальной среде иметь доступ только к физическому серверу и виртуальным средам, входящими в данную сеть.

По умолчанию любой новый адаптер подключается к сети Bridged. Чтобы включить виртуальную среду в другую сеть, можно использовать команду `prlctl set`. Например, в сеансе ниже показано, как можно подключить адаптер `net0` в виртуальной машине `MyVM` к виртуальной сети `network1`.

Перед подсоединением виртуальной машины `MyVM` к виртуальной сети `network1` можно проверить, к какому физическому сетевому адаптеру подключена данная виртуальная сеть с помощью следующей команды:

```
# prlsrvctl net list
Network ID        Type      Bound To
Host-Only         host-only
Bridged           bridged   enp0s5
network1          bridged   enp0s6
```

Из вывода команды видно, что виртуальная сеть `network1` подключена к физическому сетевому адаптеру `enp0s6` на сервере. Это означает, что после подключения виртуальной машины `MyVM` к виртуальной сети `network1` виртуальная машина будет иметь доступ ко всем компьютерам в сети, к которой подсоединен адаптер `enp0s6`.

Теперь для подключения адаптера `net1` в виртуальной машине `MyVM` к виртуальной сети `network1` можно ввести следующую команду:

```
# prlctl set MyVM --device-set net1 --network network1
```

Проверить, что сетевой адаптер (`net1`) был подключен к виртуальной сети, можно с помощью команды:

```
# prlctl list --info MyVM
ID: {f3b3d134-f512-324b-b0b1-dbd642f5220b}
Name: MyVM
...
net1 (+) dev='vme4208fa77' network='network1' mac=001C4208FA77 card=virtio
```

Глава 6

Управление лицензиями

В данной главе объясняется, как установить, обновить, просмотреть и перенести лицензии на серверы.

Примечание: Для использования функциональности ПК Р-Хранилище необходимо установить отдельную лицензию в дополнение к лицензии ПК Р-Виртуализация. Для получения подробной информации об управлении лицензиями ПК Р-Хранилище см. *Руководство администратора по ПК Р-Хранилище*.

Установка лицензии

ПК Р-Виртуализация требует, чтобы на каждый сервер была установлена отдельная лицензия. Лицензию можно установить из файла лицензии с помощью команды `vzlicload -f`:

```
# vzlicload -f <license_file>
```

Просмотр лицензии

Инструмент `vzlicview` можно использовать для просмотра информации об установленной лицензии и ее текущем статусе. Например:

```
# vzlicview
Searching for installed licenses...
VZSRV
        owner_name="Autogenerated Trial Licenses (R-Platforma)"
        status="ACTIVE"
        version=7.0
        owner_id=70295522.10134133
        hwid="3D6F.FFB2.1CD5.1E47.1325.BBE0.6C66.ACF5"
        serial="BF3D.E943.18C5.5555.5D10.99D9.0310.DFA0"
        expiration="07/08/2016 03:00:00"
        start_date="06/07/2016 03:00:00"
        issue_date="06/08/2016 16:27:11"
        graceperiod=259200 (259200)
        key_number="RVZ.10134133.0001"
        cpu_total=8 (1)
        ct_total="unlimited" (0)
        architecture="x86"
        architecture="x86_64"
        platform="Linux"
        product="RVZ"
        nr_vms="unlimited" (2)
        subscription="70295522:dd482d1a-5268-4980-ae06-2a288a4fb7ab"
```

Параметры лицензии описаны в следующей таблице:

Колонка	Описание
owner_name	Имя владельца лицензии.
status	Статус лицензии. Описание статусов лицензии см. в разделе **Статусы лицензии**.
version	Версия ПК Р-Виртуализация, для которой предназначена лицензия.
owner_id	ID владельца лицензии.
hwid	Идентификатор аппаратного оборудования сервера.
serial	Серийный номер лицензии. В частности, используется для идентификации файлов лицензии в `/etc/vz/licenses/`.
expiration	Дата окончания срока действия лицензии, если лицензия временная.
start_date	Дата начала срока действия лицензии.
issue_date	Дата предоставления лицензии.
graceperiod	Период, в секундах, в течение которого ПК Р-Виртуализация продолжает работать после истечения срока действия лицензии или при превышении числа запущенных виртуальных машин и контейнеров, допустимого лицензией.
license_update_date	Дата последнего обновления лицензии.
key_number	Номер, под которым лицензия зарегистрирована на сервере Key Authentication.
cpu_total	Общее количество ЦП на физического сервера, допустимое лицензией.
ct_total	Общее число контейнеров, которое можно одновременно запускать на сервере.
architecture	Архитектура системы, с которой совместима лицензия.
platform	Операционная платформа, с которой совместима лицензия.
product	Название продукта, для которого предназначена лицензия.
keyserver_host	Имя хоста и порт сервера Key Authentication.
nr_vms	Общее число виртуальных машин, которое можно одновременно запускать на сервере.
subscription	Ключ подписки на функции ПК Р-Виртуализация.

Статусы лицензии

При просмотре информации о лицензии следует обратить особое внимание на статус лицензии, который может быть одним из следующих:

Статус	Описание
ACTIVE	Лицензия, установленная на сервере, действительна и активна
VALID	Лицензия действительна и может быть установлена на сервере.
EXPIRED	Срок действия лицензии истек, и лицензию нельзя установить на сервер.
GRACED	Лицензия установлена на сервере, но в текущий момент предоставлен период отсрочки, так как срок действия лицензии истек или число запущенных виртуальных машин и контейнеров превысило допустимое лицензией значение.
INVALID	Лицензия недействительна (например, из-за несоответствия архитектуры сервера) или повреждена.

Глава 7

Поддержание системы в актуальном состоянии

В данной главе описываются способы поддержания физического сервера в актуальном состоянии. Необходимо следить за состоянием следующих компонентов:

- ПО ПК Р-Виртуализация
- Виртуальные машины и контейнеры на сервере.

Для применения крупных обновлений, которые обычно включают новые версии ядра, следуйте инструкции, приведенной ниже:

Примечание: Чтобы узнать доступно ли крупное обновление, выполните команду `yum info r-virtualization-release` и сравните значения в поле `Version` для установленных пакетов и пакетов, доступных для установки. Если версии отличаются третьей цифрой, например, установлена версия `7.0.4` и версия `7.0.5` доступна для установки, значит, вы можете установить крупное обновление ПК Р-Виртуализация.

1. Остановите все запущенные виртуальные среды на сервере, который вы хотите обновить, или мигрируйте их на другие серверы ПК Р-Виртуализация, чтобы избежать простоя.
2. Обновите сервере, как описано в разделе **Обновление всех компонентов** (стр. 126).
3. Перезагрузите сервер, чтобы обновить ядро.
4. Запустите виртуальные среды или мигрируйте их обратно на обновленный сервер, в зависимости от действия, выполненного в шаге 1.
5. Выполните шаги 1-4 для других серверов ПК Р-Виртуализация, которые необходимо обновить.

   Примечание: Шаги 1-4 также можно выполнить при помощи ПК Р-Управление.

6. Если вы используете ПК Р-Управление, обновите контейнер `va-mn` через ПК Р-Управление (см. *Руководство администратора ПК Р-Управление*) или выполнив следующие команды на сервере:

```
# prlctl exec va-mn yum update
# prlctl restart va-mn
```

7. Если вы используете панель управления Р-Хранилище, обновите контейнер `vstorage-ui`, выполнив следующие команды на сервере:

```
# prlctl exec vstorage-ui yum update
# prlctl restart vstorage-ui
```

Обновление ПК Р-Виртуализация

Важно: Настоятельно рекомендуется на каждом сервере иметь одну и ту же версию ПК Р-Виртуализация. В крайнем случае, версии могут отличаться только на одно крупное обновление. Например, перед обновлением сервера до ПК Р-Виртуализация Обновление 7, необходимо убедиться, что все остальные серверы в кластере имеют Обновление 6. Причина заключается в том, что виртуальные машины, созданные на более позних версиях, могут не запуститься на серверах со старой версией, таким образом, возникнут проблемы с миграцией, восстановлением из резервной копии, высокой доступностью и т. д.

ПК Р-Виртуализация позволяет быстро и просто обновлять систему с помощью утилиты `yum`, стандартной для RPM-совместимых операционных систем Linux. Основными компонентами, которые нужно обновлять, являются:

- утилиты и библиотеки,
- ядро,
- EZ-шаблоны,
- патч ReadyKernel,
- гипервизор KVM/QEMU в запущенных виртуальных машинах.

Примечание: Просмотреть доступные обновления перед их установкой можно с помощью команды `yum check-update`.

Обновление всех компонентов

Самым простым способом обновления всех компонентов ПО ПК Р-Виртуализация является команда `yum update`. Данная команда сообщает утилите `yum` выполнить следующие действия:

1. Обратиться к удаленным репозиториям ПК Р-Виртуализация.
2. Проверить доступные обновления для ядра, утилит, библиотек и EZ-шаблонов ПК Р-Виртуализация.
3. Установить найденные обновления на сервер.

Стоит обратить внимание на то, что утилита `yum` может обновлять только пакеты, установленные на сервер. Поэтому, если пакет не доступен в системе, то сначала необходимо его установить с помощью команды `yum install`.

Обновление ядра

Для обновления ядра ПК Р-Виртуализация нужно обновить пакеты `vzkernel` и `vzkernel-devel`:

```
# yum update vzkernel vzkernel-devel
```

Обновление гипервизора KVM/QEMU в виртуальных машинах

ПК Р-Виртуализация может обновлять гипервизор KVM/QEMU без перезагрузки в запущенных виртуальных машинах с KVM/QEMU версии 2.6.0 или более новой.

Для этого необходимо установить пакет `vz-qemu-engine-updater` и обновить пакет `qemu-kvm-vz`:

```
# yum install vz-qemu-engine-updater
# yum update qemu-kvm-vz
```

Обновление `qemu-kvm-vz` запускает 10-минутный таймер (чтобы дать время `yum` для завершения операции), после которого запускается инструмент `vz-qemu-engine-updater`, который начинает обновление KVM/QEMU в каждой запущенной виртуальной машине по очереди.

- если `yum` в данный момент заблокирован на сервере (в таком случае автоматическое обновление виртуальных машин будет доступно только после снятия блокировки),
- если конфигурация виртуальной машины изменяется,
- если виртуальная машина находится в переходных состояниях (например, с `running` на `stopped`),
- если для виртуальной машины создается резервная копия или
- любая другая операция `prlctl` выполняется с виртуальной машиной.

Программа обновления KVM/QEMU пропустит подобные виртуальные машины и добавит их в очередь для следующего обновления. Программа обновления попробует обновить пропущенные виртуальные машины заданное количество раз с заданным интервалом между попытками. Если попыток больше не осталось или обновление не удается по какой-либо причине, то виртуальная машина остается в запущенном состоянии с устаревшей версией KVM/QEMU.

Для отключения автоматического обновления вручную заблокируйте службу программы обновления:

```
# systemctl mask vz-qemu-engine-updater.service
```

Чтобы включить ее снова, разблокируйте службу программы обновления:

```
# systemctl unmask vz-qemu-engine-updater.service
```

Для того чтобы проверить, обновились ли виртуальные машины, просмотрите журнал:

```
# journalctl -u vz-qemu-engine-updater
<...>VM MyVM (was using qemu-kvm-vz-2.6.0-28.3.6.vz7.30) has been successfully updated.
<...>Finished updating VMs.
<...>Successfully updated QEMU engine for all running VMs.
```

Для настройки количества попыток, интервала между ними и других параметров, см. man-страницу по `vz-qemu-engine-updater .json` и измените конфигурационный файл `/var/lib/vz-qemu-engine-updater.json`.

Обновление гипервизора KVM/QEMU вручную

Если по какой-либо причине гипервизор KVM/QEMU не был автоматически обновлен в запущенной виртуальной машине, его можно обновить вручную следующим способом:

1 Убедитесь, что виртуальная машина использует устаревшую версию гипервизора.

Проверьте версию, установленную на сервере:

```
# rpm -qa qemu-kvm-vz
qemu-kvm-vz-<version>.x86_64
```

И сравните ее с версией, которую виртуальная машина использует в данный момент:

```
# virsh qemu-monitor-command <VM_name> '{"execute":"query-version"}'
<...>"package":" (qemu-kvm-vz-<version>)"}<...>
```

2 Убедитесь, что с виртуальной машиной не выполняются операции `prlctl`.

3 Обновите гипервизор, который использует виртуальная машина:

```
# prlctl update-qemu <VM_name>
```

Обновление EZ-шаблонов

EZ-шаблон можно обновить, как и любой другой RPM-пакет, при помощи команды `yum update`. Например:

```
# yum update centos-6-x86_64-ez
...
Updated:
  centos-6-x86_64-ez.noarch 0:4.7.0-1
Complete!
```

Примечания:

1. При обновлении EZ-шаблонов ОС необходимо добавить `ez` к имени шаблона.

2. Для обновления EZ-шаблона также можно использовать команду `vzpkg update template`.

Проверка обновлений

Перед обновлением можно посмотреть, какие пакеты доступны для обновления и до какой версии, при помощи команды `yum check-update`. Например:

```
# yum check-update
```

Дополнительные параметры yum

Команда `yum` позволяет не только проверить наличие обновлений и установить их. Для обновления ПК Р-Виртуализация также могут быть полезны такие параметры, как `search`, `list`, `info`, `deplist`, `provide`. Для получения подробной информации о данных и других параметрах см. man-страницу по `yum`.

Обновление ядра ПК Р-виртуализация с помощью ReadyKernel

ReadyKernel является службой на базе kpatch, которая предлагает более удобную альтернативу обновления ядра обычным способом без перезагрузки и позволяет ждать запланированного простоя сервера для применения важных обновлений для системы безопасности. ReadyKernel позволяет получать накопленные патчи для ядер Р-Виртуализация, которые устраняют важные неисправности в системе безопасности, и применять данные патчи без необходимости перезагрузки сервера. Обновления ReadyKernel выпускаются для ядер Р-Виртуализация не старше 18 месяцев. Когда ядру исполняется более 18 месяцев, его необходимо обновить, например, до последней версии для того, чтобы продолжать получать обновления ReadyKernel.

После установки патчи загружаются в ОЗУ сервера и сразу же применяются к ядру. Если сервер перезагружается, то данные патчи применяются повторно к ядру при его загрузке.

Если затем будет установлено новое ядро или основное обновление ядра, которые требуют перезагрузки, загруженные патчи останутся на сервере, но не будут применены.

Примечание: В любое время можно посмотреть подробную информацию о примененном патче ReadyKernel с помощью команды `readykernel info`.

Патчи ReadyKernel можно получать и устанавливать автоматически и вручную, как описано ниже.

Автоматическая установка патчей ReadyKernel

Чтобы включить автоматическую загрузку и установку последних патчей ReadyKernel для текущего ядра, выполните следующую команду:

```
# readykernel autoupdate enable <hour>
```

Служба будет проверять наличие патчей каждый день в определенный час `<hour>` (задается в 24-часовом формате по серверному времени) при помощи скрипта `cron.d`.

Для отключения автоматического обновления введите команду:

```
# readykernel autoupdate disable
```

Управление патчами ReadyKernel вручную

Загрузка, установка и применение патчей ReadyKernel к ядру

Чтобы загрузить, установить и мгновенно применить к ядру последний патч ReadyKernel, выполните следующие действия:

1 Проверьте наличие новых патчей ReadyKernel:

```
# readykernel check-update
```

2 Если новый патч для текущего ядра доступен, загрузите, установите и мгновенно примените его к ядру с помощью команды:

```
# readykernel update
```

Примечание: Это действие также можно выполнить с помощью `yum update`.

Патчи ReadyKernel являются накопительными, т.е. последний патч содержит все предыдущие. Для обеспечения безопасности ядра необходимо лишь установить и применить последний патч.

