Ошибки удаления LUN

1. Проблема

Иногда после удаления **LUN** из хранилищ - остаётся некоторое количество остаточных данных **multipath** и **lvm**, что выражается в ошибках в файле **/var/log/messages** хоста виртуализации вида:

```
Mar 10 03:23:02 zvirt-srv2 multipathd: 36d039ea0002e6292000001a761b0ae75: sdgr - rdac checker reports path is down: lun not connected
Mar 10 03:23:04 zvirt-srv2 multipathd: sded: Could not synchronize with kernel state
Mar 10 03:23:04 zvirt-srv2 multipathd: 36d039ea0002e629200000018461675d0b: sded - rdac checker reports path is down: lun not connected
Mar 10 03:23:04 zvirt-srv2 multipathd: sdcj: Could not synchronize with kernel state
Mar 10 03:23:04 zvirt-srv2 multipathd: 36d039ea0002e53d60000047d61657418: sdcj - rdac checker reports path is down: lun not connected
Mar 10 03:23:04 zvirt-srv2 multipathd: sddv: Could not synchronize with kernel state
Mar 10 03:23:04 zvirt-srv2 multipathd: 36d039ea0002e53d60000047e6165741f: sddv - rdac checker reports path is down: lun not connected
Mar 10 03:23:04 zvirt-srv2 multipathd: sdam: Could not synchronize with kernel state
Mar 10 03:23:04 zvirt-srv2 multipathd: 36d039ea0002e629200000018461675d0b: sdam - rdac checker reports path is down: lun not connected
Mar 10 03:23:04 zvirt-srv2 multipathd: sdw: Could not synchronize with kernel state
```

2. Решение

После удаления **LUN** из хранилищ, вывод всех хостов по одному в режим обслуживания и их перезагрузка.

Попытка автоматизации данного процесса (не рекомендовано к использованию):

```
#!/bin/bash
errors=0
for f in /dev/dm-*
do
    dd if=$f of=/dev/null count=1
    if [ $? -ne 0 ]
```

```
    then
      dmsetup remove $f
      errors=1
    fi
done
for f in /sys/block/sd*
do
  s=`cat $f/size`
  if [ "$s" == "0" ]
    then
      echo $f
      echo 1 >$f/device/delete
  fi
done
if [ $errors -eq 1 ]
then
  exec $0
fi

systemctl restart multipathd
exit 0
```

Увеличение swap (файл подкачки) на хосте

Процедура увеличения **swap** (файл подкачки) на хосте следующая:

1. Добавьте новый диск.
2. Разметьте его через `fdisk`.
3. Добавьте файл подкачки командой `mkswap /dev/имя_нового_диска`.
4. Включите файл подкачки командой `swapon /dev/имя_нового_диска`.
5. Добавьте соответсвующую запись в **/etc/fstab**. Например:

   ```
   /dev/имя_нового_диска swap swap defaults 0 0
   ```

6. Произведите монтирование командой `mount -a`.
7. Проверьте объем своп файла командой `free -h`.
8. Включите файл подкачки из `fstab` командой `swapon -a`.

Принудительное обновление размера LUN

1. Пояснение

В ряде случаев zVirt не может в автоматическом режиме обновить данные о размере LUN-а в случае подключения его к диску ВМ. Для упрощения этой операции можно воспользоваться подготовленным скриптом **refresh-lun.sh**, который обращается к API по адресу типа https://${FQDN}:443/ovirt-engine/api/disks/${LUNID}/refreshlun

2. Запуск скрипта

Скрипт необходимо разместить на ВМ HostedEngine и запустить с указанием параметра **LUNID**, например:

```
LUNID=24def43e-1f90-4c12-aaae-f496dc02b98d bash refresh-lun.sh
```

В ходе выполнения скрипта будет определено доменное имя zVirt, запрошен пароль пользователя **admin**, а также выполнена попытка обновления при помощи первого найденного хоста. В случае успешного выполнения операции вывод консоли будет примерно следующим:

```
LUNID=24def43e-1f90-4c12-aaae-f496dc02b98d bash refresh-lun.sh
Укажите пароль пользователя admin:
...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<action>
    <disk href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-f496dc02b98d"
id="24def43e-1f90-4c12-aaae-f496dc02b98d">
        <actions>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/reduce" rel="reduce"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/sparsify" rel="sparsify"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/export" rel="export"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/copy" rel="copy"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/move" rel="move"/>
```

```
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/refreshlun" rel="refreshlun"/>
        </actions>
        <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/permissions" rel="permissions"/>
        <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/statistics" rel="statistics"/>
    </disk>
    <host id="ef1e63e0-af88-4430-bdcf-ed37c336dba4"/>
    <status>complete</status>
</action>
```

Для определения идентификатора **LUN**, необходимо обратиться к меню **Хранилище** > **Диски**, перейти на вкладку **Прямой LUN** и среди списка отображенных дисков найти нужный. Значение LUNID будет в колонке **Код**.

В случае необходимости указания конкретного хоста, следует воспользоваться параметром **HOSTID**, например:

```
LUNID=24def43e-1f90-4c12-aaae-f496dc02b98d HOSTID=00521c3e-4598-4441-8bfa-
784de3b409dd bash refresh-lun.sh
Укажите пароль пользователя admin:
...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<action>
    <disk href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-f496dc02b98d"
id="24def43e-1f90-4c12-aaae-f496dc02b98d">
        <actions>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/reduce" rel="reduce"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/sparsify" rel="sparsify"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/export" rel="export"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/copy" rel="copy"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/move" rel="move"/>
            <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/refreshlun" rel="refreshlun"/>
        </actions>
        <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/permissions" rel="permissions"/>
        <link href="/ovirt-engine/api/disks/24def43e-1f90-4c12-aaae-
f496dc02b98d/statistics" rel="statistics"/>
    </disk>
    <host id="00521c3e-4598-4441-8bfa-784de3b409dd"/>
```

```
    <status>complete</status>
</action>
```

3. Возможные проблемы

1. В случае указания неверного **LUNID** будет выведено сообщение примерного вида:

```
LUNID=24def43e-1f90-4c12-aaae-f496dc02b98d  bash refresh-lun.sh
Укажите пароль пользователя admin:
...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<action>
    <fault>
        <detail>[Cannot sync Virtual Disk. The specified disk does not
exist.]</detail>
        <reason>Operation Failed</reason>
    </fault>
    <host id="ef1e63e0-af88-4430-bdcf-ed37c336dba4"/>
    <status>failed</status>
</action>
```

2. В случае указания неверного **HOSTID** будет выведено сообщение примерного вида:

```
LUNID=2d175b60-99d8-4877-9817-f3d9362bd07b HOSTID=00521c3e-4598-4441-8bfa-
784de3b409db bash refresh-lun.sh
Укажите пароль пользователя admin:
...
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<action>
    <fault>
        <detail>[Cannot sync Virtual Disk. Invalid Host Id.]</detail>
        <reason>Operation Failed</reason>
    </fault>
    <host id="00521c3e-4598-4441-8bfa-784de3b409db"/>
    <status>failed</status>
</action>
```