Применение и отмена патчей ReadyKernel

Для того чтобы вручную применить последний установленный патч ReadyKernel к ядру, выполните одно их следующих действий:

- Если применен старый патч, необходимо его сначала отменить, а затем применить последний патч с помощью команды:

```
# readykernel load-replace
```

- Если старый патч не применен, примените последний патч с помощью команды:

```
# readykernel load
```

Для отмены патча в текущем ядре выполните команду:

```
# readykernel unload
```

Установка и удаление патчей ReadyKernel для определенных ядер

Если на сервере установлено более одного ядра, можно установить патч ReadyKernel для определенного ядра:

```
# yum install readykernel-patch-<kernel_version>
```

Для удаления определенного патча ReadyKernel с сервера выполните команду:

```
# yum remove readykernel-patch-<kernel_version>
```

Возврат к предыдущему патчу ReadyKernel

При возникновении проблем с последним патчем ReadyKernel можно возвратиться к предыдущей версии патча, если она доступна.

Для возврата патча для текущего ядра к предыдущей версии выполните команду:

```
yum downgrade readykernel-patch-$(uname -r)
```

Для возврата патча для определенного ядра к предыдущей версии введите команду:

```
yum downgrade readykernel-patch-<kernel_version>
```

Данные команды можно вводить несколько раз для возврата к нужной версии патча. Вернуться к определенной версии можно также, указав желаемую версию патча. Например:

```
yum downgrade readykernel-patch-12.7-0.4-17.vl7
```

Отключение загрузки патчей ReadyKernel при запуске

Если по какой-либо причине вы не хотите, чтобы патчи ReadyKernel применялись во время загрузки, выполните следующую команду:

```
# readykernel autoload disable
```

Чтобы заново включить автоматическую загрузку патчей ReadyKernel при запуске, выполните команду:

```
# readykernel autoload enable
```

Управление журналами ReadyKernel

Журналы событий ReadyKernel хранятся в `/var/log/messages` и `/var/log/kpatch.log`. Для второго журнала можно указать параметры в конфигурационном файле `/etc/logrotate.d/kpatch`. Для получения подробной информации о параметрах, которые можно использовать, см. man-страницу по `logrotate`.

Обновление ПО в виртуальных машинах

Для поддержания программного обеспечения в виртуальных машинах в актуальном состоянии можно использовать те же средства, что и на автономных компьютерах с установленной соответствующей операционной системой:

- В виртуальных машинах Linux можно использовать собственные утилиты для обновления пакетов Linux (`up2date`, `yum` или `yast`).
- В виртуальных машинах Windows можно использовать собственные утилиты для обновления пакетов Windows (например, Windows Update).

Обновление гостевых инструментов в виртуальных машинах

В ПК Р-Виртуализация гостевые инструменты в виртуальных машинах обновляются автоматически с помощью еженедельной задачи `cron`, которая запускает инструмент `vz-guest-tools-updater`. Для этого должны выполняться следующие условия:

- На сервере установлен пакет `vz-guest-tools-updater`.
- У виртуальной машины параметр `--tools-autoupdate` имеет значение `on` (поведение по умолчанию).

Примечание: Гостевые инструменты также автоматиечски устанавливаются в виртуальные машины при их следующем запуске (см. **Установка гостевых инструментов** (стр. 25)). Чтобы отключить автоматическую установку гостевых инструментов, в конфигурационном файле `/etc/vz/tools-update.conf` параметру `InstallTools` задайте значение `false`.

Инструмент `vz-guest-tools-updater` составляет список виртуальных машин с включенным параметром `--tools-autoupdate` и устаревшими гостевыми инструментами. После этого 5-минутный таймер запускает одновременное обновление гостевых инструментов в настраиваемом числе виртуальных машин. При неудачной попытке обновления инструмент добавит данную виртуальную машину в очередь задач для следующей попытки. Если следующая попытка будет также неудачной, то гостевые инструменты виртуальной машины останутся устаревшими.

Важно: В процессе обновления образ гостевых инструментов принудительно монтируется к оптическому дисководу виртуальной машины, даже если он в это время используется.

Для завершения обновления гостевых инструментов виртуальные машины Windows необходимо перезапустить. При каждом обновлении администраторы в виртуальных машинах получают уведомление о перезагрузке при входе в систему или, если они находятся в системе, то сразу после обновления.

Число виртуальных машин, в которых будут одновременно обновляться гостевые инструменты, можно изменить в конфигурационном файле `/etc/vz/tools-update.conf` с помощью параметра `MaxVMs`.

Чтобы проверить состояние обновления гостевых инструментов в одной или нескольких виртуальных машинах, используйте параметр `--get-state` для инструмента `vz-guest-tools-updater` tool и укажите последовательно имена виртуальных машин. Например:

```
# vz-guest-tools-updater --get-state <VM1_name> [<VM2_name> ...]
```

Если гостевые инструменты в данной виртуальной машине не требуют обновления, вывод команды будет следующим:

```
{<VM_UUID>} (<VM_name>): Tools are up to date
```

Для отмены автоматического обновления гостевых инструментов в виртуальной машине выполните команду:

```
# prlctl set <VM_name> --tools-autoupdate off
```

Чтобы обновить гостевые инструменты вручную в одной или нескольких виртуальных машинах, запустите скрипт `vz-guest-tools-updater`, указав последовательно имена виртуальных машин. Например:

```
# vz-guest-tools-updater <VM1_name> [<VM2_name> ...]
```

Обновление контейнеров

В ПК Р-Виртуализация существует два способа поддержания контейнеров в актуальном состоянии:

- Обновление программных пакетов EZ-шаблонов внутри определенного контейнера с помощью утилиты `vzpkg`. Данная функция позволяет обновить любой контейнер на физическом сервере.

- Обновление кэша EZ-шаблонов ОС, установленных на сервере. Данная функция позволяет создать новые контейнеры, в которых установлены последние программные пакеты.

Обновление пакетов EZ-шаблонов в контейнерах

В ПК Р-Виртуализация можно обновить пакеты EZ-шаблона ОС, по которому был создан контейнер, и любого EZ-шаблона приложения, который был применен к контейнеру. Данное обновление можно осуществить с помощью команды `vzpkg update`. Если контейнер `MyCT` был создан на базе EZ-шаблона ОС `centos-6-x86_64`, то для обновления всех пакетов, включенных в данный шаблон, нужно выполнить команду:

```
# vzpkg update 26bc47f6-353f-444b-bc35-b634a88dbbcc centos-6-x86_64
...
  Updating: httpd                ######################### [1/4]
  Updating: vzdev                ######################### [2/4]
  Cleanup : vzdev                ######################### [3/4]
  Cleanup : httpd                ######################### [4/4]
Updated: httpd.i386 0:2.0.54-10.2 vzdev.noarch 0:1.0-4.swsoft
Complete!
Updated:
 httpd                  i386      0:2.0.54-10.2
 vzdev                  noarch    0:1.0-4.swsoft
```

Примечания:

1. Обновление EZ-шаблонов поддерживается только для запущенных контейнеров.

2. Если необходимо обновить кэш платного EZ-шаблона ОС (например, Red Hat Enterprise Server 5 или SLES 10), то сначала нужно обновить программные пакеты в удаленном репозитории, который содержит данный EZ-шаблон ОС, а только после этого обновить его кэш.

В примере выше видно, что приложения `httpd` и `vzdev` были обновлены для EZ-шаблона ОС `centos-6-x86_64`. Чтобы сразу обновить все EZ-шаблоны (включая EZ-шаблон ОС) внутри контейнера `MyCT`, нужно выполнить следующую команду:

```
# vzpkg update 26bc47f6-353f-444b-bc35-b634a88dbbcc
...
Running Transaction
  Updating  : hwdata               ######################### [1/2]
  Cleanup   : hwdata               ######################### [2/2]
Updated: hwdata.noarch 0:1.0-3.swsoft
Complete!
Updated:
 hwdata                 noarch    0:0.158.1-1
```

В примере выше только пакет `hwdata` в контейнере `MyCT` был обновлен до последней версии.

Обновление кэша EZ-шаблонов ОС

С выпуском новых обновлений для соответствующих дистрибутивов Linux созданный кэш EZ-шаблонов ОС может устареть. ПК Р-Виртуализация позволяет быстро обновить кэш EZ-шаблонов ОС с помощью команды `vzpkg update cache`.

Примечание: Если необходимо обновить кэш платного EZ-шаблона ОС (например, Red Hat Enterprise Server 5 или SLES 10), то сначала нужно обновить программные пакеты в удаленном репозитории, который содержит данный EZ-шаблон ОС, а только после этого обновить его кэш.

Команда `vzpkg update cache` проверяет директорию `cache` в области шаблона (`/vz/template/cache` по умолчанию) на физическом сервере и обновляет все существующие tar-архивы в данной директории. Однако можно указать, для какого EZ-шаблона ОС нужно обновить tar-архив, добавив имя данного EZ-шаблона ОС после команды. Например, чтобы обновить tar-архив для EZ-шаблона ОС `centos-6-x86_64`, можно выполнить следующую команду:

```
# vzpkg update cache centos-6-x86_64
Loading "rpm2vzrpm" plugin
Setting up Update Process
Setting up repositories
base0            100% |=========================|  951 B    00:00
base1            100% |=========================|  951 B    00:00
base2            100% |=========================|  951 B    00:00
base3            100% |=========================|  951 B    00:00
...
```

После выполнения команды `vzpkg update cache` к имени старого tar-архива добавляется суффикс `-old` (например, `centos-x86.tar.gz-old`).

При помощи параметра `-f` команды `vzpkg update cache` можно удалить существующий tar-архив и вместо него создать новый.

Если команда `vzpkg update cache` не найдет tar-архив для одного или нескольких EZ-шаблонов ОС, установленных на сервере, она создаст tar-архивы соответствующих EZ-шаблонов ОС и поместит их в директорию `/vz/template/cache`.

Глава 8

Управление кластерами высокой доступности

В данной главе объясняется управление высокой доступностью для серверов, включенных в кластеры ПК Р-Хранилище.

Высокая доступность обеспечивает работу виртуальных машин, контейнеров и целей iSCSI даже при отказе физического сервера, на котором они находятся. В подобном случае виртуальные среды с отказавшего сервера перемещаются на исправный физический сервер в кластере. Высокая доступность обеспечивается:

- Избыточностью метаданных. Для работы кластера ПК Р-Хранилище должны быть запущены не все, но большинство серверов метаданных. Установка нескольких серверов метаданных в кластере позволяет при отказе одного сервера метаданных продолжить работу кластера с другими серверами метаданных.
- Избыточностью данных. Копии каждого фрагмента данных хранятся на разных серверах хранилища, чтобы обеспечить доступность данных даже при отказе какого-либо сервера хранилища.

  Примечание: Избыточность достигается одним из двух методов: репликацией или избыточным кодированием (для получения подробной информации см. раздел **Понимание избыточности данных** в *Руководстве по установке ПК Р-Виртуализация*).

- Мониторингом состояния сервера.

Инструкции по управлению высокой доступностью зависят от разных сценариев. Сценарии могут быть следующими:

- Вы используете ПК Р-Хранилище с управлением при помощи командной строки и не включили высокую доступность для виртуальных сред при выборе роли клиента во время установки, но хотите ее включить.

  В этом случае читайте данную главу дальше.

- Вы используете ПК Р-Хранилище с управлением при помощи командной строки и включили высокую доступность для виртуальных сред при выборе роли клиента во время установки, но хотите изменить режим перераспределения ресурсов, выбранный по умолчанию.

  В этом случае для виртуальных машин и контейнеров на физическом сервере по умолчанию будет включен режим DRS. Чтобы изменить режим перераспределения ресурсов, например, на режим round-robin, см. инструкции в разделе **Настройка режимов перераспределения ресурсов** (стр. 139).

- Вы используете ПК Р-Хранилище с управлением при помощи графического интерфейса и еще не назначили сетевые роли iSCSI и не создали кластеров S3 (см.

Руководство по эксплуатации ПК Р-Хранилище с графическим интерфейсом), но хотите настроить высокую доступность вручную (например, чтобы самостоятельно выбрать режим перераспределения ресурсов).

В этом случае читайте данную главу дальше.

- Вы используете ПК Р-Хранилище с управлением при помощи графического интерфейса, и вы назначили сетевую роль iSCSI сетевому интерфейсу или включили сервер в кластер S3 (см. *Руководство по эксплуатации ПК Р-Хранилище с графическим интерфейсом*), но хотите изменить режим перераспределения ресурсов, выбранный по умолчанию.

  В этом случае для виртуальных машин и контейнеров на физическом сервере по умолчанию будет включен режим round-robin. Чтобы изменить режим перераспределения ресурсов, например, на DRS, см. инструкции в разделе **Настройка режимов перераспределения ресурсов на серверах, участвующих в экспорте iSCSI** (стр. 141).

Проверка требований для высокой доступности

Для работы функции высокой доступности должны соблюдаться следующие требования:

- В сети должен быть настроен кластер ПК Р-Хранилище. Высокая доступность поддерживается только для серверов, включенных в кластеры ПК Р-Хранилище.
- В кластере ПК Р-Хранилище должно быть от 5 серверов.
- Виртуальные машины и контейнеры, находящиеся на сервере, должны храниться в кластере:
  - Если вы используете ПК Р-Хранилище с управлением при помощи графического интерфейса, войдите в панель управления, создайте хранилища данных и поместите в них свои виртуальные среды, как описано в разделе **Управление хранилищами данных** в *Руководстве по эксплуатации ПК Р-Хранилище с графическим интерфейсом*.
  - Если вы используете ПК Р-Хранилище с управлением при помощи командной строки, виртуальные среды уже автоматически настроены на хранение в кластере на серверах, для которых в процессе установки была выбрана роль клиента. Вы также можете настроить их вручную (для этого см. *Руководство по эксплуатации ПК Р-Хранилище с командной строкой*).
- Выбранный режим избыточности должен защищать данные при одновременном отказе двух или более серверов. Список режимов избыточности и их отличие см. в разделе **Понимание избыточности данных** в *Руководстве по установке ПК Р-Виртуализация*.
  - Если вы используете ПК Р-Хранилище с управлением при помощи графического интерфейса, вы выбираете режим избыточности при создании хранилищ данных.

- Если вы используете ПК Р-Хранилище с управлением при помощи командной строки, то вам нужно изменить стандартные параметры избыточности для директорий, в которых хранятся ваши виртуальные среды.

  Например, чтобы задать режим 3:2 реплики для директорий `vmprivate` и `private` (с виртуальными машинами и контейнерами соответственно) в кластере `stor1`, выполните следующие команды:

```
# vstorage set-attr -R /vstorage/stor1/vmprivate replicas=3
# vstorage set-attr -R /vstorage/stor1/private replicas=3
```

  Примечание: Не рекомендуется использовать режим избыточного кодирования для виртуальных сред Р-Виртуализация.

- У каждого сервера в кластере должен быть задан параметр `SERVER_UUID` в файле `/etc/vz/vz.conf`. ПК Р-Хранилище использует данный параметр для предоставления доступа к дискам контейнеров.

Включение и отключение высокой доступности на серверах

Включение высокой доступности на сервере означает включение ее для всех виртуальных сред, находящихся на данном сервере и хранящихся в кластере ПК Р-Хранилище.

Примечания:

1. Когда вы назначаете сетевую роль iSCSI сетевому интерфейсу физического сервера в панели управления Р-Хранилище, то высокая доступность автоматически включается для виртуальных машин и контейнеров на данном физическом сервере в режиме round-robin.

2. Когда вы включаете высокую доступность для виртуальных машин и контейнеров при выборе роли клиента во время установки ПК Р-Хранилище с управлением при помощи командной строки, высокая доступность автоматически включается в режиме DRS.

Чтобы включить высокую доступность на сервере, необходимо выполнить следующие действия:

1 Обновите ПК Р-Хранилище до последней версии с помощью `yum update`.

2 Выполните скрипт `hastart`. Например, если сервер включен в кластер `stor1`, выполните:

```
# hastart -c <cluster> -n <storage_network/network_mask>
```

  где *<cluster>* является именем кластера, например, `vstor1`, *<storage_network>* - внутренней сетью кластера, а *<network_mask>* включает все серверы в кластере.

  Данный скрипт автоматически установит, настроит и запустит службы высокой доступности на сервере, а также добавит сервер в конфигурацию высокой доступности.

После включения высокой доступности на сервере вы можете посмотреть результат, используя команду `shaman stat`.

Отключение высокой доступности для отдельных виртуальных сред

По умолчанию если высокая доступность включена на сервере, то она автоматически включается для всех виртуальных сред, находящихся на данном сервере. При необходимости можно отключить высокую доступность для отдельных виртуальных машин и контейнеров с помощью команды `prlctl set`. Например:

```
# prlctl set MyVM --ha-enable no
```

Чтобы заново включить высокую доступность, нужно ввести:

```
# prlctl set MyVM --ha-enable yes
```

Включение высокой доступности для целей iSCSI

Примечание: Если вы используете ПК Р-Хранилище с управлением при помощи графического интерфейса, включите высокую доступность для целей iSCSI путем назначения сетевой роли iSCSI сетевому интерфейсу физического сервера (см. *Руководство по эксплуатации ПК Р-Хранилище*). Имейте в виду, что данное действие также включает высокую доступность для виртуальных машин и контейнеров, находящихся на данном сервере и хранящихся в кластере ПК Р-Хранилище.

Если вы используете ПК Р-Хранилище с управлением при помощи командной строки, то высокая доступность для целей iSCSI отключена по умолчанию. Чтобы включить ее на сервере, необходимо выполнить следующие действия:

1. Если высокая доступность еще не включена на сервере, включите ее с помощью скрипта `hastart`, как описано в разделе выше.
2. Добавьте для сервера роль iSCSI, используя утилиту `shaman`. Например, на сервере из кластера `stor1` выполните:

```
# shaman -c stor1 add-role ISCSI
```

Если вы хотите, чтобы высокая доступность работала только для целей iSCSI, измените роли `shaman` на сервере с помощью следующей команды:

```
# shaman -c stor1 set-roles ISCSI
```

Отключение высокой доступности на серверах

Отключение высокой доступности на сервере означает отключение ее для всех виртуальных сред и целей iSCSI, находящихся на данном сервере и хранящихся в кластере ПК Р-Хранилище.

Чтобы отключить высокую доступность на сервере, необходимо выполнить следующие действия:

1 Отключите и остановите службы высокой доступности:

```
# systemctl disable shaman.service
# systemctl stop shaman.service
# systemctl disable pdrs.service
# systemctl stop pdrs.service
```

2 Удалите сервер из конфигурации высокой доступности. Например, для сервера из кластера `vstor1` выполните:

```
# shaman -c stor1 leave
```

Настройка режимов перераспределения ресурсов

Для кластера можно настроить его поведение в случае отказа одного из серверов. Доступно три режима:

- DRS (по умолчанию). В данном режиме виртуальные машины и контейнеры, которые были запущены на отказавшем сервере, перемещаются на исправные серверы на основе доступной свободной памяти и количества виртуальных сред, разрешенного лицензией. Данный режим может использоваться для серверов, на которых запущена служба `pdrs`.

  Примечание: При использовании пулов процессоров виртуальные среды можно переместить только на другие серверы из того же пула. Для получения подробной информации см. **Управление пулами ЦП** (стр. 142).

  Режим DRS работает следующим образом. Мастер DRS постоянно собирает через SNMP данные с каждого исправного сервера в кластере:

  - общий объем ОЗУ сервера,
  - общий объем ОЗУ, используемый виртуальными машинами,
  - общий объем ОЗУ, используемый контейнерами,
  - максимальное число виртуальных машин, которое разрешается одновременно запускать,
  - максимальное число контейнеров, которое разрешается одновременно запускать,
  - максимальное число виртуальных сред, которое разрешается одновременно запускать.

  При отказе сервера служба `shaman` отправляет список виртуальных сред, которые были запущены на данном сервере, мастеру DRS, который сортирует его по требуемому объему ОЗУ. Используя собранные данные по ОЗУ и лицензиям серверов, мастер DRS пытается найти сервер с большим объемом свободной памяти и походящей лицензией для виртуальной среды, возглавляющей список (требующей большой объем ОЗУ). Если такой сервер находится, мастер DRS помечает виртуальную среду для перемещения на данный сервер. В противном случае, он помечает виртуальную среду как неисправную. Затем мастер DRS обрабатывает следующую виртуальную среду в списке, корректируя собранные данные с серверов

требованиями предыдущей виртуальной среды. После обработки всех виртуальных сред из списка мастер DRS отправляет список службе `shaman` для их перемещения.

- Spare. В данном режиме виртуальные машины и контейнеры с отказавшего сервера перемещаются на свободный сервер — пустой сервер, имеющий достаточно ресурсов и подходящую лицензию, позволяющие разместить все виртуальные среды с любого сервера в кластере. Такой сервер требуется для работы высокой доступности в данном режиме.

  Перед переключением на данный режим убедитесь, что свободный сервер добавлен в конфигурацию высокой доступности и не имеет ресурсов (виртуальных машин, контейнеров, целей iSCSI и кластеров S3), хранящихся на нем. Для этого выполните команду `shaman stat` на любом сервере в кластере и проверьте, чтобы у сервера в столбце **RESOURCES** были отображены нули:

```
# shaman stat
Cluster 'stor1'
Nodes: 3
Resources: 12

   NODE_IP       STATUS     ROLES                    RESOURCES
*  10.10.20.1    Active     VM:QEMU,CT:VZ7,ISCSI,S3  0 CT, 0 S3, 0 VM, 0 ISCSI
 M 10.10.20.2    Active     VM:QEMU,CT:VZ7,ISCSI,S3  4 CT, 1 S3, 3 VM, 0 ISCSI
   10.10.20.3    Active     VM:QEMU,CT:VZ7,S3        1 CT, 1 S3, 1 VM, 1 ISCSI
```

  В примере выше текущий сервер (помечен звездочкой) является пустым и может быть использован в качестве свободного сервера.

  Если на сервере есть ресурсы, вы можете освободить его:

  - от виртуальных машин и контейнеров путем их миграции на другие серверы кластера,
  - от целей iSCSI путем отмены их регистрации на данном сервере и регистрации их на других серверах кластера,
  - от ресурсов S3 путем освобождения данного сервера от кластера S3.

  При наличии свободного сервера в кластере вы можете переключиться в этот режим, выполнив команду:

```
# shaman set-config RESOURCE_RELOCATION_MODE=spare
```

- Round-robin (запасной вариант по умолчанию). В данном режиме виртуальные машины, контейнеры и цели iSCSI с отказавшего сервера перемещаются на исправные серверы в порядке круговой очереди. Чтобы переключиться в этот режим, выполните следующую команду:

```
# shaman set-config RESOURCE_RELOCATION_MODE=round-robin
```

Дополнительно можно настроить запасной режим перемещения в случае, если выбранный режим не удастся. Например:

```
# shaman set-config RESOURCE_RELOCATION_MODE=drs,spare
```

Настройка режимов перераспределения ресурсов на серверах, участвующих в экспорте iSCSI

Важно: Следуйте инструкции, приведенной в данном разделе, только если вы используете ПК Р-Хранилище с управлением при помощи графического интерфейса в сценарии, описанном ниже. В противном случае следует пропустить данный раздел.

- Если вы используете ПК Р-Хранилище с графическим интерфейсом и назначили роль **iSCSI** сетевому интерфейсу сервера (см. **Роли сетевых интерфейсов** в *Руководство по установке ПК Р-Виртуализация*), то высокая доступность автоматически включается для виртуальных машин, контейнеров и целей iSCSI на данном физическом сервере в режиме перераспределения ресурсов round-robin.
- Если вы используете ПК Р-Хранилище с графическим интерфейсом и включили сервер в кластер S3 (см. **Создание S3-кластера** в *Руководство по эксплуатации ПК Р-Хранилище*), то высокая доступность автоматически включается для виртуальных машин, контейнеров и ресурсов S3 на данном физическом сервере в режиме перераспределения ресурсов round-robin.

Примечание: Все режимы перераспределения ресурсов описаны в разделе **Настройка режимов перераспределения ресурсов** (стр. 139).

Если вас устраивает режим round-robin, то дальнейшей настройки сервера не требуется. Однако если вы хотите изменить режим перераспределения ресурсов, например, на DRS, то вам необходимо зайти на сервер через SSH и выполнить скрипт `hastart`, как описано в разделе **Включение и отключение высокой доступности на серверах** (стр. 137).

После выполнения скрипта режим высокой доступности автоматически сменится на DRS. Если вам понадобится изменить его снова, см. инструкции в разделе **Настройка режимов перераспределения ресурсов** (стр. 139).

Настройка приоритета высокой доступности для виртуальных сред

Приоритет высокой доступности определяет, какая виртуальная среда будет перемещена первой при отказе сервера, на котором она находится. Чем выше приоритет, тем выше шанс, что виртуальная среда будет перемещена на исправный сервер при нехватке ресурсов диска в кластере ПК Р-Хранилище.

По умолчанию приоритет всех новых виртуальных сред равен 0. Можно использовать команду `prlctl set`, чтобы настроить приоритет высокой доступности для виртуальной среды, например:

```
# prlctl set MyVM1 --ha-prio 1
# prlctl set MyVM2 --ha-prio 2
```

Данные команды указывают для виртуальных машин MyVM1 и MyVM2 значение приоритета высокой доступности 1 и 2 соответственно. При отказе сервера, на котором находятся данные виртуальные машины, MyVM2 будет перемещена на исправный сервер в первую очередь, затем MyVM1 , а потом все остальные виртуальные среды, у которых приоритет по умолчанию равен 0.

Управление пулами ЦП

Внимание: Данная функция является экспериментальной. Libvirt может не знать о новых функциях ЦП, которые могут быть использованы в пулах ЦП. Это может привести к проблемам с миграцией на целевые серверы, не имеющих данные функции ЦП.

ПК Р-Виртуализация позволяет избежать остановки виртуальных сред на сервере (например, для обслуживания сервера) путем временной живой миграции на другой сервер. Чтобы иметь возможность живой миграции, ЦП на исходном и целевом серверах должны иметь одного производителя, а также функции ЦП на целевом сервере должны совпадать с функциями на исходном сервере или быть расширенными.

Данное требование может служить основанием для возникновения двух проблем. Во-первых, если целевой сервер имеет больше функций ЦП, чем исходный сервер, то обратная живая миграция на исходный сервер невозможна. Во-вторых, если сервер из высокодоступного кластера откажет и его виртуальные среды будут перемещены на другой сервер, то целевой сервер может иметь ЦП от другого производителя или с отличным набором функций, что не позволит выполнить живую миграцию обратно на исходный сервер.

Пулы ЦП решают эти две проблемы путем разделения серверов ПК Р-Виртуализация на группы (пулы), в которых всегда возможна живая миграция между двумя любыми серверами. Гарантия миграции достигается при помощи определения общих для всех серверов в пуле функций ЦП и маскирования (отключения) остальных функций ЦП на тех серверах, где их больше. Таким образом, пулом ЦП является группа серверов с одинаковыми функциями ЦП.

Примечание: Добавление серверов с одинаковыми ЦП к разным пулам ЦП не препятствует живой миграции между данными серверами.

Добавление серверов в пулы ЦП

Примечание: Серверы с ЦП от различных производителей, например, Intel и AMD, не могут быть добавлены в один и тот же пул ЦП.

На сервере, который будет добавлен к пулу ЦП, не должно быть запущенных виртуальных сред. Чтобы удовлетворить данное требование и в то же время избежать простоя виртуальных сред, можно выполнить живую миграцию всех запущенных виртуальных сред

на другой сервер (и обратную живую миграцию на сервер после добавления данного сервера к пулу).

Самым простым способом добавить сервер в пул ЦП является выполнение на нем следующей команды:

```
# cpupools join
```

Сервер будет добавлен в стандартный пул ЦП.

Стандартные пулы имеют следующие свойства и ограничения:

- Шаблон наименования: `default_{intel|amd}`*N*, например, `default_intel0`, `default_amd0` и т.п.
- Предварительно установленная, неизменяемая основная маска ЦП обеспечивает максимальную совместимость оборудования в ущерб расширенным функциям ЦП. Разные маски функций ЦП используются для разных производителей ЦП.
- Серверы, которые не поддерживают основную маску функций ЦП, помещаются в разные стандартные пулы ЦП, например, `default_intel1`, `default_amd2` и т.д..
- Серверы нельзя намеренно добавить в определенные стандартные пулы ЦП.

Чтобы убедиться, что у серверов в пуле включено максимальное доступное число общих функций ЦП для лучшей производительности, можно переместить нужные серверы в собственный пул ЦП, выполнив следующие действия:

1 На каждом добавляемом сервере выполните команду `cpupools move`. Например:

```
# cpupools move mypool
```

Сервер будет добавлен в пул ЦП `mypool`. Если данный пул не существует, он будет создан.

Примечание: Собственные пулы ЦП создаются с той же основной маской функций ЦП, что и стандартные пулы.

2 На любом сервере в новом пуле выполните команду `cpupools recalc`, чтобы обновить маску функций ЦП и убедиться, что включено максимальное доступное число общих функций ЦП. Например:

```
# cpupools recalc mypool
```

Теперь, когда сервер находится в нужном пуле ЦП, можно приступить к обратной живой миграции его виртуальных сред.

Основная рекомендация заключается в группировке серверов с процессорами, имеющими одинаковую микроархитектуру, поколение или линию, так как они имеют одинаковые функции ЦП. Таким образом, большинство функций ЦП останется доступными после применения маски функций ЦП к пулу. Данный подход помогает обеспечить наилучшую возможную производительность для серверов, а также гарантирует совместимость при живой миграции.

Мониторинг пулов ЦП

Посмотреть список пулов ЦП, существующих в кластере, и серверов, входящих в них, можно с помощью выполнения команды `cpupools stat` на любом сервере в кластере. Например:

```
# cpupools stat
default_intel0:
320117e17894401a
bec9df1651b041d8
eaea4fc0ddb24597
mypool:
ca35929579a448db
* f9f2832d4e5f4996
```

Перечисленные идентификаторы являются ID серверов ПК P-Хранилище, которые можно получить при помощи команды `shaman -v stat`. Например:

```
# shaman -v stat
Cluster \'vstor1'
Nodes: 5
Resources: 1
   NODE_IP          STATUS    NODE_ID                RESOURCES
   10.29.26.130     Active    bec9df1651b041d8       0 CT
*  10.29.26.134     Active    f9f2832d4e5f4996       0 CT
   10.29.26.137     Active    ca35929579a448db       0 CT
   10.29.26.141     Active    320117e17894401a       0 CT
 M 10.29.26.68      Active    eaea4fc0ddb24597       1 CT
...
```

Примечание: Звездочкой (*) обозначается текущий сервер (сервер, на котором была выполнена команда).

Исключение серверов из пулов ЦП

Для исключения текущего сервера из пула ЦП выполните на нем команду:

```
# cpupools leave
```

Мониторинг статуса кластера

Для мониторинга общего статуса кластера и его ресурсов можно использовать команды `shaman top` и `shaman stat`. Команда `shaman stat` показывает снапшот текущего статуса кластера и его ресурсов, а команда `shaman top` отображает динамическое состояние кластера в реальном времени. Пример ниже показывает вывод команды `shaman stat`:

```
# shaman stat
Cluster \'vstor1'
Nodes: 3
Resources: 8

      NODE_IP       STATUS        RESOURCES
```

```
   M  10.30.24.176 Active       0 CT, 0 VM, 1 ISCSI
 *    10.30.25.33  Active       1 CT, 3 VM, 1 ISCSI
      10.30.26.26  Active       0 CT, 0 VM, 2 ISCSI

       CT ID    STATE    STATUS  OWNER_IP      PRIORITY
       101      stopped  Active  10.30.25.33   0

       VM NAME  STATE    STATUS  OWNER_IP      PRIORITY
       vm1      stopped  Active  10.30.25.33   0
       vm2      running  Active  10.30.25.33   0
       vm3      stopped  Active  10.30.25.33   0

      ISCSI ID                   STATE   STATUS   OWNER_IP     PRIORITY
iqn.2014-04.com.pstorage:ps1     running  Active  10.30.24.176       0
iqn.2014-04.com.pstorage:ps2     running  Active  10.30.25.33        0
iqn.2014-04.com.pstorage:ps3     running  Active  10.30.26.26        0
```

Информация в выводе представлена в виде таблиц со следующими полями:

Поле	Описание
`Cluster`	Имя кластера. Буква "M" рядом с сервером указывает на то, что он является главным в кластере (master-сервер), а знак звездочки (*) обозначает сервер, на котором была выполнена команда `shaman stat`.
`Nodes`	Количество серверов с включенной поддержкой высокой доступности в кластере.
`Resources`	Количество ресурсов, которые контролирует `shaman`.
`NODE_IP`	IP-адрес, назначенный серверу.
`STATUS`	Статус сервера. Может быть одним из следующих: • `Active`. Сервер запущен и контролируется `shaman`. • `Inactive`. Сервер запущен, но `shaman` не контролирует его (например, служба `shamand` остановлена).
`RESOURCES`	Ресурсы сервера.
`CT ID`	ID контейнера.
`VM NAME`	Имя виртуальной машины.
`ISCSI ID`	Имя цели iSCSI.
`STATE`	Указывает, запущена или остановлена виртуальная среда.
`STATUS`	Статус высокой доступности виртуальной машины, контейнера или цели iSCSI (по сообщению `shaman`). Может быть одним из следующих: • `Active`. Исправные виртуальные машины, контейнеры или цели iSCSI в кластере. • `Broken`. Виртуальные машины, контейнеры или цели iSCSI, которые не удалось переместить с отказавшего на исправный сервер. • `Pool`. Виртуальные машины, контейнеры или цели iSCSI, ждущие перемещения с отказавшего на исправный сервер.
`OWNER_IP`	IP-адрес сервера, на котором находится виртуальная машина, контейнер или цель iSCSI.
`PRIORITY`	Текущий приоритет виртуальной среды. Для получения подробной информации см. **Настройка приоритета высокой доступности для виртуальных сред** (стр. 141).