4. Код скрипта **refresh-lun.sh**

```
#!/usr/bin/bash

if [ -z ${LUNID} ]; then
    echo "Запустите скрипт с указанием LUN-а"
    echo "LUNID=24def43e-1f90-4c12-aaae-f496dc02b98d bash ./refresh-lun.sh"
    exit
```

```
fi

# detect zvirt fqdn
FQDN=$(hostname -f)

# ask admin password
echo "Укажите пароль пользователя admin:"
read PASSWD

if [ -z ${HOSTID} ]; then
    # get first' host id
    HOSTID=$(curl -s \
    --cacert '/etc/pki/ovirt-engine/apache-ca.pem' \
    --request GET \
    --header 'Version: 4' \
    --header 'Accept: application/xml' \
    --user 'admin@internal:'${PASSWD} \
    https://${FQDN}/ovirt-engine/api/hosts | grep host | grep id | head -n1 |
awk -F\" '{print $4; exit}')
fi

# send refresh command to lun
curl \
--cacert '/etc/pki/ovirt-engine/apache-ca.pem' \
--request POST \
--header 'Accept: application/xml' \
--header "Content-Type: application/xml" \
--header "Host: ${FQDN}" \
--user 'admin@internal:'${PASSWD} \
--data "<action><host id='${HOSTID}'/></action>" \
https://${FQDN}/ovirt-engine/api/disks/${LUNID}/refreshlun
```

Загрузка образов дисков в хранилище с помощью утилиты "upload disk"

Загрузка образов с помощью **upload disk** является альтернативным способом загрузки образов дисков в **Домен Хранения** и может применяться, когда загрузка классическим способом невозможна. Загрузка данным способом возможно как с менеджера управления, так и с хостов **zVirt**.

1. Подготовка:

Для загрузки образов с помощью **upload disk**, необходимо переустановить **python3-ovirt-engine-sdk4**.

Для этого можно воспользоваться командой:

```
dnf reinstall -y python3-ovirt-engine-sdk4
```

Так же необходимо создать конфигурационный файл **ovirt.conf** по пути **~/.config/**. Файл описывается в **ini-формате** и должен содержать блок конфигураций со следующими параметрами:

1. В квадратных скобках необходимо задать название конфигурации. Данное название в дальнейшем будет использовать как ключ утилиты **upload disk**.
2. **engine_url** — адрес менеджера управления.
3. **username** — имя пользователя, под которым произойдёт авторизация в **zVirt** во время выполнения **upload disk**. Имя пользователя указывается полностью с пространством имён, в случае использования внутренних учётных записей, пространство имён будет **@internal**. В случае интеграции с Keycloak используйте пространство имен **@internalsso**
4. **password** — пароль пользователя.
5. **secure** — загрузка в Домен хранения по зашифрованному каналу.
6. **cafile** — корневой сертификат **zVirt**.

   В этом параметре должен быть указан путь до сертификата удостоверяющего центра. В случае с менеджером управления путь будет следующий: **/etc/pki/ovirt-engine/ca.pem**, если же запуск `upload disk` будет осуществляться с хоста виртуализации, путь будет следующий: **/etc/pki/vdsm/certs/cacert.pem**.

Пример конфигурационного файла:

```
[conf-name]
engine_url = https://zvirt.example.ru
username = admin@zvirt@internalsso
password = admin
secure = yes
cafile = /etc/pki/ovirt-engine/ca.pem
```

2. Загрузка образа

поддерживаются форматы образов **qcow2, iso, img**.

Для загрузки образа, необходимо скопировать образ в любой каталог на хосте/менеджере управления. Из каталога с образом диска необходимо выполнить команду:

```
python3 /usr/share/doc/python3-ovirt-engine-sdk4/examples/upload_disk.py -c
conf-name --sd-name StorageDomainName YourImage.iso
```

С помощью ключа **-c** указывается конфигурация, созданная в файле **ovirt.conf**. В ключе **--sd-name** указывается название **Домена хранения**, в который будет загружен образ.

Если утилита запускается не из каталога, в котором содержится образ — нужно будет указать относительный или абсолютный путь до файла.

3. Устранения ошибок:

При появлении ошибки

```
virtsdk4.Error: Fault reason is "Operation Failed". Fault detail is "[Cannot add
Virtual Disk. Disk configuration (COW Preallocated backup-None) is incompatible
with the storage domain type.]". HTTP response code is 409.
```

Попробуйте добавить флаг **--enable-backup**.

Пример команды с флагом:

```
python3 /usr/share/doc/python3-ovirt-engine-sdk4/examples/upload_disk.py -c
conf-name --sd-name StorageDomainName YourImage.iso --enable-backup
```

Если конвертация всё-равно завершается с ошибкой, то необходимо конвертировать диск в формат **raw** с помощью команды:

```
qemu-img convert -O raw YourImageIn.img YourImageOut.img
```

После указания формата **raw** нужно указать исходный образ, затем указать имя нового конвертированного образа.

Технический справочник. Основные компоненты zVirt

1. Менеджер управления

Менеджер управления обеспечивает централизованное управление средой виртуализации. Доступ к Менеджеру управления можно получить через несколько различных интерфейсов, при этом каждый по-разному обеспечивает доступ к среде виртуализации.

Рисунок 1. Архитектура Менеджера управления

Менеджер управления предоставляет графические интерфейсы и API-интерфейс. Каждый интерфейс подключается к Менеджеру управления, т. е. к приложению, доставляемому встроенным экземпляром Wildfly. Помимо Wildfly, существует ряд других компонентов, которые поддерживают Менеджер управления.

2. Среда исполнения zVirt

К среде zVirt подключены один или несколько хостов. **Хост** — это сервер, который предоставляет физическое оборудование, используемое виртуальными машинами.

На хосте работает оптимизированная среда исполнения zVirt Node, предназначенная для создания хостов виртуализации.