Вывод команды `shaman top` похож на вывод `shaman stat`. Дополнительно можно использовать клавиши клавиатуры для изменения вывода команды по ходу работы. Поддерживаются следующие клавиши:

- **g**: сгруппировать или разгруппировать ресурсы по их статусу.
- **v**: показать или спрятать дополнительную информацию.
- **ENTER** или **SPACE**: обновить экран.
- **q** или **Esc** или **CTRL-C**: выйти.
- **h**: показать справочный экран.

Управление ресурсами кластера с помощью скриптов

В ПК Р-Хранилище входит набор скриптов, которые используются утилитой `shaman` для мониторинга ресурсов кластера и управления ими. Есть два типа скриптов:

- Обычные скрипты. Обычные скрипты находятся в директории `/usr/share/shaman` и используются утилитой `shaman` для вызова скриптов для отдельных ресурсов.
- Скрипты для отдельных ресурсов. Для каждого из обычных скриптов есть один или более скриптов для отдельных ресурсов. Скрипты для отдельных ресурсов предназначаются для каждого ресурса кластера и находятся в отдельных поддиректориях. Для виртуальных машин и контейнеров этими директориями являются `/usr/share/shaman/vm-` и `/usr/share/shaman/ct-` соответственно. Скрипты для отдельных ресурсов используются для выполнения различных действий с ресурсами кластера.

Пример ниже описывает процесс использования скрипта перемещения:

1. Демон `shaman-monitor` проверяет через равные промежутки времени, не требуется ли переместить какие-либо виртуальные среды (обычно при отказе сервера виртуальны среды, находящиеся на нем должны быть перемещены).
2. Если некоторые виртуальные среды ждут перемещения, `shaman-monitor` вызывает обычный скрипт `/usr/share/shaman/relocate`.
3. Обычный скрипт `relocate` вызывает скрипты `/usr/share/shaman/vm-/relocate` и `/usr/share/shaman/ct-/relocate` для перемещения виртуальной среды на исправный сервер.

При необходимости можно настроить любой скрипт для удовлетворения своих требований.

Полный список скриптов и их описание см. на man-старице по `shaman-scripts`.

Глава 9

Дополнительные задачи

Данная глава содержит различные задачи конфигурации и управления, некоторые из которых требуют углубленных знаний Linux и ПК Р-Виртуализация и должны выполняться с осторожностью.

Настройка политик управления автоматической памятью

ПК Р-Виртуализация предлагает несколько техник для автоматического управления памятью виртуальных сред. Управление осуществляется демоном `vcmmd` с помощью политик, описанных ниже. Включение правильной политики может обеспечить лучшую производительность и плотность для ПК Р-Виртуализация.

В текущий момент доступны следующие политики:

- `performance` (по умолчанию). Рекомендуется для физических серверов с некрупными виртуальными средами (относительно размера узла NUMA).

  Важно: Настоятельно не рекомендуется выделять больше ресурсов, чем есть на сервере, когда включена политика `performance`. Если вам нужно выделить больше ресурсов, чем есть на сервере, следует переключиться на политику `density`.

- `density`. Рекомендуется для физических серверов, на которых виртуальным средам выделено больше ресурсов памяти, чем есть на сервере.
- `NoOpPolicy`. Данная политика автоматически устанавливается, если на физическом сервере не установлена лицензия или если срок действия текущей лицензии истек.

Если вы активировали лицензию после установки ПК Р-Виртуализация или обновили лицензию с истекшим сроком действия, то для применения изменений необходимо перезапустить демон `vcmmd`. Для этого:

1. Остановите все виртуальные среды на сервере или временно мигрируйте их онлайн на другой сервер.
2. Выполните следующую команду:

```
# systemctl restart vcmmd
```

3. Запустите виртуальные среды или мигрируйте их обратно на сервер, в зависимости от действия, выполненного в шаге 1.

После перезапуска `vcmmd` по умолчанию будет применена политика `performance`.

Политики можно переключить при помощи инструмента `prlsrvctl`.

Примечание: Перед тем как задать политику VCMMD, необходимо остановить все виртуальные машины и контейнеры на сервере или временно выполнить живую миграцию запущенных виртуальных сред на другой сервер.

Например, для переключения на политику `density`, выполните следующие действия:

1 Убедитесь, что на сервере нет запущенных виртуальных машин или контейнеров:

```
# prlctl list
UUID                                    STATUS    IP_ADDR     T  NAME
```

2 Задайте политику с помощью команды `prlsrvctl set --vcmmd-policy`:

```
# prlsrvctl set --vcmmd-policy density
```

Чтобы проверить, включена ли политика, выполните следующую команду:

```
# prlsrvctl info | grep "policy"
Vcmmd policy: density config=density
```

Оптимизация памяти виртуальных машин с помощью KSM

Для оптимизации использования памяти виртуальными машинами ПК Р-Виртуализация использует функцию Linux под названием Kernel Same-Page Merging (KSM). Демон KSM `ksmd` периодически сканирует память на наличие страниц с одинаковым содержанием и объединяет их в одну страницу. Данная страница помечается как копирование при записи (COW), и когда ее содержимое изменяется виртуальной машиной, ядро создает новую копию для данной виртуальной машины.

KSM позволяет серверу:

- избегать подкачки благодаря объединению идентичных страниц;
- запускать больше виртуальных машин;
- выделять больше памяти виртуальным машинам, чем есть на сервере;
- ускорить работу ОЗУ и, следовательно, некоторых приложений и гостевых ОС.

В ПК Р-Виртуализация KSM управляется с помощью `vcmmd` и работает с политиками `performance` и `density`. Однако если включена политика `performance`, KSM объединяет одинаковые страницы для каждого узла NUMA отдельно, что замедляет его работу во столько раз, сколько узлов NUMA находится на сервере (например, в четыре раза, если на сервере четыре узла NUMA).

Примечание: Для получения подробной информации см. **Настройка политик управления автоматической памятью** (стр. 147).

Управление службами хоста с помощью VCMMD

ПК Р-Виртуализация позволяет задавать лимиты и гарантии памяти службам хоста, которые управляются при помощи `vcmmd`. Демон учитывает настройки памяти, заданные для служб, при выделении ресурсов виртуальным средам.

Примечание: Настройки сбрасываются при перезагрузке физического сервера.

Например, для того чтобы служба `service` управлялась с помощью `vcmmd`, выполните следующие действия:

1. Поместите процессы службы в директорию `/sys/fs/cgroup/memory/service.slice`.
2. Зарегистрируйте `service.slice` в `vcmmd` и задайте желаемые параметры памяти. Например:

```
# vcmmdctl register SRVC service.slice --guarantee 100000000 --limit 100000000 --swap 100000
```

3. Активируйте службу:

```
# vcmmdctl activate service.slice
```

Чтобы проверить, управляется ли служба при помощи `vcmmd`, выполните команду:

```
# vcmmdctl list
name                                type   active   guarantee   limit     swap
<...>
service.slice                       SRVC   yes      97656       97656     97
```

Можно изменять параметры памяти службы “на лету”, используя команду `vcmmdctl update`.

Например, для изменения параметров `service.slice`, выполните команду:

```
# vcmmdctl update service.slice --guarantee 200000000 --limit 200000000
```

Чтобы проверить, верно ли применены изменения, выполните:

```
# vcmmdctl list
name                                type   active   guarantee   limit     swap
<...>
service.slice                       SRVC   yes      195312      195312    max
```

Вы можете отменить регистрацию службы в `vcmmd` “на лету” командой `vcmmdctl unregister`.

Управление службами ПК Р-Хранилище с помощью VCMMD

При загрузке сервера ПК Р-Хранилище автоматически создается `vstorage.slice/vstorage-services.slice` в cgroup памяти.

Если вы развернули ПК Р-Хранилище поверх сервера Р-Виртуализация и запустили службы `vstorage` вручную без перезагрузки сервера, то для применения изменений необходимо перезапустить демон `vcmmd`. Для этого:

1. Остановите все виртуальные среды на сервере или временно мигрируйте их онлайн на другой сервер.
2. Выполните следующую команду:

```
# systemctl restart vcmmd
```

3. Запустите виртуальные среды или мигрируйте их обратно на сервер, в зависимости от действия, выполненного в шаге 1.

После перезапуска `vcmmd` в cgroup памяти будет создан `vstorage.slice/vstorage-services.slice`.

Чтобы проверить, управляется ли `vstorage` при помощи `vcmmd`, выполните команду:

```
# vcmmdctl list
name                                    type   active   guarantee   limit     swap
<...>
vstorage.slice/vstorage-services.slice  SRVC   yes      4194304     103854973 0
```

Можно временно изменять лимиты и гарантии памяти для служб `vstorage` “на лету” с помощью команды `vcmmdctl update`. Например:

```
# vcmmdctl update vstorage.slice/vstorage-services.slice --guarantee 100000000 --limit 100000000
```

При следующем перезапуске `vcmmd` или перезагрузке сервера параметры будут сброшены до стандартных значений.

Чтобы изменить лимиты и гарантии для служб `vstorage` на постоянной основе, необходимо выполнить следующие действия:

1. Измените параметры памяти в файле `/etc/vz/vstorage-limits.conf`. (Для неограниченной памяти укажите `-1`.)
2. Перезапустите `vcmmd`, чтобы применить изменения:

```
# systemctl restart vcmmd
```

Создание специализированных контейнеров

Для использования пользовательских приложений в нескольких идентичных контейнерах можно создать контейнеры с необходимыми приложениями, которые заранее установлены и настроены согласно требованиям.

ПК Р-Виртуализация предлагает несколько способов создания специализированных контейнеров с уже установленными приложениями:

- из образа типа golden (кэш EZ-шаблона ОС с заранее установленными шаблонами приложений),

- из специализированного EZ-шаблона ОС, в котором указан настраиваемый список пакетов приложений,
- из настраиваемого файла образца конфигурации, в котором указаны EZ-шаблоны настраиваемых приложений.

Использование функции образа типа golden

Функция образа типа golden позволяет заранее устанавливать шаблоны приложений в кэш EZ-шаблонов ОС для сокращения времени создания нескольких контейнеров на базе одного набора шаблонов ОС и приложений. Образ типа golden является на текущий момент самым простым и быстрым способом создания контейнеров с заранее установленными приложениями.

Создать подобный кэш можно следующим образом:

1 Создать настраиваемый файл образца конфигурации, содержащий информацию об EZ-шаблоне ОС, который нужно кэшировать, и EZ-шаблонах приложений, которые необходимо заранее установить. Например:

```
# cp /etc/vz/conf/ve-basic.conf-sample /etc/vz/conf/ve-centos-6-x86_64-mysql-
devel.conf-sample
```

> **Примечание:** Если уже существует настраиваемый файл образца конфигурации с указанными в нем EZ-шаблонами приложений, можно повторно его использовать, вместо создания нового образца.

2 Добавить информацию об EZ-шаблоне ОС и EZ-шаблонах приложений в новый файл конфигурации. Каждое имя шаблона ОС и приложений должно начинаться с точки. Несколько последовательных имен EZ-шаблонов приложений должны разделяться пробелами. Например:

```
# cd /etc/vz/conf
# echo 'OSTEMPLATE=".centos-6-x86_64"' >> ve-centos-6-x86_64-mysql-devel.conf-sample
# echo 'TEMPLATES=".mysql .devel"' >> ve-centos-6-x86_64-mysql-devel.conf-sample
```

3 Выполнить команду `vzpkg create appcache` с указанием файла конфигурации в качестве параметра. Например:

```
# vzpkg create appcache --config centos-6-x86_64-mysql-devel
```

> **Примечание:** Если итоговый кэш приложений уже существует, он не будет создаваться повторно, и появится соответствующее сообщение. Чтобы повторно создать кэш приложений, необходимо использовать команду `vzpkg update appcache`.

Итоговый архив будет помещен в директорию `/vz/template/cache` на физическом сервере. Проверить его наличие и содержание в нем нужных шаблонов приложений можно с помощью следующей команды:

```
# vzpkg list appcache
centos-6-x86_64             2012-07-20 16:51:36
     mysql
     devel
```

Отключение функции образа типа golden

Функция образа типа golden включена по умолчанию в глобальном конфигурационном файле `/etc/sysconfig/vz/vz.conf`. Для ее отключения можно выполнить одно из следующих действий:

- Указать `no` для параметра `GOLDEN_IMAGE` в глобальном конфигурационном файле. Функция образа типа golden будет отключена глобально для всего сервера.
- Указать `no` для параметра `GOLDEN_IMAGE` в файле образца конфигурации контейнера. Функция образа типа golden будет отключена для команд, которые использует данный определенный файл образца конфигурации.
- Создать файл с именем `golden_image`, содержащий `no` в конфигурационном каталоге EZ-шаблона ОС. Функция образа типа golden будет отключена для данного определенного EZ-шаблона ОС.
- Создать файл с именем `golden_image`, содержащий `no` в конфигурационном каталоге шаблона приложения. Функция образа типа golden будет отключена для данного определенного шаблона приложения, таким образом, он не будет установлен заранее в кэш EZ-шаблонов ОС.

Использование специализированных EZ-шаблонов

Можно создавать специализированные шаблоны ОС и приложений с учетом своих потребностей. В подобном шаблоне необходимо только указать параметры, которые отличаются от параметров стандартного шаблона. Все остальные параметры, которые не задаются в специализированного шаблоне, берутся из соответствующего стандартного шаблона.

Для создания пользовательского шаблона нужно выполнить следующие действия:

1. Если потребуется, установить стандартный шаблон ОС на физический сервер. Например:

```
yum install centos-7-x86_64-ez
```

2. Создать директорию для своего шаблона в том же месте, где находится директория стандартных шаблонов. Например, для специализированного 64-битного шаблона CentOS `mytmpl` нужно создать директорию `/vz/template/centos/7/x86_64/config/os/mytmpl`.

3. При создании специализированного шаблона ОС, нужно указать репозитории. Например, копировать файл `mirrorlist` из директории стандартных шаблонов в свою директорию шаблонов:

```
# cp /vz/template/centos/7/x86_64/config/os/default/mirrorlist \
/vz/template/centos/7/x86_64/config/os/mytmpl
```

4. В своей директории шаблонов создать файл `packages` со списком нужных RPM-пакетов, по одному на каждой строчке. Например:

```
systemd
yum
```

Примечание: Минимальный список пакетов, который нужно включить в специализированный шаблон, может отличаться в зависимости от гостевой ОС. Например, для шаблонов CentOS 7 необходимо указать `systemd` в файле `packages`, чтобы команда `prlctl enter` работала с итоговыми контейнерами.

5 Также можно, но необязательно, изменить другие параметры шаблона с учетом своих потребностей (описание параметров приведено в следующем разделе).

Специализированный шаблон готов. В данном примере получился шаблон ОС, содержащий `systemd`, `yum` и все их необходимые требования. Теперь можно приступить к созданию контейнеров по данному шаблону. Например:

```
# prlctl create MyCT --vmtype ct --ostemplate centos-7-x86_64-mytmpl
```

После создания шаблона приложения его можно добавить в конфигурационный файл контейнера, как описано в разделе **Использование функции образа типа golden** (стр. 151).

Конфигурационные файлы EZ-шаблонов

Все EZ-шаблоны хранятся в `/vz/template`, в поддиректориях, названных в соответствии с именем, версией и архитектурой ОС. Например, `/vz/template/centos/7/x86_64`. Каждый шаблон содержит набор конфигурационный файлов, находящихся в поддиректории `/config/os/`<*template_name*> (шаблоны ОС) или поддиректории `/config/app/`<*template_name*> (шаблоны приложений).

В конфигурационной поддиректории шаблона могу быть следующие файлы:

- `ct2vm` – Скрипт миграции контейнера в виртуальную машину.
- `description` – Подробная информация об EZ-шаблоне.
- `distribution` – Только для шаблонов ОС. Имя дистрибутива Linux, для которого был создан EZ-шаблон.
- `environment` — Только для шаблонов ОС. Список переменных среды, указанных в виде <*key*>=<*value*>.
- `mirrorlist` — Дает ссылку на файлы со списками репозиториев, из которых можно загрузить пакеты в EZ-шаблон.
- `osrelease` — Только для шаблонов ОС. Содержит собственную версию ядра дистрибутива CentOS 7.
- `package_manager` — Только для шаблонов ОС. Указывает систему управления пакетами, использованную для обработки EZ-шаблона.
- `packages` — Содержит список имен пакетов, включенных в соответствующий EZ-шаблон.
- `pre-cache`, `post-cache` — Только для шаблонов ОС. Скрипты, выполняющиеся перед и после установки пакетов в EZ-шаблон на физическом сервере.
- `pre-install`, `post-install` — Скрипты, выполняющиеся внутри контейнера перед и после транзакции управления пакетами.

- `pre-install-hn`, `post-install-hn` — Скрипты, выполняющиеся на физическом сервере перед и после транзакции управления пакетами.
- `pre-upgrade`, `post-upgrade` — Только для шаблонов ОС. Скрипты, выполняющиеся перед и после обновления пакетов внутри контейнера.
- `pre-remove`, `post-remove` — Скрипты, выполняющиеся перед и после удаления EZ-шаблона приложения или пакета из контейнера.
- `release` — Содержит номер выпуска шаблона.
- `repositories` — Содержит список репозиториев, где хранятся пакеты в EZ-шаблоне.
- `summary` — Краткие сведения об EZ-шаблоне.
- `upgradable_versions` — Только для шаблонов ОС.
- `version` — Содержит номер версии шаблона.

Создание RPM-пакетов специализированных EZ-шаблонов

Чтобы сделать специализированный EZ-шаблон общим для нескольких физических серверов, можно создать RPM-пакет с ним следующим образом:

1. Загрузите исходник стандартного шаблона ОС по ссылке http://download.openvz.org/virtuozzo/releases/7.0/source/SRPMS.
2. Измените шаблон согласно своим требованиям, например, измените параметры шаблона ОС, добавьте, измените или удалите шаблоны приложений и др.
3. Создать RPM-пакет из файла `.spec` в чистой среде при помощи стандартных инструментов. Не рекомендуется создавать одновременно более одного шаблона.

Установка Docker в контейнеры ПК Р-Виртуализация

Текущая версия ПК Р-Виртуализация поддерживает контейнеры Docker внутри контейнеров Р-Виртуализация. Вы можете установить Docker в контейнер Р-Виртуализация из шаблона приложения Docker из официального репозитория Docker, как описано в *Руководстве по установе Docker*, или из наиболее понравившегося репозитория ОС. При установке Docker из репозитория Docker или ОС необходимо убедиться, что ваша версия Docker использует драйвер хранения overlayfs (для получения подробной информации о драйверах хранения Docker см. по ссылке https://docs.docker.com/engine/userguide/storagedriver/selectadriver).

ПК Р-Виртуализация позволяет запускать Docker в контейнерах, созданных на базе CentOS 7, Debian 8, Ubuntu 14.04 и 16.04. Для запуска контейнеров Docker контейнеру Р-Виртуализация нужно сетевое соединение, а также достаточный размер свободного места на диске и оперативной памяти.

Чтобы установить Docker в контейнер Р-Виртуализация, необходимо выполнить следующие действия:

1 Создайте контейнер Р-Виртуализация на основе одной из вышеперечисленных гостевых ОС. Например:

```
# prlctl create MyCT --vmtype ct --ostemplate centos-7-x86_64
```

2 Настройте сеть в контейнере Р-Виртуализация:

```
# prlctl set MyCT --netif_add eth0
# prlctl set MyCT --ifname eth0 --dhcp yes --network Bridged
```

Для получения дополнительной информации см. **Настройка сети** (стр. 27).

3 Запустите контейнер Р-Виртуализация. Например:

```
# prlctl start MyCT
```

4 Установите Docker в контейнер Р-Виртуализация:

```
# prlctl exec MyCT yum install docker -y
```

5 Запустите демон Docker:

```
# prlctl exec MyCT systemctl start docker.service
```

Контейнер Р-Виртуализация готов для запуска контейнеров Docker.

Чтобы проверить, правильно ли установлен Docker, можно создать и запустить тестовые контейнеры Docker в контейнере Р-Виртуализация. Например, MySQL и Wordpress:

1 При необходимости увеличьте оперативную память и дисковое пространство. Например:

```
# prlctl set MyCT --memsize 4G --device-set hdd0 --size 50G
```

2 Запустите MySQL:

```
# prlctl exec MyCT docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=123qwe -d mysql
```

3 Запустите WordPress:

```
# prlctl exec MyCT docker run --name test-wordpress --link test-mysql:mysql -p 8080:80
-d wordpress
```

4 Перейдите по IP-адресу контейнера Р-Виртуализация и порту 8080. Должен появиться стандартный экран установки WordPress.

Установка Docker для запуска в режиме роя

ПК Р-Виртуализация поддерживает запуск Docker в режиме роя (swarm) внутри контейнеров Р-Виртуализация. Режим роя включается автоматически путем либо создания роя, либо включения в существующий рой. Рой представляет собой кластер узлов Docker.

Для того чтобы иметь возможность запустить Docker в режиме роя, необходимо установить Docker в контейнер Р-Виртуализация следующим образом:

1 Выполните шаги 1-4 из инструкции, приведенной в разделе **Установка Docker в контейнеры ПК Р-Виртуализация** (стр. 154).

2 Создайте файл `.dockerenv` в контейнере:

```
# prlctl exec MyCT touch  /.dockerenv
```

3 Загрузите модуль `ip_vs_ftp` на сервер:

```
# modprobe ip_vs_ftp
```

4 Запустите демон Docker:

```
# prlctl exec MyCT systemctl start docker.service
```

После установки Docker в контейнер Р-Виртуализация можно приступить к созданию роя. Далее см. Getting started with swarm mode.

Ограничения для Docker в контейнерах ПК Р-Виртуализация

1 ПК Р-Виртуализация не поддерживает проверку в контрольных точках и живую миграцию контейнеров с Docker внутри.

2 ПК Р-Виртуализация поддерживает только драйвер хранения overlayfs для Docker внутри контейнеров Р-Виртуализация.

3 Модули и сторонние надстройки, зависящие от операций, которые запрещены в контейнерах (загрузка модулей ядра, монтирование блочных устройств, прямой доступ к физическому оборудованию), могут не работать в контейнерах.

Следует обратиться в службу технической поддержки Р-Платформа при возникновении проблем, связанных с Docker.

Управление шифрованием виртуального жесткого диска контейнеров

ПК Р-Виртуализация предлагает возможность шифрования виртуального жесткого диска контейнеров на основе `dm-crypt` и `cryptsetup`. В текущей реализации используется алгоритм шифрования AES-256. Механизм шифрования отделен от управления ключами шифрования, что позволяет использовать собственную систему управления ключами (KMS) для выдачи и управления ключами шифрования.

Важно: Доступ к операциям шифрования на хосте должен быть только у пользователя root.

Процедура шифрования может быть полностью описана следующим образом. От конечного пользователя поступает запрос на шифрование диска контейнера (создание нового контейнера с зашифрованным диском, шифрование существующего диска и т.д.). KMS хранит закрытый ключ и ID открытого ключа шифрования, назначенные конечному пользователю. Администратор (или средство автоматизации) получает ID ключа шифрования конечного пользователя от KMS и передает его соответствующей команде `prlctl`. Команда `prlctl` передает ID ключа исполняемому модулю `getkey`, который обращается к KMS и возвращает ключ шифрования, который затем передается `cryptsetup`. Инструмент `cryptsetup` выдает уникальный главный ключ для диска контейнера, зашифровывает содержимое диска с помощью главного ключа и затем зашифровывает главный ключ, хранящийся в заголовке LUKS диска контейнера, используя ключ шифрования от `getkey`.

Двойное шифрование экономит ресурсы сервера в таких ситуациях, когда нужно заменить ключ шифрования (например, в целях ротации ключей). В подобных случаях требуется заново зашифровать только главный ключ в заголовке LUKS, вместо всего содержимого диска.

Единственным шагом настройки, который необходимо выполнить прежде, чем приступить к использованию возможностей шифрования дисков в ПК Р-Виртуализация, является установка генератора запросов для ключей шифрования, как описано ниже, для получения ключей шифрования по их ID для соответствующих команд `prlctl`.

Установка генератора запросов для ключей шифрования

Механизм шифрования обращается к KMS следующим образом: выполняет файл `/usr/libexec/ploop/crypt.d/getkey` с единственным строковым параметром–ID ключа шифрования–и считывает возвращаемое значение ключа шифрования из стандартного вывода. `getkey` может быть скриптом, который вызывает бинарный файл KMS и передает ему указанный ID ключа шифрования.

При успешном выполнении операции исполнительный модуль должен завершить работу с нулевым кодом, а значение ключа должно быть напечатано в стандартном выводе в двоичной форме. При неудачном исходе операции скрипт должен завершить работу с ненулевым кодом.

Примечание: Значение ключа может содержать произвольные байты (например, `\x00`, `\n` и т.п.), которые могут по-разному обрабатываться различными скриптовыми языками. Например, если назначить значение ключа переменной Bash, то нулевые байты будут отделены от него, например, `key_ value\x00lue` станет `key_value`.

Чтобы установить генератор запросов для ключей шифрования на сервере ПК Р-Виртуализация, нужно поместить скрипт `/usr/libexec/ploop/crypt.d/getkey` и сделать его выполнимым и доступным только для пользователя root:

```
# chown root:root /usr/libexec/ploop/crypt.d/getkey
# chmod 700 /usr/libexec/ploop/crypt.d/getkey
```

Шифрование и дешифрование виртуальных жестких дисков контейнеров

Ниже приведен список операций, связанных с шифрованием, которые можно выполнить с виртуальными жесткими дисками контейнеров. Для всех операций, кроме дешифрования, нужен ID ключа шифрования, полученный от KMS.

Примечание: Для зашифрованных дисков утилита `pfcache` отключена.

- Создание контейнера с зашифрованным корневым диском.

```
# prlctl create <CT_name|CT_UUID> --vmtype ct --encryption-keyid <key_id>
```

- Добавление нового зашифрованного диска в контейнер.

```
# prlctl set <CT_name|CT_UUID> --device-add hdd --encryption-keyid <key_id>
```

- Шифрование незашифрованного диска. В процессе данной операции создается новый чистый зашифрованный диск, данные перемещаются на него с незашифрованного диска, который затем очищается без возможности восстановления при помощи `shred` (если не указан параметр `--no-wipe`). Для выполнения операции на хосте должно быть свободное дисковое пространство, равное размеру диска контейнера, который нужно зашифровать.

```
# prlctl set <CT_name|CT_UUID> --device-set hdd0 --encrypt --encryption-keyid <key_id>
[--no-wipe]
```

- Изменение ID ключа шифрования для зашифрованного диска, включая или исключая повторное шифрование содержимого целого диска. По умолчанию заново шифруется только заголовок LUKS зашифрованного диска для экономии ресурсов сервера. Чтобы повторно зашифровать содержимое целого диска, нужно добавить к команде параметр `--reencrypt`. В процессе данной операции создается новый чистый зашифрованный диск, данные перемещаются на него со старого зашифрованного диска, который затем очищается без возможности восстановления при помощи `shred` (если не указан параметр `--no-wipe`). Для выполнения операции на хосте должно быть свободное дисковое пространство, равное размеру диска контейнера, который нужно повторно зашифровать.

```
# prlctl set <CT_name|CT_UUID> --device-set hdd0 --encryption-keyid <key_id> [--
reencrypt] [--no-wipe]
```

- Дешифрование зашифрованного диска контейнера. В процессе данной операции создается новый чистый незашифрованный диск, и данные перемещаются на него с зашифрованного диска. В случае успешного выполнения операции зашифрованный диск удаляется. В случае неудачи частично дешифрованные данные удаляются без возможности восстановления при помощи `shred` (если не указан параметр `--no-wipe`), и зашифрованный диск остается в неизмененном виде. Для выполнения операции на хосте должно быть свободное дисковое пространство, равное размеру диска контейнера, который нужно дешифровать.

```
# prlctl set <CT_name|CT_UUID> --device-set hdd0 --decrypt [--no-wipe]
```

Шифрование системной подкачки

Даже если контейнеры зашифрованы, их память можно подкачать к незашифрованному разделу подкачки в системе. На серверах, где предполагается запускать зашифрованные контейнеры, рекомендуется также зашифровать раздел подкачки.

Для шифрования раздела подкачки со случайным ключом шифрования, необходимо выполнить следующие действия:

1 Определите разделы подкачки в системе:

```
# swapon -s
Filename Type Size Used Priority
/dev/<partition> partition XXXXXXX 0 -1
```

2 Добавьте запись раздела подкачки в файл `/etc/crypttab`. Например:

```
swap /dev/<partition> /dev/urandom swap,noearly
```

В начале процесса шифрования данная запись подсоединит `/dev/<partition>` к `/dev/mapper/swap` в качестве зашифрованного раздела подкачки.

Примечание: Для создания ключа шифрования можно использовать `/dev/random` вместо `/dev/urandom`. Однако в этом случае загрузка системы может занять продолжительное время.

3 Узнайте UUID раздела подкачки:

```
# blkid /dev/<partition>
/dev/<partition>: UUID="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" TYPE="swap"
```

4 Закомментируйте запись раздела подкачки в файле `/etc/fstab` и добавьте запись для подсоединенного файла `/dev/mapper/swap`. Например:

```
#UUID=<partition_UUID> swap swap defaults 0 0
/dev/mapper/swap none swap sw 0 0
```

5 Перезагрузите систему, чтобы приступить к шифрованию.

6 Выполните команду `lsblk` , чтобы проверить, зашифрован ли раздел подкачки. В выводе параметр TYPE раздела подкачки должен иметь значение `crypt`.

```
# lsblk
NAME                      MAJ:MIN      RM     SIZE   RO     TYPE   MOUNTPOINT
...
 └─<swap_partition>       X:X          0      XG     0      crypt  [SWAP]
...
```

В выводе команды `swapon -s` будет отображено другое имя файла:

```
# swapon -s
Filename      Type          Size          Used   Priority
/dev/dm-X     partition     XXXXXXX       0      -1
```

Предоставление доступа по VNC к виртуальным средам

Для подключения к виртуальным средам и управления ими можно использовать любые VNC-клиенты после выполнения следующих шагов:

1 (Рекомендуется) Защитить VNC-соединения на сервере с помощью SSL.

2 Предоставление доступа по VNC для желаемой виртуальной машины или контейнера.

3 Подключение к виртуальной среде с помощью VNC-клиента.

Шаги подробно описаны ниже.