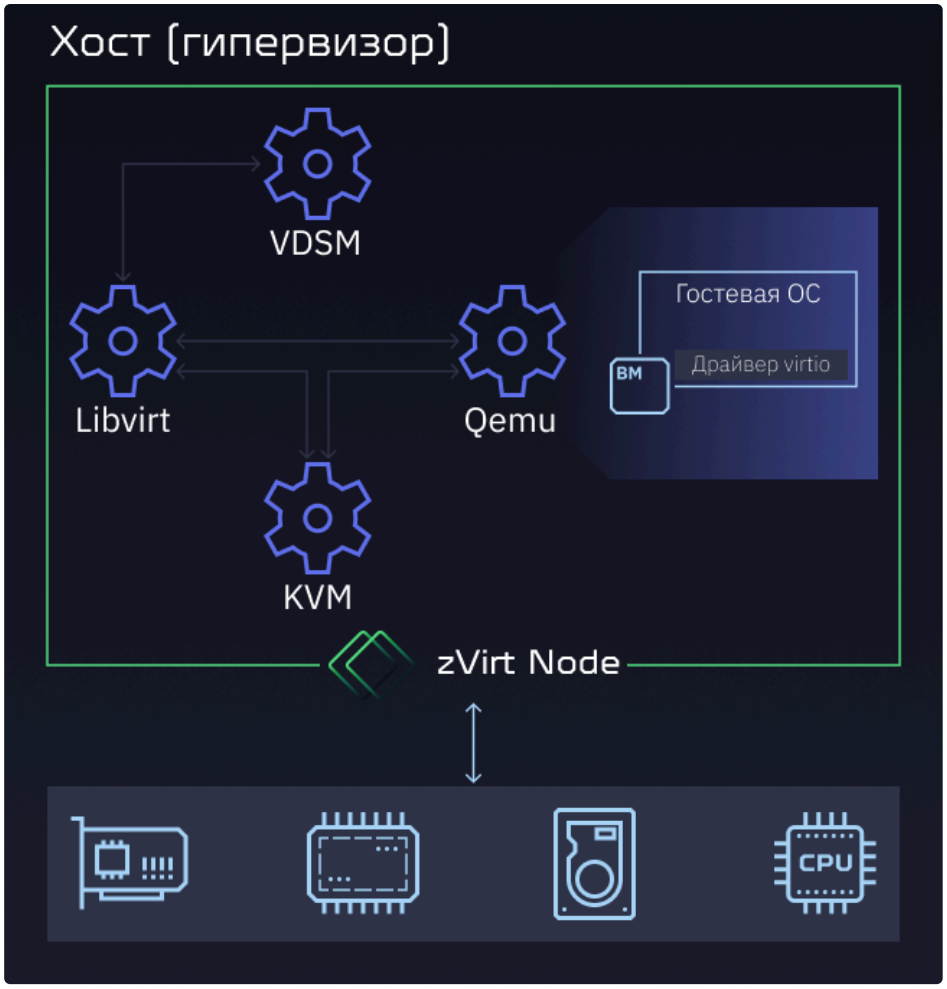

Рисунок 2. Архитектура хоста

Модуль Kernel-based Virtual Machine (KVM)

Модуль Kernel-based Virtual Machine (KVM) — это загружаемый модуль ядра, который обеспечивает полную виртуализацию с помощью расширений аппаратных средств Intel VT или AMD-V. Хотя сам модуль KVM работает в пространстве ядра, выполняющиеся на нем гостевые процессы работают как отдельные процессы QEMU в пространстве пользователя. С помощью KVM хост делает свое физическое оборудование доступным для виртуальных машин.

Эмулятор QEMU

QEMU — это мультиплатформенный эмулятор для полной эмуляции системы. QEMU эмулирует систему полностью, например, ПК, в том числе один или несколько процессоров и периферийные устройства. Эмулятор можно использовать для запуска различных операционных систем или для отладки системного кода. QEMU, работая в связке с KVM и процессором с соответствующими расширениями виртуализации, обеспечивает полную аппаратную виртуализацию.

Агенты хоста Менеджера управления, VDSM

В zVirt, **VDSM** запускает действия над виртуальными машинами и хранилищем, а также поддерживает связь между хостами. VDSM отслеживает ресурсы хоста: память, хранилище и сеть. Кроме того, VDSM занимается диспетчеризацией задач при создании

виртуальных машин, сборе статистики и журналов. Экземпляр VDSM запускается на каждом хосте и получает команды управления от Менеджера управления через перенастраиваемый порт `54321`.

VDSM-REG

VDSM с помощью **VDSM-REG** регистрирует каждый хост в Менеджере управления. VDSM-REG предоставляет информацию о самом себе и своем хосте через порт `80` или порт `443`.

libvirt

Libvirt поддерживает управление виртуальными машинами и ассоциированными с ними виртуальными устройствами. Когда Менеджер управления запускает команды жизненного цикла виртуальной машины (запуск, остановка, перезагрузка), VDSM вызывает libvirt на соответствующих хостах для их выполнения.

Менеджер пула хранения (Storage Pool Manager, SPM)

Менеджер пула хранения (Storage Pool Manager, SPM) — это роль, назначаемая одному хосту в центре данных. Хост SPM обладает исключительным правом вносить все изменения в метаданные структуры домена хранения в конкретном центре данных. Сюда относятся создание, удаление и управление виртуальными дисками, снимками и шаблонами, а также выделение ресурсов хранилища для динамически расширяемых блочных устройств в SAN хранилищах. Роль SPM можно передать любому хосту в центре данных. В результате у всех хостов в центре данных должен быть доступ ко всем доменам хранения, заданным в центре данных.

Менеджер управления обеспечивает постоянную доступность SPM. При возникновении ошибок подключения хранилища Менеджер управления назначает роль SPM другому хосту.

Гостевая операционная система

Гостевые операционные системы можно установить на виртуальные машины в среде zVirt без модификаций. Гостевая операционная система и любые приложения поверх нее не знают, что находятся в среде виртуализации, и работают как обычно.

zVirt предоставляет расширенные драйверы устройств для более быстрого и эффективного доступа к виртуализированным устройствам. Кроме того, можно установить гостевой агент zVirt, который передает более подробную информацию о гостевых машинах на консоль управления.

3. Компоненты, поддерживающие Менеджер управления

3.1. WildFly

WildFly — это сервер приложений Java, предоставляющий фреймворк для эффективной разработки и доставки кросс-платформенных приложений Java. Менеджер управления доставляется через WildFly.

Версию WildFly, поставляемую вместе с Менеджером управления, нельзя использовать для обслуживания других приложений. Она была настроена специально для обслуживания Менеджера управления. Если сервер WildFly, входящий в состав Менеджера управления, использовать в дополнительных целях, то это отрицательно сказывается на его способности обслуживать среду zVirt.

3.2. Сбор отчетов и данных истории

Менеджер управления включает в себя хранилище (**DWH**, **Data Warehouse**), которое собирает данные мониторинга о хостах, виртуальных машинах и хранилищах, а также в нем предусмотрено несколько предварительно заготовленных отчетов. Заказчики могут анализировать свои среды и создавать отчеты с помощью любых инструментов запросов, поддерживающих SQL.

В процессе установки Менеджера управления создаются две базы данных:

- Менеджер управления использует базу данных **engine** в качестве основного хранилища данных. В этой базе данных хранится информация о среде виртуализации, например, о ее состоянии, конфигурации и производительности.
- В базе данных **ovirt_engine_history** содержится информация о конфигурации и статистические показатели, которые консолидируются с течением времени из операционной базы данных **engine**. Данные о конфигурации в базе данных **engine** проверяются ежеминутно, а изменения реплицируются в базу данных **ovirt_engine_history**. Отслеживание изменений в базе данных позволяет получить информацию об объектах в базе данных, а также проанализировать и повысить производительность среды zVirt и устранить неполадки.

  Дополнительные сведения о создании отчетов на основе базы данных **ovirt_engine_history** см. в разделе База данных истории в Руководстве по хранилищу zVirt.

Репликация данных в базу данных **ovirt_engine_history** выполняется сервисом ovirt-engine-dwhd.

Эти базы данных создаются на экземпляре Postgres, который выбирается во время установки.

3.3. Службы каталогов

Службы каталогов поддерживают централизованное сетевое хранилище пользовательской и организационной информации, включая такие типы информации как настройки приложений, профили пользователей, данные групп, политики и контроль доступа. Менеджер управления поддерживает:

- 389ds
- 389ds RFC-2307 Schema
- Active Directory
- IBM Security Directory Server
- IBM Security Directory Server RFC-2307 Schema
- IPA
- Novell eDirectory RFC-2307 Schema
- OpenLDAP RFC-2307 Schema
- OpenLDAP Standard Schema
- Oracle Unified Directory RFC-2307 Schema
- RFC-2307 Schema (Generic)
- RHDS
- RHDS RFC-2307 Schema
- iPlanet

Существует также локальный, внутренний домен, предназначенный только для целей администрирования. В этом внутреннем домене есть только один пользователь: администратор (admin).

4. Хранилище

В zVirt используется централизованная СХД для виртуальных дисков, шаблонов, снимков и файлов ISO. Хранилище логически сгруппировано в пулы хранения, которые состоят из доменов хранения. Домен хранения объединяет в себе емкость хранилища и метаданные, которые описывают внутреннюю структуру хранилища (См. раздел Типы доменов хранения):

- **Домен данных** — это единственный домен, обязательный для каждого центра данных. У каждого центра данных свой относящийся только к нему домен хранения данных.
- **Домены экспорта** и **ISO** — не строго обязательные домены.

Домены хранения — общие ресурсы, которые должны быть доступны для всех хостов центра данных.

5. Сеть (Network)

Сетевая архитектура zVirt обеспечивает связь между разными элементами среды zVirt. Сетевая архитектура не только поддерживает сетевое подключение, но и позволяет разделять сети.

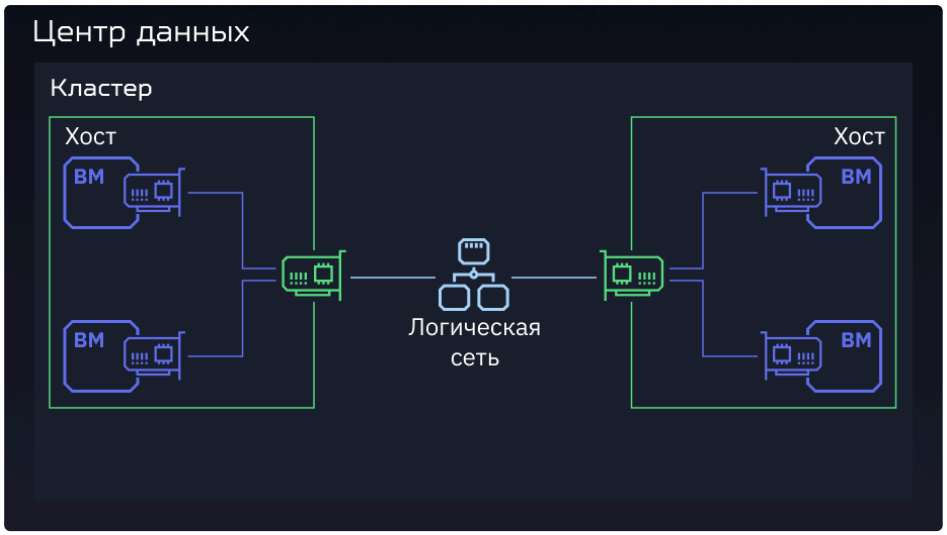

Рисунок 3. Сетевая архитектура

В zVirt сетевая архитектура состоит из нескольких уровней. Необходимо создать и настроить базовую физическую сетевую инфраструктуру, чтобы обеспечить связь между аппаратными и логическими компонентами среды zVirt.

5.1. Уровень сетевой инфраструктуры

В основе сетевой архитектуры zVirt лежат некоторые стандартные аппаратные и программные устройства:

- **Сетевые карты** (**NIC**) — это физические устройства сетевого интерфейса, через которые хост подключается к сети.
- **Виртуальные сетевые карты** (**vNIC**) — это логические сетевые карты, которые работают на физических сетевых картах хоста и обеспечивают сетевое подключение виртуальных машин.
- Несколько сетевых карт объединяют в единый bond-интерфейс.
- **Мосты** (**bridges**) — это метод пересылки пакетов в сетях с коммутацией пакетов. Это основа логических сетей виртуальных машин.

5.2. Логические сети

Логические сети позволяют разделять сетевой трафик в зависимости от требований среды. Типы логических сетей:

- логические сети, передающие сетевой трафик виртуальных машин
- логические сети, не передающие сетевой трафик виртуальных машин
- необязательные логические сети
- обязательные сети

Все логические сети могут быть либо обязательными, либо необязательными.

Логическая сеть, передающая сетевой трафик виртуальных машин, реализуется на уровне хоста как программный мост. По умолчанию при установке Менеджера управления задается одна логическая сеть: сеть управления **ovirtmgmt**.

Другие логические сети, которые может добавить администратор: выделенная логическая сеть хранилища и выделенная логическая сеть отображения. В логических сетях, не передающих трафик виртуальных машин, нет ассоциированного устройства моста на хостах. Они ассоциированы с сетевыми интерфейсами хоста напрямую.

zVirt отделяет сетевой трафик, связанный с управлением, от сетевого трафика, связанного с миграцией, что позволяет использовать выделенную сеть (без маршрутизации) для миграции на лету и гарантирует, что сеть управления (ovirtmgmt) не потеряет связь с гипервизорами во время миграции.

5.3. Описание логических сетей на разных уровнях

Логические сети по-разному воздействуют на каждый уровень среды виртуализации.

Уровень центра данных

Логические сети задаются на уровне центра данных. В каждом центре данных по умолчанию есть сеть управления **ovirtmgmt**. Желательно, но необязательно добавить другие логические сети. Назначить **Сеть ВМ** (**VM Network**) и задать пользовательское значение MTU можно на уровне центра данных. Логическую сеть, заданную центру данных, также необходимо добавить в кластеры, которые используют эту логическую сеть.

Уровень кластера

Логические сети предоставляются из центра данных, и их необходимо добавить в кластеры, которые будут их использовать. По умолчанию каждый кластер подключен к сети управления. При желании можно добавить в кластер логические сети, которые были заданы для родительского центра данных кластера. Когда необходимая логическая сеть добавлена в кластер, она должна быть добавлена для каждого хоста в кластере. Дополнительные логические сети можно добавить к хостам по мере необходимости.

Уровень хоста

Логические сети виртуальных машин реализуются для каждого хоста в кластере в виде программного моста, ассоциированного с определенным сетевым интерфейсом. В логических сетях невиртуальных машин нет ассоциированных мостов; они ассоциированы с сетевыми интерфейсами хостов напрямую. На каждом хосте есть сеть управления, реализованная как мост с помощью одного из сетевых устройств, включенного в среду zVirt. Дополнительные необходимые логические сети, добавленные в кластер, следует ассоциировать с сетевыми интерфейсами на каждом хосте, чтобы они заработали с кластером.

Уровень виртуальной машины

Логические сети могут быть доступны для виртуальных машин так же, как сеть может быть доступна для физической машины. Виртуальная машина может подключить свою виртуальную сетевую карту к любой логической сети виртуальной машины, которая была реализована на хосте, на котором она работает. После этого виртуальная машина сможет подключиться к любым другим устройствам или приемникам, доступным в логической сети, к которой она подключена.