Защита VNC-соединения с помощью SSL

Чтобы установить SSL для всех VNC-соединений на сервере, необходимо выполнить следующие действия:

1 Получите SSL-сертификат и ключ из доверенного центра сертификации.

2 Настройте VNC-сервер для использования сертификата и ключа:

```
# prlsrvctl set --vnc-ssl-certificate <path_to_crt_file> --vnc-ssl-key
<path_to_key_file>
```

Для отключения шифрования VNC выполните команду с пустыми аргументами. Например:

```
# prlsrvctl set --vnc-ssl-certificate '' --vnc-ssl-key ''
```

Предоставление доступа по VNC к виртуальным машинам

Для предоставления доступа по VNC к виртуальной машине необходимо выполнить следующие шаги:

1. Включить поддержку VNC в виртуальной машине.
2. Указать номер порта TCP на физическом сервере, который будет использоваться для прослушивания VNC-соединений для виртуальной машины.

   Примечание: Для каждой виртуальной машины, к которой необходимо подключиться по VNC, нужно указать уникальный номер порта.

3. Установить пароль для обеспечения безопасности VNC-соединения.

Данные операции можно выполнить одной командой. Например:

```
# prlctl set MyVM --vnc-mode manual --vnc-port 5901 --vnc-passwd XXXXXXXX
```

Изменения будут применены к виртуальной машине при ее следующем запуске.

Предоставление доступа по VNC к контейнерам

Для предоставления доступа по VNC к контейнеру нужно выполнить следующие шаги:

1. Убедиться, что в контейнере есть действительная учетная запись, чтобы иметь возможность в него войти.
2. Убедиться, что контейнер запущен.
3. Установить VNC-режим и пароль для контейнера. Например:

```
# prlctl set MyCT --vnc-mode manual --vnc-port 6501 --vnc-passwd XXXXXXXX
```

Примечание: Для каждого контейнера, к которому предоставляется доступ по VNC, нужно указать уникальный номер порта. В автоматическом режиме правильные номера портов назначаются автоматически. В режиме ручной настройки необходимо самостоятельно убедиться, что номера портов уникальны.

Подключение с помощью VNC-клиента

После предоставления доступа по VNC виртуальной среде можно подключиться к ней с помощью любого VNC-клиента. Для подключения нужно передать VNC-клиенту следующие параметры:

- IP-адрес сервера, на котором находится виртуальная среда.
- Номер порта и пароль, которые были указаны при предоставлении доступа по VNC.
- Действительная учетная запись в виртуальной среде.

Управление модулями iptables

В данном разделе описано, как управлять модулями `iptables` как для физических серверов, так и для контейнеров.

Использование модулей iptables в ПК Р-Виртуализация

Фильтрация сетевых пакетов на физическом сервере с ПК Р-Виртуализация аналогична фильтрации, выполняемой на обычном сервере Linux. Можно использовать стандартный инструмент `iptables` , чтобы контролировать вход, перемещение и выход сетевых пакетов в сетевом стеке внутри ядра ПК Р-Виртуализация.

По умолчанию трассировка соединений отключена на физическом сервере. Настройка правил `iptables`, которые требуют функции `conntrack`, позволяет трассировать новые соединения и делает сервер уязвимым для DoS-атак, так как число слотов `conntrack` ограничено. Однако настройка данных правил для определенных виртуальных сред (например, для NAT) оставляет доступ к другим виртуальным средам и физическому серверу в случае DoS-атаки.

Примечание: Если функция `conntrack` включена для контейнера, ее невозможно отключить без перезагрузки сервера или перезапуска данного контейнера.

Чтобы обнаружить трассировку активных соединений на физическом сервере, проверьте, содержит ли файл `/proc/net/nf_conntrack` какие-либо записи:

```
# cat /proc/net/nf_conntrack
```

Для получения справки по использованию `iptables` в серверах Linux можно использовать ресурсы ниже:

- *Red Hat Enterprise Linux 7 Security Guide* содержит раздел, содержащий основы фильтрации пакетов и объясняющий различные доступные параметры для `iptables`.
- *iptables Tutorial 1.2.2* подробно объясняет устройство и работу `iptables`.

Использование модулей iptables в контейнерах

Для использования модулей `iptables` в контейнерах требуется дополнительная настройка со стороны пользователя.

Настройка модулей iptables

Задать состояние модулям `iptables` для резервного копирования/восстановления или живой миграции можно при помощи команды `prlctl set --netfilter`. Если некоторые из модулей `iptables`, разрешенных для контейнера, не загрузились на физический

сервер, на который данный контейнер был восстановлен или мигрирован, то они будут автоматически загружены при запуске контейнера. Например:

```
# prlctl set MyCT --netfilter stateful
```

С помощью данной команды все модули, кроме относящихся к NAT, будут разрешены и загружены для контейнера `MyCT` (при необходимости) на физический сервер, куда данный контейнер был восстановлен или мигрирован.

Примечание: Настройкой по умолчанию является `full`, что разрешает все модули.

Использование правил conntrack и таблиц NAT

Чтобы ограничить максимальное число слотов conntrack, доступных для каждого контейнера на физическом сервере, необходимо указать значение для переменной `net.netfilter.nf_conntrack_max`. Например:

```
# sysctl -w net.netfilter.nf_conntrack_max=50000
```

Значение `net.netfilter.nf_conntrack_max` не может превышать значение `net.nf_conntrack_max`.

Примечание: Даже если контейнер подвергнется DoS-атаке и все его слоты conntrack будут использованы, это не повлияет на другие контейнеры, они смогут создать максимально такое число соединений, какое указано в параметре `net.netfilter.nf_conntrack_max`.

Использование SCTP в контейнерах и виртуальных машинах

ПК Р-Виртуализация поддерживает Stream Control Transmission Protocol (SCTP) в контейнерах и виртуальных машинах. В виртуальных машинах SCTP можно настроить и использовать, как на любом физическом сервере. В контейнерах по умолчанию поддержка SCTP отключена. Чтобы включить SCTP в контейнерах, нужно загрузить модуль ядра `sctp` на физическом сервере командой:

```
# modprobe sctp
```

После загрузки модуля можно создавать и использовать сокеты SCTP в контейнерах при помощи стандартных инструментов Linux.

При необходимости добавьте запись `sctp` в файл `/etc/modules-load.d/vz.conf` для автоматической загрузки модуля ядра `sctp` при загрузке физического сервера.

Примечание: Живая миграция через SCTP не поддерживается.

Создание конфигурационных файлов для новых дистрибутивов Linux

Конфигурационные файлы дистрибутивов используются для различия контейнеров с разными версиями Linux и определения, какие скрипты должны запускаться при выполнении соответствующих операций, относящихся к контейнерам (например, назначение нового IP-адреса контейнеру).

Все дистрибутивы Linux, поставляемые вместе с ПК Р-Виртуализация, имеют конфигурационные файлы, хранящиеся в директории `/usr/libexec/libvzctl/dists/` на физическом сервере. Однако можно создать собственные конфигурационные файлы дистрибутивов для поддержки новых выпущенных версий Linux. Например, для запуска в контейнерах дистрибутив Linux CentOS 7 нужно создать конфигурационный файл дистрибутива `centos-7.conf`, чтобы определить, какие скрипты будут запускаться при выполнении основных задач с контейнерами с данной версией Linux. Чтобы создать данный конфигурационный файл, необходимо выполнить следующие действия:

1. В конфигурационном файле контейнера (с именем `/etc/vz/conf/<UUID>.conf`) указать значение `centos-7` для переменной `DISTRIBUTION` (например, `DISTRIBUTION="centos-7"`).
2. Создать конфигурационный файл `centos-7.conf` в директории `/usr/libexec/libvzctl/dists/`. Самый простой способ—это скопировать один из имеющихся конфигурационных файлов путем выполнения следующей команды в директории `/usr/libexec/libvzctl/dists/`:

```
# cp fedora.conf centos-7.config
```

   В примере выше предполагается, что файл `fedora.conf` существует в директории `/usr/libexec/libvzctl/dists/` на физическом сервере. В противном случае, можно использовать любой другой конфигурационный файл дистрибутива, доступный на сервере.

3. Открыть файл `centos.conf` для редактирования и в правой части первой записи указать имя скрипта, который будет выполняться при запуске команды `prlctl` с параметром, указанным в левой части данной записи. Например, если нужно, чтобы скрипт выполнялся при назначении нового IP-адреса контейнеру и имя скрипта `my_centos_script`, то запись должна выглядеть следующим образом:

```
ADD_IP=my_centos_script-add_ip.sh
```

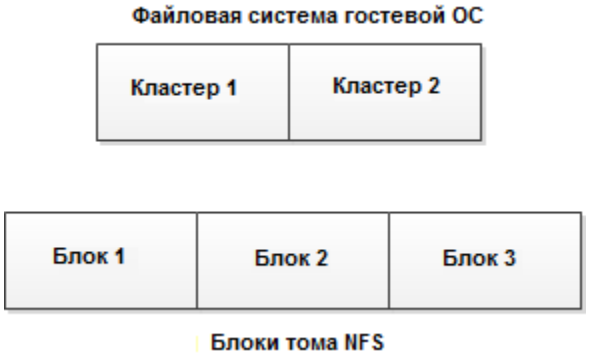
4. Повторить **Шаг 3** для всех записей в файле.
5. Поместить скрипты для нового дистрибутива Linux в директорию `/usr/libexec/libvzctl/dists/scripts` на сервере. Следует убедиться, что имена данных скриптов соответствуют именам в файле `centos-7.conf`.

Выравнивание дисков и разделов в виртуальных машинах

Большинство современных операционных систем автоматически выравнивают разделы при установке в виртуальные машины. Например, Windows Server 2008 создает по умолчанию смещение разделов на 1024 КБ для удовлетворения требований к выравниванию разделов. На рисунке ниже изображен пример правильного выравнивания разделов:

В данном примере любой кластер (наименьшая единица данных) в файловой системе гостевой ОС выровнен по границам блока NFS, и для операций чтения из кластера или записи в него требуется только доступ к одному блоку NFS. Например, операция чтения из Кластера 1 вызывает только одну операцию чтения из Блока 1.

В то же время, виртуальные машины с устаревшими системами (например, Windows Server 2008 или Red Hat Enterprise Linux 5) обычно имеют неправильно выровненные разделы, как показано на рисунке ниже:



В данном примере кластеры файловой системы гостевой ОС обычно не совпадают с границами блоков NFS, и для операций чтения из кластера или записи в него требуется доступ к нескольким блокам NFS. Например, операция чтения из Кластера 1 вызывает две

операции чтения: из Блока 1 и из Блока 2. Неправильно выровненные разделы приводят к увеличению времени на операции чтения и снижению производительности.

Выравнивание разделов

В основном, для выравнивания дисков и разделов в виртуальных машинах необходимо создать смещение, чтобы кластеры в файловой системе гостевой ОС соответствовали размеру блоков тома в хранилище NFS. Обычно размер блоков во многих сетевых хранилищах равен 512 байтам или кратен 512 байтам. В качестве примера разделы ниже описывают процедуру выравнивания дисков и разделов для виртуальных машин Linux и Windows, если считать, что размер блоков NFS составляет 512 байтов.

При выравнивании дисков и разделов следует учитывать, что этот процесс уничтожает все данные на дисках и разделах. Поэтому, если необходимо получить правильно выровненный системный раздел, нужно выровнять диски и разделы перед созданием виртуальной машины и установкой в нее операционной системы. Если правильно выровненный системный раздел не нужен, можно сначала создать виртуальную машину и установить гостевую ОС в нее, а затем выровнять диски данных из виртуальной машины.

В разделах ниже объясняется, как выровнять диски и разделы перед началом установки гостевой ОС. Однако можно использовать похожую процедуру для выравнивания дисков и разделов из виртуальных машин.

Проверка выравнивания разделов в виртуальных машинах

Прежде всего, можно проверить, выровнены ли разделы виртуальной машины. В зависимости от установленной в виртуальной машине операционной системы, можно выполнить следующие действия.

В виртуальных машинах Linux

Для проверки выравнивания разделов в виртуальной машине Linux нужно осуществить вход в данную виртуальную машину и выполнить следующую команду:

```
# fdisk -l -u /dev/device_name
```

Например, проверить, выровнены ли разделы на устройстве sdc, можно с помощью команды:

```
# fdisk -l -u /dev/sdc
Disk /dev/sdc: 73.0 GB, 73014444032 bytes
255 heads, 63 sectors/track, 8876 cylinders, total 142606336 sectors
Units = sectors of 1 * 512 = 512 bytes
   Device    Boot   Start        End    Blocks   Id   System
/dev/sdc1    *         63     208844    104391   83   Linux
/dev/sdc2           208845  142592939   192047+   8e   Linux LVM
```

Следует обратить внимание на число секторов в колонке **Start**. Обычно сектор состоит из 512 байтов, что составляет 32256 байтов для 63 секторов раздела `/dev/sdc1` и 26105625 байтов для 208845 секторов раздела `/dev/sdc2`. Чтобы раздел был правильно выровнен, он должен совпадать с границами 4096 байтов (если считать, что размер блока в

хранилище равен 4 КБ). Так как 32256 и 106928640 не кратны 4096, разделы `/dev/sdc1` и `/dev/sdc2` не выровнены должным образом. Для их выравнивания необходимо сместить

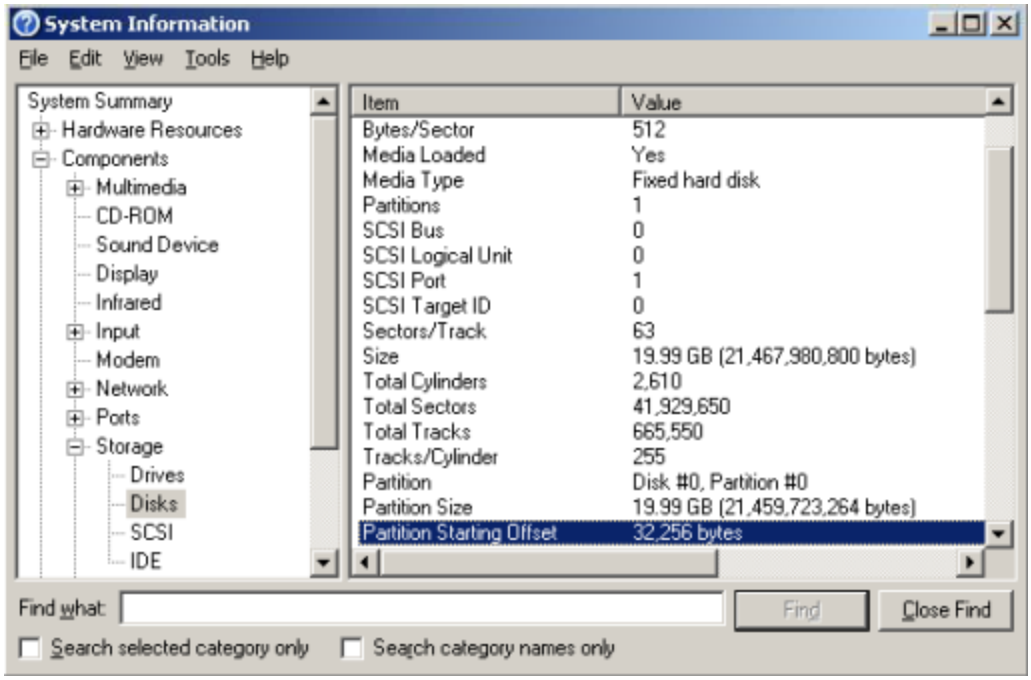
- раздел `/dev/sdc1` на 1 сектор, чтобы он начинался с 64. Тогда 64 сектора, содержащие в каждом 512 байтов, составят 32768, что является кратным 4096.
- раздел `/dev/sdc2` на 3 сектора, чтобы он начинался с 208848. Тогда 208848 сектора, содержащие в каждом 512 байтов, составят 106930176, что является кратным 4096.

В виртуальных машинах Windows

Для проверки выравнивания разделов в виртуальной машине Windows необходимо выполнить следующие действия:

1 Щелкнуть **Start** > **Run**, ввести `msinfo32.exe` и нажать **Enter** , чтобы открыть окно **System Information**.