Пример 1. Сеть управления (Management Network)

Логическая сеть управления под названием **ovirtmgmt** создается автоматически при установке Менеджера управления. Сеть **ovirtmgmt** выделена для трафика управления между Менеджером управления и хостами. Если другие мосты специального назначения не настроены, то **ovirtmgmt** будет мостом по умолчанию для всего трафика.

6. Центры данных

Центр данных — это высший уровень абстракции в zVirt. В центре данных содержится три вида информации:

Хранилище

Включает типы хранилищ, домены хранилищ и информацию о подключении доменов хранилищ. Хранилище задано для центра данных и доступно всем кластерам в нем. Все хосты в кластерах центра данных имеют доступ к одним и тем же доменам хранения.

Логические сети

Сюда относятся такие данные, как сетевые адреса, теги VLAN и поддержка STP. Можно задать логические сети для центра данных и применить их к кластерам.

Кластеры

Кластеры — это группы хостов с совместимыми ядрами процессоров, будь то процессоры AMD или Intel. Кластеры представляют собой домены миграции; виртуальные машины можно переместить на лету на любой хост в кластере, но не на другие кластеры.

В одном центре данных может быть несколько кластеров, а каждый кластер может содержать несколько хостов.

7. Уровни совместимости центров данных и кластеров

У центров данных и кластеров zVirt есть версия совместимости.

Версия совместимости центра данных обозначает версию zVirt, с которой должен быть совместим конкретный центр данных. Все кластеры в таком центре данных должны поддерживать требуемый уровень совместимости.

Версия совместимости кластера описывает возможности zVirt, поддерживаемые всеми хостами в кластере. Совместимость кластера устанавливается в соответствии с версией операционной системы наименее подходящего хоста в кластере.

8. Безопасность

8.1. Шифрование паролей

Для повышения уровня информационной безопасности системные пароли в конфигурационных файлах шифруются с помощью алгоритма AES (Advanced Encryption Standard).

Шифрование паролей применяется к следующим файлам:

- /etc/ovirt-engine/engine.conf.d/11-setup-sso.conf
- /etc/ovirt-engine/engine.conf.d/10-setup-database.conf
- /etc/ovirt-engine/engine.conf.d/10-setup-dwh-database.conf
- /etc/ovirt-engine/engine.conf.d/10-setup-pki.conf
- /etc/ovirt-engine-dwh/ovirt-engine-dwhd.conf.d/10-setup-grafana-database.conf.
- /etc/ovirt-engine/aaa/***.local.properties
- /etc/ovirt-engine/engine.conf.d/10-setup-cinderlib-database.conf

8.2. Цифровая подпись пакетов

Для повышения доверия к распространяемому программному обеспечению реализовано подписание файлов с помощью электронной цифровой подписи. В метаданных теперь указывается производитель «Orionsoft».

Для проверки цифровой подписи используйте команду `rpm -K <package>`.

Пример 2. Проверка цифровой подписи пакетов

```
rpm -K cockpit-299.fl-1.19278.wip.zvirt.el8.x86_64.rpm ①
cockpit-299.fl-1.19278.wip.zvirt.el8.x86_64.rpm: digests signatures OK ②

rpm -K cockpit-310.2-1.fc40.src.rpm ③
cockpit-310.2-1.fc40.src.rpm: digests SIGNATURES NOT OK ④
```

① - проверка цифровой подписи пакета, скачанного с официального репозитория Orionsoft.

② - проверка пройдена успешно.

③ - проверка цифровой подписи пакета, скачанного со стороннего репозитория.

④ - проверка не пройдена.

Также можно проверить производителя в метаданных пакетов с помощью `rpm -qi <package>`.

```
rpm -qi cockpit-299.fl-1.19278.wip.zvirt.el8.x86_64.rpm ①

Name        : cockpit
Version     : 299.fl
...
Vendor      : "Orionsoft" ②
...

rpm -qi cockpit-310.2-1.fc40.src.rpm ③

warning: cockpit-310.2-1.fc40.src.rpm: Header V4 RSA/SHA256 Signature, key ID
a15b79cc: NOKEY ④
Name        : cockpit
Version     : 310.2
...
Vendor      : Fedora Project ⑤
...
```

① - проверка метаданных пакета, скачанного с официального репозитория Orionsoft.

② - вендором данного пакета является **Orionsoft**.

③ - проверка метаданных пакета, скачанного со стороннего репозитория.

④ - присутствует предупреждение об отсутствии ключа.

⑤ - вендором данного пакета является **Fedora Project**.

Технический справочник. Драйверы оборудования и устройства

1. Виртуализированное оборудование

zVirt представляет три разных типа системных устройств виртуализированным гостевым машинам. Все эти аппаратные устройства отображаются для виртуализированной гостевой машины как физически подключенные аппаратные устройства, но драйверы устройств работают по-разному.

Эмулируемые устройства

Эмулируемые устройства, иногда называемые виртуальными, существуют целиком и полностью в программном виде. Драйверы эмулируемых устройств представляют собой слой трансляции между операционной системой, работающей на хосте (которая управляет исходным устройством), и операционными системами, работающими на гостевых машинах. Инструкции уровня устройства, поступающие на эмулируемое устройство и с него, перехватываются и транслируются гипервизором. Любое устройство того же типа, что и эмулируемое и распознаваемое ядром Linux, можно использовать в качестве базового исходного устройства для эмулируемых драйверов.

Паравиртуализированные устройства

Для паравиртуализированных устройств требуется установка драйверов устройств в гостевой операционной системе, предоставляющих ей интерфейс для связи с гипервизором на хосте. Этот интерфейс используется, например, для задач с традиционно интенсивным вводом-выводом, выполняемых вне среды виртуализации. Такой способ снижения присущих виртуализации накладных расходов призван приблизить производительность гостевой операционной системы к производительности, ожидаемой при работе непосредственно на физическом оборудовании.

Совместно используемые физические устройства

Некоторые аппаратные платформы позволяют виртуализированным гостевым машинам напрямую обращаться к различным аппаратным устройствам и компонентам. Этот процесс в виртуализации называется сквозным доступом или назначением устройств. Сквозной доступ позволяет устройствам отображаться и вести себя так, как если бы они были физически подключены к гостевой операционной системе.

2. Устойчивые адреса устройств в zVirt

Назначенные виртуальному оборудованию PCI-адреса сохраняются в базе данных **ovirt-engine**.

QEMU назначает PCI-адреса при создании виртуальной машины, а **libvirt** передает их в **VDSM**. **VDSM** передает их обратно Менеджеру управления, где они и хранятся в базе данных **ovirt-engine**.

Когда виртуальная машина запускается, Менеджер управления извлекает из базы данных адрес устройства и отправляет его в **VDSM**. **VDSM** передает их в **libvirt**, который запускает виртуальную машину, используя PCI-адреса устройств, назначенные при первом запуске виртуальной машины.

Когда устройство удаляется из виртуальной машины, все ссылки на него, включая устойчивый PCI-адрес, также удаляются. Если для замены удаленного устройства добавляется другое устройство, **QEMU** назначает ему PCI-адрес, который вряд ли будет совпадать с адресом удаленного устройства.

3. Центральный процессор (ЦП)

Каждый хост в кластере имеет несколько виртуальных ЦП (vCPU), которые, в свою очередь, предоставляются гостевым машинам, работающим на хостах. Все виртуальные ЦП, предоставляемые хостами в кластере, относятся к типу, выбранному при первоначальном создании кластера с помощью Менеджера управления. Смешивание типов виртуальных ЦП в кластере невозможно.

Характеристики каждого доступного типа виртуального ЦП основаны на одноименных физических ЦП. Для гостевой операционной системы виртуальный ЦП неотличим от физического.

4. Системные устройства

Системные устройства имеют ключевое значение для работы гостевой машины и не могут быть удалены. Каждое системное устройство, подключенное к гостевое машине, также занимает доступный PCI-слот. Системные устройства по умолчанию:

- Мост хоста
- Мост ISA и мост USB (мосты USB и ISA - это одно и то же устройство)
- Графическая карта, использующая драйвер VGA или qxl
- Устройство балунинга памяти

5. Сетевые устройства

zVirt может предоставлять гостевым машинам сетевые карты трех типов. Тип сетевой карты, предоставляемой гостевой машине, выбирается при создании гостевой машины, но его можно изменить в Менеджере управления.

- Сетевая карта **e1000** предоставляет гостевым машинам виртуализированное устройство Intel PRO/1000 (e1000).
- Сетевая карта **virtio** предоставляет гостевым машинам паравиртуализированное устройство.
- Сетевая карта **rtl8139** предоставляет гостевым машинам виртуализированное устройство **Realtek Semiconductor Corp RTL8139**.

Одна гостевая машина может иметь несколько сетевых карт. Каждая добавляемая сетевая карта занимает свободный PCI-слот гостевой машины.

6. Графические устройства

Для подключения к эмулируемым графическим устройствам можно использовать графические протоколы SPICE или VNC.

На Портале администрирования можно выбрать **Тип видео** (**Video Type**):

- **QXL**: Эмулирует паравиртуализированную видеокарту, которая лучше всего работает с гостевыми драйверами QXL.
- **VGA**: Эмулирует фиктивную карту VGA с VESA-расширениями Bochs
- **BOCHS**: Эмулирует фиктивную карту VGA без устаревшей эмуляции для гостевых машин с UEFI. Этот эмулятор видеокарты дисплея используется по умолчанию для серверов UEFI.

Выбор типа видео влияет на доступность соответствующих графических протоколов. В zVirt 4.X сопоставление **видео - протокол** следующее:

- **QXL** - SPICE и VNC
- **VGA** - VNC
- **BOCHS** - VNC

7. Устройства хранения

Устройства хранения и пулы хранения могут использовать драйверы блочных устройств для подключения устройств хранения к виртуализированным гостевым машинам. Обратите внимание, что драйверы устройств хранения - это не устройства хранения. Драйверы используются для подключения базового устройства хранения, файла или тома пула

хранения к виртуализированной гостевой машине. Базовым устройством хранения может быть любой поддерживаемый тип устройства хранения, файла или тома пула хранения.

- Драйвер **IDE** предоставляет эмулируемое блочное устройство гостевым машинам. Эмулируемый драйвер **IDE** позволяет подключить до 4 виртуализированных жестких дисков IDE или виртуализированных CD-ROM приводов **IDE** (в любой комбинации) к каждой виртуализированной гостевой машине. Эмулируемый драйвер **IDE** также используется для предоставления виртуализированных дисков DVD-ROM.
- Драйвер **VirtIO** предоставляет паравиртуализированное блочное устройство гостевым машинам. Паравиртуализированный блочный драйвер - это драйвер для всех устройств хранения, поддерживаемых гипервизором и подключенных к виртуализированной гостевой машине (за исключением дисководов гибких дисков, которые необходимо эмулировать).

8. Звуковые устройства

Доступны два эмулируемых звуковых устройства:

- Так, **ac97** эмулирует звуковую карту, совместимую с **Intel 82801AA AC97 Audio**.
- А **es1370** эмулирует звуковую карту **ENSONIQ AudioPCI ES1370**.

9. Последовательный драйвер

Паравиртуализированный последовательный драйвер (**virtio-serial**) - это драйвер байтового и символьного потоков. Паравиртуализированный последовательный драйвер обеспечивает простой интерфейс между пользовательским пространством хоста и пользовательским пространством гостевой машины в случаях, когда сетевое подключение недоступно или непригодно для использования.

10. Драйвер балунинга

Драйвер балунинга позволяет гостевым машинам сообщать гипервизору, сколько памяти им требуется. Драйвер балунинга позволяет хосту эффективно выделять и освобождать память на гостевой машине, а также делает возможным выделение свободной памяти другим гостевым машинам и процессам.

Гостевые машины, использующие драйвер балунинга, могут помечать разделы ОЗУ гостевой машины как неиспользуемые (уменьшение выделяемой области). Гипервизор может освобождать память и использовать ее для других процессов хоста или других гостевых

машин на этом хосте. Когда гостевой машине снова потребуется освобожденная память, гипервизор может снова выделить ОЗУ гостевой машине (увеличение выделяемой области).

Репликация и аварийное восстановление виртуальных машин

1. Общие сведения

Репликация и аварийное восстановление (Disaster Recovery, DR) — это сервис, предназначенный для обеспечения непрерывности работы бизнес-приложений и их восстановления в случае аварийных ситуаций. Этот сервис помогает пользователям планировать, тестировать и запускать процессы восстановления виртуальных машин (ВМ) между основным и резервным центрами обработки данных (ЦОД). Таким образом, он позволяет минимизировать риски простоя и потери данных, обеспечивая высокую доступность и надежность критически важных для бизнеса приложений.

Основные причины использования сервисов репликации и аварийного восстановления:

Обеспечение непрерывности бизнеса

В случае сбоев в работе основной инфраструктуры (например, отказ оборудования, аварии) сервис репликации позволяет быстро восстановить работу виртуальных машин на резервных площадках. Это особенно важно для критически важных приложений и сервисов, которые не могут позволить себе длительные простои.

Защита данных

Репликация данных между основным и резервным ЦОД обеспечивает сохранность информации в случае потери или повреждения данных на основной площадке.

Снижение времени восстановления (RTO)

Быстрое восстановление виртуальных машин после аварии или сбоя критически важно для поддержания непрерывности работы бизнеса. Сервис репликации позволяет сократить время восстановления (RTO) до минимально возможного, что особенно важно для высоконагруженных систем.

Улучшение отказоустойчивости

Репликация и аварийное восстановление обеспечивают высокую степень отказоустойчивости, так как позволяют мгновенно переключаться на резервные копии виртуальных машин в случае сбоя основной системы. Это снижает риск простоев и увеличивает общую надежность инфраструктуры.

zVirt версии 4.3 предоставляет два решения для репликации и DR:

- **Агентская репликация (репликация на уровне гипервизора)**. При репликации виртуальных машин на уровне гипервизора создается вспомогательная виртуальная машина с установленным агентом, которая имеет доступ к дисковой подсистеме гипервизора и может читать и передавать данные снимков виртуальных машин. Вспомогательные ВМ (агенты) должны быть развернуты на источнике (основной ЦОД) и приёмнике репликации (резервный ЦОД).
- **СХД-репликация**. При репликации на уровне системы хранения данных (СХД) не требуются вспомогательные ВМ. Репликация осуществляется на уровне СХД между основной и резервной площадками и прозрачна для платформы виртуализации.