2 Выбрать **Components** > **Storage** > **Disks** в левой части окна и проверить поле **Partition Starting Offset** в правой части окна.



Чтобы узнать, правильно ли выровнен раздел, можно использовать метод, описанный в предыдущем разделе для виртуальных машин Linux.

Выравнивание дисков для виртуальных машин Linux

Чтобы выровнять разделы для виртуальной машины Linux, нужна рабочая виртуальная машина Linux. В данной виртуальной машине необходимо выполнить следующие действия:

1 Создать новый диск для виртуальной машины.

   На данном диске нужно создать выровненные разделы. Затем необходимо подключить диск к новой виртуальной машине и установить на диск гостевую ОС Linux.

2 Запустить виртуальную машину и войти в нее при помощи SSH.

3 Запустить утилиту `fdisk` для диска, который нужно выровнять.

4 Создать первичный раздел и задать начальный номер блока для созданного раздела.

5 Повторить **Шаги 3-4** для создания и выравнивания всех разделов, которые будут в новой виртуальной машине.

В примере ниже создается раздел #1 размером в 1 ГБ на устройстве `/dev/sda` и смещается на 64 КБ.

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.
The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
      (e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
Command (m for help): n
Command action
      e      extended
      p      primary partition (1-4)
p
Partition number (1-4): 1
First sector (63-16777215, default 63): 64
Last sector or +size or +sizeM or +sizeK (64-16777215, default 16777215): 208848
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

После выравнивания всех нужных разделов нужно отключить диск от виртуальной машины. При создании новой виртуальной машины необходимо выбрать данный диск для использования в виртуальной машине.

Выравнивание разделов для виртуальных машин Windows

Чтобы выровнять разделы для виртуальной машины Windows, нужна рабочая виртуальная машина Windows. В данной виртуальной машине можно использовать утилиту `diskpart` или `diskpar` (в зависимости от ОС) для выравнивания диска. Необходимо выполнить следующие действия:

1 Создать новый диск для виртуальной машины.

   На данном диске нужно создать выровненные разделы. Затем необходимо подключить диск к новой виртуальной машине и установить на диск гостевую ОС Windows.

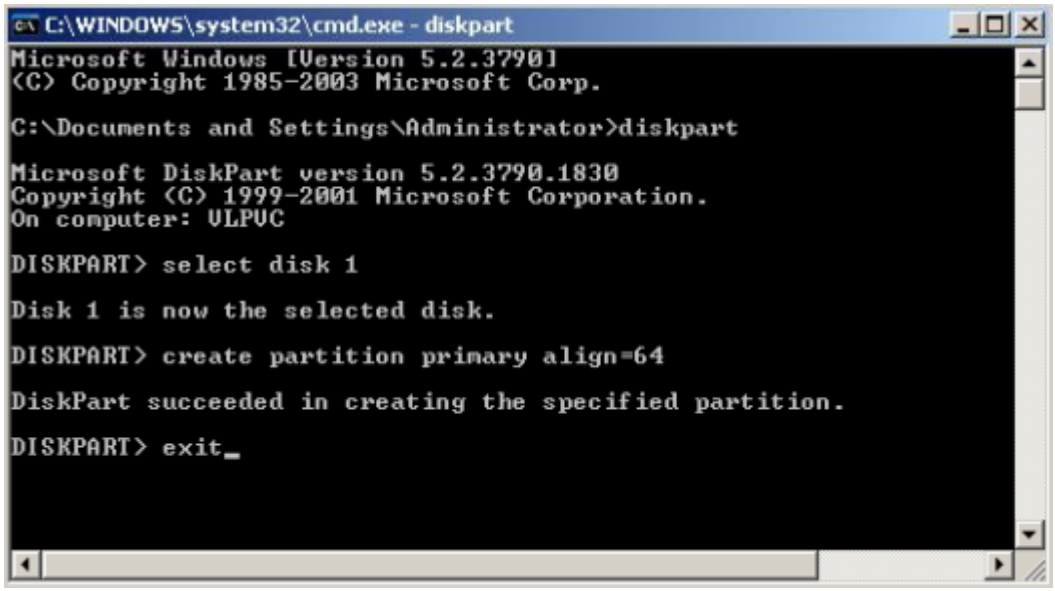
2 Открыть окно командной строки и запустить утилиту `diskpart` или `diskpar`.

3 Выбрать диск, который нужно выровнять.

4 Создать первичный раздел на диске и выровнять его.

5 Выйти из утилиты `diskpart` или `diskpar` и закрыть окно командной строки.

В примере ниже показано использование утилиты `diskpart` для выравнивания диска 1 путем его смещения на 64:



После выравнивания виртуального диска нужно отключить его от виртуальной машины. При создании новой виртуальной машины необходимо выбрать данный диск для использования в виртуальной машине.

Создание шаблона виртуальной машины с выровненными разделами

Чтобы облегчить процедуру создания виртуальных машин с выровненными системными разделами, можно создать шаблон выровненной виртуальной машины и использовать его для создания новых виртуальных машин.

Например, если выровнять диск, следуя инструкции из подраздела **Выравнивание разделов для виртуальных машин Windows** (стр. 167), потом создать виртуальную машину, которая использует данный диск, а затем установить операционную систему Windows Server 2008 в виртуальную машину, то получится чистая установка Windows Server 2008 на правильно выровненный диск. Далее можно создать шаблон данной виртуальной машины и использовать этот шаблон каждый раз для создания виртуальной машины с Windows Server 2008.

Удаление гостевых инструментов из виртуальных машин

Примечание: Запущенные виртуальные машины без установленных гостевых инструментов нельзя настроить с физического сервера.

Если гостевые инструменты несовместимы с каким-либо программным обеспечением виртуальной машины, вы можете удалить их. Действия по удалению гостевых инструментов отличаются в зависимости от гостевой операционной системы и описаны в разделах ниже.

Удаление гостевых инструментов из виртуальных машин Linux

Чтобы удалить гостевые инструменты из Linux ОС, войдите в виртуальную машину и выполните следующие действия:

1 Удалите пакеты:

- В системах на базе RPM (CentOS и другие):

```
# yum remove dkms-vzvirtio_balloon prl_nettool qemu-guest-agent-vz vz-guest-udev
```

- В системах на базе DEB (Debian и Ubuntu):

```
# apt-get remove vzvirtio-balloon-dkms prl-nettool qemu-guest-agent-vz vz-guest-udev
```

Если какой-либо пакет из перечисленных в примере выше не установлен в системе, команда выведет ошибку. В этом случае исключите данные пакеты из команды и выполните ее повторно.

2 Удалите файлы:

```
# rm -f /usr/bin/prl_backup /usr/share/qemu-ga/VERSION /usr/bin/install-tools \
/etc/udev/rules.d/90-guest_iso.rules /usr/local/bin/fstrim-static \
/etc/cron.weekly/fstrim
```

3 Перезагрузите правила `udev`:

```
# udevadm control --reload
```

После удаления гостевых инструментов перезапустите виртуальную машину.

Примечание: После удаления гостевых инструментов из виртуальной машины их состояние отображается как `possibly installed`.

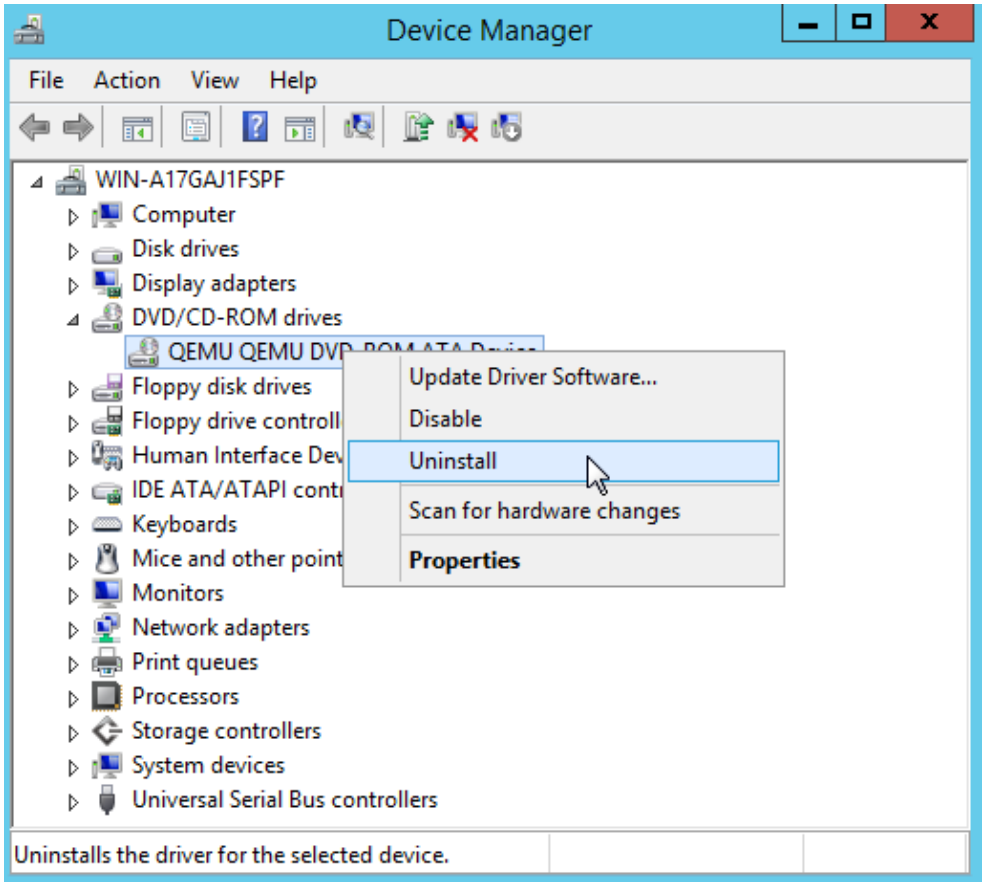
Удаление гостевых инструментов из виртуальных машин Windows

Чтобы удалить гостевые инструменты из Windows ОС, войдите в виртуальную машину и выполните следующие действия:

1 Удалите драйвера для виртуализированных устройств:

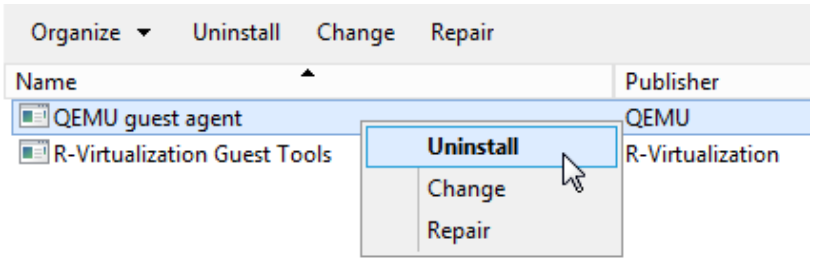
Важно: Не удаляйте драйвера для жесткого диска VirtIO/SCSI и сети NetKVM. Без первого драйвера виртуальная машина не загрузится; без второго – потеряет соединение с сетью.

1. В меню **Start** откройте **Control Panel** > **System and Security** > **System** > **Device Manager**.
2. Щелкните дважды по устройству, чтобы развернуть список установленных драйверов.
3. Щелкните правой кнопкой мыши по драйверу, который вы хотите удалить, и выберите **Uninstall** из ниспадающего меню.



2 Удалите QEMU guest agent и R-Virtualization Guest Tools:

1. В меню **Start** откройте **Control Panel** > **Programs** > **Programs and Features**.
2. Щелкните правой кнопкой мыши по **QEMU guest agent** и выберите **Uninstall** из ниспадающего меню.



3. Щелкните правой кнопкой мыши по **R-Virtualization Guest Tools** и выберите **Uninstall** из ниспадающего меню.

3 Остановите и удалите Guest Tools Monitor:

```
> sc stop VzGuestToolsMonitor
> sc delete VzGuestToolsMonitor
```

4 Отмените регистрацию Guest Tools Monitor в Event Log:

```
> reg delete HKLM\SYSTEM\CurrentControlSet\services\eventlog\Application\
VzGuestToolsMonitor
```

5 Удалите из реестра ключ автозапуска для RebootNotifier:

```
> reg delete HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v VzRebootNotifier
```

6 Удалите директорию `C:\Program Files\Qemu-ga\`.

Если файл `VzGuestToolsMonitor.exe` заблокирован, закройте все окна Event Viewer. Если файл остался заблокированным, перезапустите службу `eventlog`:

```
> sc stop eventlog
> sc start eventlog
```

После удаления гостевых инструментов перезапустите виртуальную машину.

Примечание: После удаления гостевых инструментов из виртуальной машины их состояние отображается как `possibly installed`.

Включение режима отладки для устаревших виртуальных машин

Режим отладки для устаревших виртуальных машин обеспечивает, что устаревшие виртуальные машины, которые не удалось конвертировать в формат ПК Р-Виртуализация новой версии в процессе миграции или восстановления из резервной копии, не удаляются с целевого сервера с новой версией ПК Р-Виртуализация после неудачной конвертации. Если режим отладки включен, то такие виртуальные машины будут находиться на целевом сервере в остановленном или запущенном состоянии с отключенной сетью, чтобы команда технической поддержки имела возможность изучить дамп памяти и найти причину проблемы.

Если при миграции или восстановлении из резервной копии устаревшей виртуальной машины конвертация не удается, можно включить режим отладки на целевом сервере с новой версией ПК Р-Виртуализация, предпринять еще одну попытку миграции или

восстановления, отправить отчет об ошибке и обратиться в службу технической поддержки.

Чтобы включить режим отладки для устаревших виртуальных машин на целевом сервере с новой версией ПК Р-Виртуализация, нужно выполнить следующие действия:

1 Остановите диспетчер:

```
# systemctl stop prl-disp.service
```

> **Примечание:** Пока диспетчер остановлен, управление виртуальными средами и ведение их статистики невозможно. Запущенные виртуальные среды не останавливаются.

2 В файле `/etc/vz/dispatcher.xml` измените `<LegacyVmUpgrade>0</LegacyVmUpgrade>` на `<LegacyVmUpgrade>1</LegacyVmUpgrade>`.

3 Запустите диспетчер:

```
# systemctl start prl-disp.service
```

Установка дополнительных пакетов ПК Р-Виртуализация

В ПК Р-Виртуализация установлены все необходимые пакеты. Также можно установить дополнительные пакеты ПК Р-Виртуализация из удаленных репозиториев при помощи команды `yum`.

> **Примечание:** Для получения дополнительной информации по использованию `yum` в ПК Р-Виртуализация см. **Обновление ПК Р-Виртуализация** (стр. 126) и man-страницу по `yum`.

Включение вложенной виртуализации в виртуальных машинах

> **Внимание:** Данная функция является экспериментальной и тестирована только на гостевых операционных системах Linux. Работа вложенных виртуальных машин может быть нестабильной.

ПК Р-Виртуализация поддерживает вложенную виртуализацию Intel VT-x в виртуальных машинах.

Для включения вложенной виртуализации необходимо выполнить следующие действия:

1 Остановите все запущенные или поставленные на паузу виртуальные машины на сервере.

2 Выгрузите модуль `kvm-intel` из ядра:

```
# rmmod kvm_intel
```

3 Добавьте строку `kvm-intel nested=y` в файл `dist.conf`:

```
# echo 'options kvm-intel nested=y' >> /etc/modprobe.d/dist.conf
```

4 Загрузите модуль `kvm-intel`:

```
# modprobe kvm_intel
```

5 Включите вложенную виртуализацию в нужной виртуальной машине:

```
# prlctl set MyVM --nested-virt on
```

Примечания:

1. Гостевая операционная система во вложенной виртуальной машине не сможет получить IP-адрес через DHCP, если параметрам `ipfilter`, `macfilter` и `preventpromisc` сетевого адаптера хостовой ВМ, соединенного через мост, задано значение `no`.

2. Невозможно изменить маску функций ЦП для вложенных виртуальных машин с помощью команды `prlsrvctl set --cpu-features-mask`.

Глава 10

Устранение неисправностей

В данной главе представлена информация о тех проблемах, которые могут возникнуть в процессе работы с ПК Р-Виртуализация, и способах их решения.