2. Глоссарий

Термин	Определение
Общие понятия	
ОЦОД	Основная площадка, из которой выполняется репликация в РЦОД
РЦОД	Резервная площадка, на которую выполняется репликация. При запуске планов восстановления на ней запускаются реплицированные виртуальные машины.
Репликация	Механизм синхронизации данных, связанных с виртуальными машинами, между ОЦОД и РЦОД.
Группа репликации	Набор виртуальных машин, объединенных в группу, для которой настроены параметры репликации.
План восстановления	Шаблон, в котором описана последовательность восстановления виртуальных машин из реплик в РЦОД.
Реплика виртуальной машины	Копия виртуальной машины на системе хранения, находящейся на одной из площадок (основной или резервной).
Агентская репликация	
Агент-отправитель	Виртуальная машина(ы), которая развертывается в ОЦОД (на уровне всей инсталляции/центра данных/кластера/хоста и выполняет репликацию ВМ в РЦОД)
Агент-приемник	Виртуальная машина(ы), которая развертывается в РЦОД и работает с реплицированными данными ВМ в РЦОД
Группа восстановления	Группа ВМ, восстанавливаемых в РЦОД. Состав группы формируется при создании плана восстановления
Контроллер репликации	ВМ, развернутая в РЦОД, которая управляет процессами репликации

Термин	Определение
Снимок	Функция, которая позволяет администратору создавать точки восстановления операционной системы, приложений и данных ВМ на определенный момент времени
Общий диск	Общий (shared) диск, который может быть подключен к нескольким ВМ
Ранг	Приоритет запуска ВМ. ВМ с рангом «0» запускаются первыми, затем запускаются ВМ с рангом «1» и так далее
СХД-репликация	
LUN (Logical unit number)	Адрес диска или дискового устройства в сетях хранения.
Домен хранения	Набор образов, имеющих общий интерфейс хранения. Домен хранения содержит полные образы шаблонов и виртуальных машин (включая снимки) или файлы ISO. Домены хранения могут базироваться на блочном и файловом хранилищах. Домен данных поддерживает все типы хранилищ, поддерживаемые zVirt.
Репликационная пара	Пара LUN, реплицируемых между СХД.
Аварийное восстановление	Процесс запуска виртуальных машин в РЦОД в случае аварии в ОЦОД с переключением или разрывом репликации между СХД.
Плановое перемещение	Процесс переноса виртуальных машин в РЦОД в плановом режиме после выключения виртуальных машин в ОЦОД и изменения направления репликации.
Обратное перемещение	Процесс переноса виртуальных машин с РЦОД в ОЦОД, выполняемый после планового перемещения или аварийного восстановления.
Очистка ресурсов в ОЦОД	Процесс приведения ресурсов в ОЦОД (виртуальные машины, центры данных, домены хранения, кластеры) в состояние, позволяющее выполнить обратное перемещение из РЦОД в ОЦОД.
Инициализация плана восстановления	Процесс перевода плана восстановления в статус готовности к выполнению.

3. Агентская репликация

3.1. Архитектура и основные компоненты

Решение состоит из следующих компонентов:

- **Контроллер репликации zVirt**. Отвечает за конфигурацию, запуск репликации и восстановление, управляет агентами. Контроллер репликации размещается как виртуальная машина в РЦОД.

- **Агент-отправитель**. Отвечает за создание снимков состояния реплицируемых ВМ и их передачу в РЦОД. Агент-отправитель размещается как виртуальная машина в ОЦОД.
- **Агент-приемник**. Отвечает за запись данных реплицируемых ВМ в РЦОД. Агент-приемник размещается как виртуальная машина в РЦОД.
- **Две независимые площадки zVirt**. Каждая из площадок управляется своим Менеджером управления. каждая площадка подключена к собственной СХД.

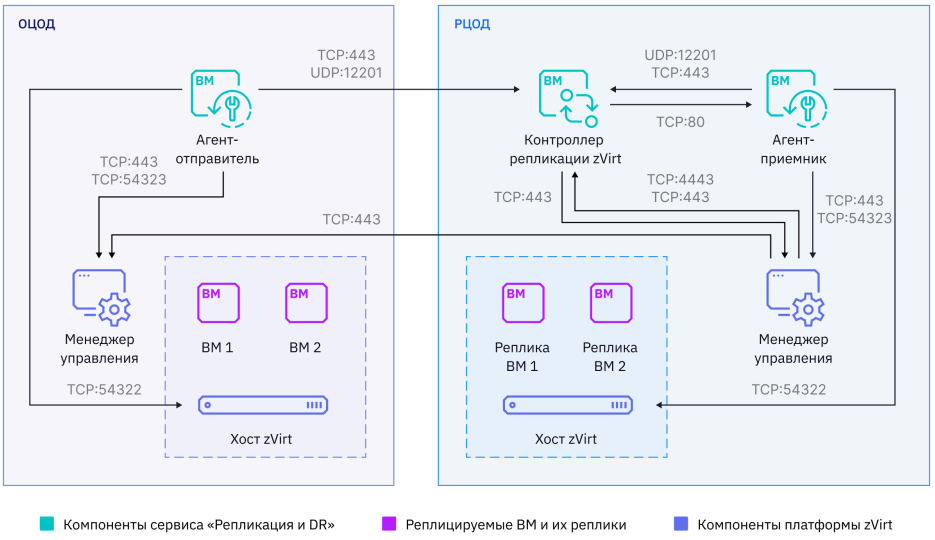

Рисунок 1. Архитектура сервиса агентской репликации и DR.

На схеме:

- ВМ 1 и ВМ 2 — защищаемые ВМ, расположенные на платформе zVirt в ОЦОД.
- Реплика ВМ 1 и ВМ 2 — ВМ, создаваемые контроллером репликации в РЦОД на платформе zVirt для защиты ВМ 1 и ВМ 2.
- zVirt Engine (Менеджер управления) и узел виртуализации zVirt — независимые платформы виртуализации zVirt, размещаемые в ОЦОД и РЦОД. Каждый Менеджер управляет гипервизорами только в рамках своего ЦОД. Платформы виртуализации в ОЦОД и РЦОД не имеют общего хранилища и могут не иметь растянутой L2-сети.

На схеме не отображены порты служебных сервисов. Информацию об используемых портах см. в требованиях к компонентам руководства по использованию сервиса Агентской репликации и DR.

3.2. Принцип работы

Агент-отправитель, размещаемый в ОЦОД, при запуске сервисов инициирует соединение с контроллером и получает конфигурацию ОЦОД. Для получения данных дисков реплицируемых ВМ агент-отправитель обращается к гипервизору, на котором работает реплицируемая ВМ, и забирает данные дисков.

Сетевое взаимодействие реплицируемой ВМ с агентами или контроллером репликации не требуется.

Возможно масштабирование количества агентов-отправителей для увеличения пропускной способности репликации ВМ.

Контроллер репликации zVirt располагается в РЦОД. Взаимодействие агентов-отправителей с контроллером может осуществляться как с трансляцией адресов (используется параметр «Публичный IP-адрес контроллера репликации» у контроллера), так и напрямую с использованием маршрутизации. Контроллер обращается к API zVirt Engine для управления репликами ВМ, а также передает реплицируемые данные через агент-приемник. Контроллер выполняет функции хранения данных о настроенных группах репликации, планах восстановления, о состоянии репликации ВМ, запускает и управляет репликациями и планами восстановления.

Горизонтальное масштабирование функций контроллера между несколькими ВМ не поддерживается.

На стадии репликации ВМ с ОЦОД на РЦОД происходит поблочное копирование дисков. На стадии переключения ВМ подготавливается ВМ в РЦОД (добавление сетевых интерфейсов, назначение вычислительных ресурсов согласно плану восстановления) и запуск ВМ с cloud-init конфигурацией.

Требуется предварительная настройка сервиса гостевой ОС ВМ.

4. СХД-репликация

4.1. Архитектура и основные компоненты

Решение состоит из следующих компонентов:

- Поддерживаемые модели СХД.
- Две независимые площадки zVirt. Каждая из площадок управляется своим Менеджером управления. каждая площадка подключена к собственной СХД.

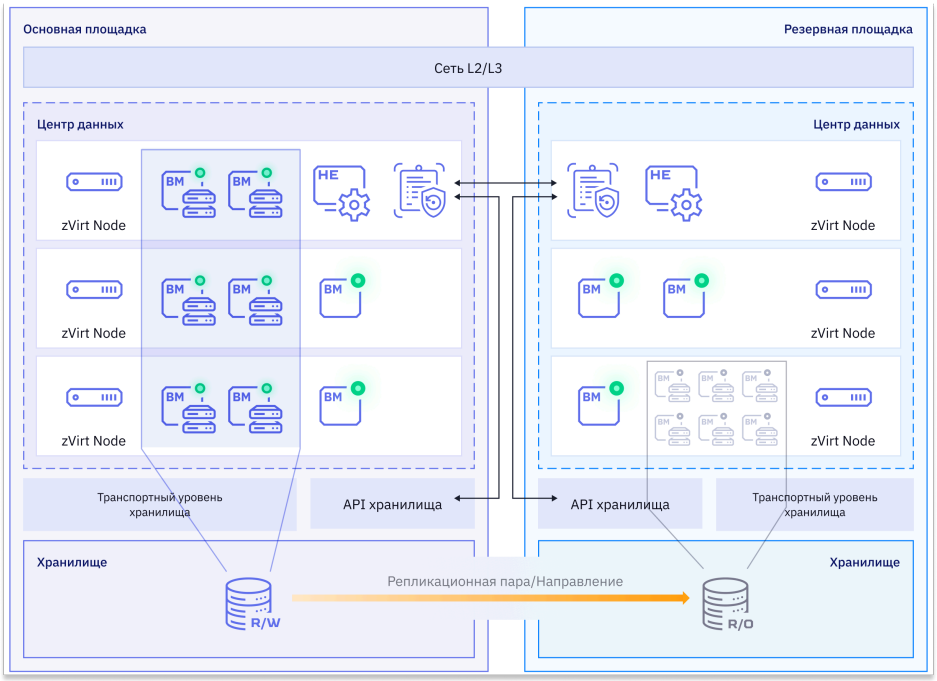

Рисунок 2. Архитектура сервиса СХД-репликации и DR.

4.2. Принцип работы

СХД-репликация - это механизм создания "клона" тома на удаленной СХД. При активной репликации серверы (клиенты) могут писать только в том на локальной СХД, для активации тома на удаленной площадке требуется провести некоторые операции с репликацией и томом.

Репликация не подразумевает одновременный доступ к тому на локальной и удаленной площадке, а также автоматического переключения тома на работу на удаленной площадке.

На уровне СХД возможны следующие типы репликации:

- **Синхронная репликация** - при записи данных сервер получит подтверждение только после того, как удаленная СХД подтвердила факт записи.
- **Асинхронная репликация** - при записи данных сервер получит подтверждение в момент, как только локальная СХД выполнила запись. Запись на удаленную СХД может быть отложена или выполняться сразу.

Асинхронная репликация может выполняться:

- **Постоянно** - локальная СХД при получении блока на запись начинает запись локально и отправляет запрос на удаленную СХД.
- **По расписанию** - локальная СХД при получении блока на запись начинает запись локально и выполняет запись на удаленную площадку только в настроенное время. Обычно периодичность выполнения операции составляет от 5 минут до 24 часов. Для сохранения разницы данных между локальной и удаленной копией тома используется механизм снимков: локальная СХД создает снимок тома, синхронизирует его с

удалённой площадкой, через назначенный период времени создает еще один снимок, синхронизирует его и т.д.

При этом, в зависимости от настроек СХД/тома, возможен переход синхронной репликации к асинхронной, а также от постоянной асинхронной к репликации по расписанию. Обычно это требуется при потере связи между СХД или при отказе удаленной СХД. такой подход позволяет продолжить запись в локальную СХД при авариях или технически работах на удаленной СХД.

В общем виде настройка репликации на СХД выглядит следующим образом:

1. Создание пары СХД (локальная СХД - удаленная СХД). Выполняется на одной из двух СХД, происходит аутентификация/авторизация между СХД, обмен служебными данными.
2. Создание тома на локальной СХД.
3. Создание репликации с локальной на удаленную СХД. В некоторых требуется выполнить активацию функции репликации.
4. Создание разрешающих правил доступа хостов к тому на локальной СХД.

После настройки репликации хосты имеют доступ в режиме RW (чтение/запись) к тому на локальной СХД, репликация находится в активном состоянии. Том на удаленной площадке или находится в неактивном состоянии, или презентован только в режиме RO (только чтение).

СХД поддерживают следующие варианты переключений:

- **Штатное переключение** - локальная СХД работает и доступна. Может использоваться для проведения плановых работ на локальной СХД или для тестирования плана восстановления.
- **Аварийное переключение** - локальная СХД недоступна для удаленной СХД, но при этом может еще работать.

Штатное переключение состоит из следующих шагов:

1. Остановка записи на том (выполняется на уровне платформы zVirt).
2. Отключение доступа к тому (демонтирование или перевод тома в режим RO).
3. Выполнение репликации данных, которые не успели отреплицироваться (в случае асинхронной репликации).
4. Остановка/разворот репликации.
5. Перевод тома на удаленной площадке в режим RW.
6. Монтирование тома хостам на удаленной площадке (при необходимости в зависимости от СХД).

Аварийное переключение состоит из следующих шагов:

1. Остановка/разворот репликации.
2. Перевод тома на удаленной площадке в режим RW.
3. Монтирование тома хостам на удаленной площадке (при необходимости в зависимости от СХД)

После восстановления работы СХД на локальной площадке требуется выполнить дополнительные действия для активации репликации с удаленной на локальную площадку. Набор действий зависит от реализации СХД.