Общие положения

При обнаружении неисправностей системы рекомендуется учитывать общие положения, описанные. Следует внимательно их изучить прежде, чем пытаться устранить более конкретные проблемы.

- Необходимо всегда помнить свое местоположение в текущий момент в терминале. Можно периодически проверять его с помощью команд `pwd`, `hostname`, `ifconfig`, `cat /proc/vz/veinfo`. Одна и та же команда, выполненная в виртуальной среде и на физическом сервере, может иметь разные последствия. Также можно настроить переменную среды `PS1`, чтобы она показывала полный путь в `bash`, добавив следующие строки в `/root/.bash_profile`:

```
PS1="[\u@\h \w]$ "
export PS1
```

- При снижении производительности физического сервера можно использовать `vmstat`, `ps` (`ps axfw`), `dmesg`, `htop` (`vztop`), чтобы найти причину; не рекомендуется выполнять перезагрузку машины без выяснения причины. Если не удается вернуть нормальное функционирование, можно использовать последовательность клавиш Alt+SysRq для отображения памяти (`showMem`) и процессов (`showPc`).
- Не рекомендуется запускать бинарные файлы или скрипты, принадлежащие контейнеру, непосредственно с физического сервера. Например, категорически запрещается выполнять команду:

```
cd /vz/root/99/etc/init.d
./httpd status
```

  Любой скрипт в контейнере, может быть изменен по усмотрению его владельца: он может содержать троян, быть заменен на что-то, вроде `rm –rf`, и т.п. Для выполнения программ в контейнерах необходимо использовать команды `prlctl exec/prlctl enter`.
- Не следует использовать скрипты init на физическом сервере. Скрипт init может использовать `killall` для остановки службы, тем самым завершая все похожие процессы во всех контейнерах. Можно проверить `/var/run/Service.pid` и завершить соответствующий процесс выборочно.
- Необходимо быть способным обнаружить руткит в контейнере. Для его обнаружения рекомендуется использовать пакет `chkrootkit` (последнюю версию можно загрузить

по ссылке http://www.chkrootkit.org) или выполнить следующую команду для проверки, изменилась ли MD5-сумма для какого-либо RPM-пакета:

```
rpm -Va|grep "S.5"
```

Также можно запустить nmap, чтобы проверить, если ли открытые порты, которые должны быть закрыты, например:

```
# nmap -p 1-65535 192.168.0.1
Starting nmap V. 2.54BETA22 (www.insecure.org/nmap/ )
Interesting ports on  (192.168.0.1):
(The 65531 ports scanned but not shown below are in
  state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
Nmap run completed -- 1 IP address (1 host up) scanned
  in 169 seconds
```

Однако удаление руткита из контейнера и гарантия, что он 100% удален, может стать проблемой. Если нет уверенности, следует создать новый контейнер для данного пользователя и мигрировать на него сайты и почту пользователя.

- Необходимо проверить директорию `/var/log/` на физическом сервере, чтобы узнать, что происходит в системе. Ряд журналов ведется системой и ПК Р-Виртуализация (`boot.log`, `messages` и т.д.), но другие службы и программы могут также хранить там свои журналы, в зависимости от дистрибутива Linux и запущенных служб и приложений. Например, могут быть журналы, связанные с запуском почтового сервера (`maillog`), автоматическими задачами (файл `cron`) и др. Однако в первую очередь при обнаружении неисправностей следует изучить журнал `/var/log/messages`. Он содержит системные сообщения, появившиеся при запуске системы, а также другие сообщения о состоянии, возникшие в ходе работы системы. Ошибки операций ввода-вывода, сети и другие основные системные ошибки записываются в данный файл. Таким образом, следует начать с журнала `messages`, а затем уже переходить к другим файлам из директории `/var/log/`.
- Следует подписаться на системы отслеживания ошибок. Необходимо отслеживать новые общедоступные инструменты DoS или удаленные эксплойты для ПО и устанавливать их в контейнеры или на физические серверы.
- В `iptables` существует простое правило для использования цепочек, чтобы обеспечить безопасность как физического сервера, так и его контейнеров:
  - использовать INPUT, OUTPUT для фильтрации пакетов, получаемых/отправляемых физическим сервером,
  - использовать FORWARD to для фильтрации пакетов, предназначенных для контейнеров.

Устранение неисправностей ядра

Использование последовательностей клавиш ALT+SYSRQ

Можно нажать ALT+SYSRQ+H и проверить вывод, появившийся в консоли физического сервера, например:

```
SysRq: unRaw Boot Sync Unmount showPc showTasks showMem loglevel0-8 tErm kIll \
       killalL Calls Oops
```

Вывод показывает, какие последовательности клавиш ALT+SYSRQ можно использовать для выполнения различных команд. Заглавные буквы в названиях команд идентифицируют последовательность. Таким образом, при появлении неисправностей в машине рекомендуется сначала использовать следующие последовательности клавиш перед нажатием кнопки **Power**:

- ALT+SYSRQ+M для отображения информации о памяти
- ALT+SYSRQ+P для отображения состояний процессов
- ALT+SYSRQ+S для синхронизации дисков
- ALT+SYSRQ+U для размонтирования файловых систем
- ALT+SYSRQ+L для завершения всех процессов
- ALT+SYSRQ+U для попытки повторного размонтирования
- ALT+SYSRQ+B для перезагрузки

Если сервер не перезагрузится после этого, можно нажать кнопку **Power**.

Сохранение ошибок ядра (OOPS)

Для проверки сообщений ядра, которые должны быть переданы разработчикам ПК P-Виртуализация можно использовать следующую команду:

```
grep -E "Call Trace|Code" /var/log/messages*
```

Затем необходимо найти строки, относящиеся к ядру, в соответствующем журнале и определить, какое ядро было загружено, когда возникла ошибка ядра. Следует искать с конца строку `Linux`, вроде:

```
Sep 26 11:41:12 kernel: Linux version 2.6.18-8.1.1.el5.028stab043.1 \
(root@rhel5-32-build) (gcc version 4.1.1 20061011 (Red Hat 4.1.1-30)) \
#1 SMP Wed Aug 29 11:51:58 MSK 2007
```

Ошибка ядра обычно начинается с описания того, что произошло, и заканчивается строкой Code. Например:

```
Aug 25 08:27:46 boar BUG: unable to handle kernel NULL pointer dereference at \
virtual address 00000038
Aug 25 08:27:46 boar printing eip:
Aug 25 08:27:46 boar f0ce6507
Aug 25 08:27:46 boar *pde = 00003001
```

```
Aug 25 08:27:46 boar Oops: 0000 [#1]
Aug 25 08:27:46 boar SMP
Aug 25 08:27:46 boar last sysfs file:
Aug 25 08:27:46 boar Modules linked in: snapapi26(U) bridge(U) ip_vzredir(U) \
vzredir(U) vzcompat(U) vzrst(U) i
p_nat(U) vzcpt(U) ip_conntrack(U) nfnetlink(U) vzlinkdev(U) vzethdev(U) vzevent(U) \
vzlist(U) vznet(U) vzmo
n(U) xt_tcpudp(U) ip_vznetstat(U) vznetstat(U) iptable_mangle(U) iptable_filter(U) \
ip_tables(U) vztable(U) vzdquota(U) vzdev(U) autofs4(U) hidp(U) rfcomm(U) l2cap(U) \
bluetooth(U) sunrpc(U) ipv6(U) xt_length(U) ipt_ttl(U) xt_tcpmss(U) ipt_TCPMSS(U) \
xt_multiport(U) xt_limit(U) ipt_tos(U) ipt_REJECT(U) x_tables(U) video(U) sbs(U) \
i2c_ec(U) button(U) battery(U) asus_acpi(U) ac(U) lp(U) floppy(U) sg(U) pcspkr(U) \
i2c_piix4(U) e100(U) parport_pc(U) i2c_core(U) parport(U) cpqphp(U) eepro100(U) \
mii(U) serio_raw(U) ide_cd(U) cdrom(U) ahci(U) libata(U) dm_snapshot
(U) dm_zero(U) dm_mirror(U) dm_mod(U) megaraid(U) sym53c8xx(U) \
scsi_transport_spi(U) sd_mod(U) scsi_mod(U) ext3(U) jbd(U) ehci_hcd(U) ohci_hcd(U) \
uhci_hcd(U)
Aug 25 08:27:46 boar CPU: 1, VCPU: -1.1
Aug 25 08:27:46 boar EIP: 0060:[<f0ce6507>] Tainted: P VLI
Aug 25 08:27:46 boar EFLAGS: 00010246 (2.6.18-028stab043.1-ent #1)
Aug 25 08:27:46 boar EIP is at clone_endio+0x29/0xc6 [dm_mod]
Aug 25 08:27:46 boar eax: 00000010   ebx: 00000001   ecx: 00000000   edx: 00000000
Aug 25 08:27:46 boar esi: 00000000   edi: b6f52920   ebp: c1a8dbc0   esp: 0b483e38
Aug 25 08:27:46 boar ds: 007b   es: 007b   ss: 0068
Aug 25 08:27:46 boar Process swapper (pid: 0, veid: 0, ti=0b482000 task=05e3f2b0 \
task.ti=0b482000)
Aug 25 08:27:46 boar Stack: 0b52caa0 00000001 00000000 b6f52920 00000000f0ce64de \
00000000 02478825
Aug 25 08:27:46 boar 00000000 c18a8620 b6f52920 271e1a8c 024ca03800000000 00000000 \
00000000
Aug 25 08:27:46 boar 00000000 00000000 c18a3c00 00000202 c189e89400000006 00000000 \
05cb7200
Aug 25 08:27:46 boar Call Trace:
Aug 25 08:27:46 boar [<f0ce64de>] clone_endio+0x0/0xc6 [dm_mod]
Aug 25 08:27:46 boar [<02478825>] bio_endio+0x50/0x55
Aug 25 08:27:46 boar [<024ca038>] __end_that_request_first+0x185/0x47c
Aug 25 08:27:46 boar [<f0c711eb>] scsi_end_request+0x1a/0xa9 [scsi_mod]
Aug 25 08:27:46 boar [<02458f04>] mempool_free+0x5f/0x63
Aug 25 08:27:46 boar
Aug 25 08:27:46 boar [<f0c713c3>] scsi_io_completion+0x149/0x2f3 [scsi_mod]
Aug 25 08:27:46 boar [<f0c333b9>] sd_rw_intr+0x1f1/0x21b [sd_mod]
Aug 25 08:27:46 boar [<f0c6d3b9>] scsi_finish_command+0x73/0x77 [scsi_mod]
Aug 25 08:27:46 boar [<024cbfa2>] blk_done_softirq+0x4d/0x58
Aug 25 08:27:46 boar [<02426452>] __do_softirq+0x84/0x109
Aug 25 08:27:46 boar [<0242650d>] do_softirq+0x36/0x3a
Aug 25 08:27:46 boar [<024050b7>] do_IRQ+0xad/0xb6
Aug 25 08:27:46 boar [<024023fa>] default_idle+0x0/0x59
Aug 25 08:27:46 boar [<0240242b>] default_idle+0x31/0x59
Aug 25 08:27:46 boar [<024024b1>] cpu_idle+0x5e/0x74
Aug 25 08:27:46 boar =======================
Aug 25 08:27:46 boar Code: 5d c3 55 57 89 c7 56 89 ce 53 bb 01 00 00 00 83 ec 0c \
8b 68 3c 83 7f 20 00 8b 45 00 8b 00 89 44 24 04 8b 45 04 89 04 24 8b 40 04 <8b> \
40 28 89 44 24 08 0f 85 86 00 00 00 f6 47 10 01 75 0a 85 c9
Aug 25 08:27:46 boar EIP: [<f0ce6507>] clone_endio+0x29/0xc6 [dm_mod] \
SS:ESP0068:0b483e38
Aug 25 08:27:46 boar Kernel panic - not syncing: Fatal exception in interrupt
```

Можно сохранить ошибку ядра в файле, чтобы можно было предоставить ее при обращении в службу технической поддержки.

Обнаружение функции ядра, вызвавшее у процесса состояние D

Если обнаружилось большое число процессов в состоянии D (непрерывного сна) и невозможно определить причину, можно выполнить команду:

```
# objdump -Dr /boot/vmlinux-\`uname -r\` >/tmp/kernel.dump
```

И затем получить список процессов:

```
# ps axfwln
  F UID   PID  PPID PRI NI  VSZ  RSS  WCHAN STAT TTY TIME COMMAND
100   0 20418 20417  17  0 2588  684      - R    ?   0:00 ps axfwln
100   0     1     0   8  0 1388  524 145186 S    ?   0:00 init
040   0  8670     1   9  0 1448  960 145186 S    ?   0:00 syslogd -m 0
040   0  8713     1  10  0 1616 1140 11ea02 S    ?   0:00 crond
```

В колонке **WCHAN** следует посмотреть номер для нужного процесса. Далее необходимо открыть в редакторе `/tmp/kernel.dump`, найти данный номер в первой колонке и промотать обратно до первого имени функции, которое имеет следующий вид:

```
"c011e910 <sys_nanosleep>:"
```

Тогда можно понять, является ли процесс "живым" или заблокирован в данную функцию.

Проблемы управления контейнерами

В данный раздел включены рекомендации по решению определенных проблем контейнеров.

Отказ при запуске контейнера

Попытка запуска контейнера не удалась.

Решение 1

Если в консоли системы появилось сообщение: `IP address is already used`,— необходимо выполнить команду `cat /proc/vz/veinfo`. Для каждого запущенного контейнера отобразится информация, содержащая: UUID, класс, количество процессов в контейнере и IP-адреса. Вывод команды также покажет, что контейнер запущен без назначенных ему IP-адресов. Назначить ему IP-адреса можно с помощью команды:

```
# prlctl set <CT_name> --ipadd <IP_address>
```

где <*CT_name*> является именем контейнера, а <*IP_address*>—желаемым IP-адресом.

Решение 2

Возможно, контейнер неправильно настроен. Можно попробовать проверить конфигурацию контейнера и определить, какие параметры вызвали ошибку. Необходимо указать подходящие значения с помощью команды `prlctl set`.

Решение 3

Возможно, контейнер использовал всю свою дисковую квоту (дисковое пространство). Следует проверить квоту (см. **Управление дисковыми квотами** (стр. 74)) и настроить ее параметры при необходимости.

Решение 4

Можно запустить утилиту `prlctl console` для входа в контейнер и получения доступа к его консоли. Утилита предоставит вывод запуска/отключения для контейнера, который можно использовать для выявления проблемы. Например:

```
# prlctl console MyCT
```

где `MyCT` является именем контейнера.

Отказ в доступе к контейнеру по сети

Решение 1

Необходимо убедиться, что IP-адрес, назначенный контейнеру, не используется в сети. Проверить проблемный IP-адрес контейнера можно при помощи следующей команды:

```
# grep IP_ADDRESS /etc/vz/conf/<UUID>.conf
IP_ADDRESS="10.0.186.101"
```

IP-адреса других запущенных контейнеров можно проверить командой:

```
# cat /proc/vz/veinfo
```

Решение 2

Следует убедиться, что для контейнера настроена правильная маршрутизация. Контейнеры могут использовать в сети маршрутизатор по умолчанию или можно настроить, чтобы физический сервер выступал в роли маршрутизатора для контейнеров.

Отказ при входе в контейнер

Контейнер запущен, но в него не удается войти.

Решение 1

Осуществляется попытка соединения через SSH, но в доступе отказывается. Возможно, для пользователя `root` еще не установлен пароль, или данного пользователя не существует. В таком случае можно использовать команду `prlctl set --userpasswd`. Например, для контейнера `MyCT` можно выполнить следующую команду:

```
# prlctl set MyCT --userpasswd root:secret
```

Решение 2

Следует проверить параметры переадресации с помощью команды:

```
# cat /proc/sys/net/ipv4/conf/venet0/forwarding
```

Если в выводе отобразится значение `0`, то необходимо изменить его на `1` командой:

```
# echo 1 > /proc/sys/net/ipv4/conf/venet0/forwarding
```

Получение технической поддержки

Техническую поддержку можно получить с помощью рассылки, системы отслеживания ошибок и на форуме поддержки. Дополнительную информацию можно получить у вашего технического партнера.