Общие сведения о REST API

1. Типы

API использует концепцию **типов** для описания различных видов принимаемых и возвращаемых объектов.

Существует три соответствующих вида типов:

- Примитивные типы - Описывают простые виды объектов, такие как string (строки) или integer (целые числа).
- Перечисляемые типы - Описывают списки допустимых значений, таких как `VmStatus` или `DiskFormat`.
- Структурированные типы - Описывают структурированные объекты с несколькими атрибутами и ссылками, например `Vm` или `Disk`.

2. Примитивные типы данных

В этом разделе описываются примитивные типы данных, поддерживаемые API.

2.1. Тип данных string

Конечная последовательность символов Unicode.

2.2. Тип данных boolean

Представляет собой понятия "ложь (false)" и "истина (true)", используемые в математической логике.

Допустимыми значениями являются строки `false` и `true`.

Регистр игнорируется Менеджером, поэтому, например, `False` и `FALSE` также являются допустимыми значениями. Однако сервер всегда будет возвращать значения в нижнем регистре.

Для обратной совместимости со старыми версиями Менеджера также принимаются значения `0` и `1`. Значение `0` `имеет тот же смысл, что и` `false`, а `1` - тот же смысл, что и `true`. Старайтесь не использовать эти значения, поскольку в будущем их поддержка может быть прекращена.

2.3. Тип данных integer

Представляет собой математическое понятие целого числа.

Допустимыми значениями являются конечные последовательности десятичных цифр.

В настоящее время Менеджер реализует этот тип с помощью знакового 32-битного целого числа, поэтому минимальное значение равно -2^{31} (-2147483648), а максимальное - $2^{31}-1$ (2147483647).

Однако в системе есть некоторые атрибуты, для которых диапазон значений, возможных при использовании 32 бит, недостаточен. В этих исключительных случаях Менеджер использует 64-битные целые числа, в частности, для следующих атрибутов:

- Disk.actual_size
- Disk.provisioned_size
- GlusterClient.bytes_read
- GlusterClient.bytes_written
- Host.max_scheduling_memory
- Host.memory
- HostNic.speed
- LogicalUnit.size
- MemoryPolicy.guaranteed
- NumaNode.memory
- QuotaStorageLimit.limit
- StorageDomain.available
- StorageDomain.used
- StorageDomain.committed
- VmBase.memory

Для этих исключений минимальное значение равно -2^{63} (-9223372036854775808), а максимальное - $2^{63}-1$ (9223372036854775807).

В будущем тип **integer** будет реализован с использованием целых чисел с неограниченной точностью, поэтому указанные выше ограничения и исключения со временем исчезнут.

2.4. Тип данных decimal

Представляет собой математическое понятие действительного числа.

В настоящее время Менеджер реализует этот тип, используя 32-битные числа IEEE 754 с плавающей запятой одинарной точности.

Для некоторых атрибутов такой точности недостаточно. В этих исключительных случаях Менеджер использует 64-битные числа с плавающей запятой двойной точности, в частности, для следующих атрибутов:

- QuotaStorageLimit.usage
- QuotaStorageLimit.memory_limit
- QuotaStorageLimit.memory_usage

В будущем десятичный тип будет реализован с использованием десятичных чисел неограниченной точности, поэтому описанные выше ограничения и исключения со временем исчезнут.

2.5. Тип данных date

Представляет собой дату и время.

Формат, возвращаемый Менеджером, соответствует формату, описанному в спецификации XML Schema при запросе XML. Например, для получения XML-представления виртуальной машины можно отправить такой запрос:

```
GET /ovirt-engine/api/vms/123
Accept: application/xml
```

Тело ответа будет содержать следующий XML-документ:

```
<vm id="123" href="/ovirt-engine/api/vms/123">
  ...
  <creation_time>2016-09-08T09:53:35.138+03:00</creation_time>
  ...
</vm>
```

При запросе JSON-представления Менеджер использует другой формат: целое число, содержащее количество миллисекунд с 1 января 1970 года, называемое также временем эпохи. Например, если послать такой запрос для получения JSON-представления виртуальной машины:

```
GET /ovirt-engine/api/vms/123
Accept: application/json
```

Тело ответа будет содержать следующий JSON-документ:

```
{
  "id": "123",
  "href="/ovirt-engine/api/vms/123",
  ...
  "creation_time": 1472564909990,
  ...
}
```

В обоих случаях даты, возвращаемые Менеджером, используют часовой пояс, настроенный на сервере, где он запущен, в приведенных примерах это UTC+3.

3. Идентифицируемые типы

Многие из типов, используемых API, представляют собой идентифицируемые объекты, объекты, которые имеют уникальный идентификатор и существуют независимо от других объектов. Типы, используемые для описания этих объектов, содержат следующий набор общих атрибутов:

Атрибут	Тип	Описание
id	String	Каждый объект в инфраструктуре виртуализации содержит `id`, который действует как уникальный идентификатор.
href	String	Каноническое расположение объекта как абсолютный путь.
name	String	Вводимое пользователем удобочитаемое имя объекта. Имя `name` уникально для всех объектов одного типа.
description	String	Пользовательское описание объекта в свободной форме, удобочитаемое для человека.

В настоящее время для большинства типов объектов атрибут `id` фактически представляет собой случайно сгенерированный UUID, но это деталь реализации, и пользователям не следует полагаться на это, так как это может измениться в будущем. Вместо этого пользователи должны предполагать, что эти идентификаторы являются просто строками.

4. Объекты

Объекты — это отдельные экземпляры типов, поддерживаемых API. Например, виртуальная машина с идентификатором `123` является объектом типа **Vm**.

5. Коллекции

Коллекция — это набор объектов одного типа.

6. Представления

Состояние объектов должно быть представлено при их передаче между клиентом и сервером. API поддерживает XML и JSON в качестве представления состояния объектов как для ввода, так и для вывода.

6.1. XML-представление

XML-представление объекта состоит из элемента XML, соответствующего типу объекта, атрибутов XML для атрибутов `id` и `href` и вложенных элементов XML для остальных атрибутов. Например, представление XML для виртуальной машины выглядит следующим образом:

```xml
<vm id="123" href="/ovirt-engine/api/vms/123">
  <name>myvm</name>
  <description>My VM</description>
  <memory>1073741824</memory>
  ...
</vm>
```

XML-представление набора объектов состоит из элемента XML, названного по типу объектов во множественном числе. Он содержит представления объектов коллекции. Например, представление XML для набора виртуальных машин выглядит следующим образом:

```xml
<vms>
  <vm id="123" href="/ovirt-engine/api/vms/123">
    <name>yourvm</name>
    <description>Your VM</description>
    <memory>1073741824</memory>
    ...
  </vm>
  <vm id="456" href="/ovirt-engine/api/vms/456">
    <name>myname</name>
    <description>My description</description>
    <memory>2147483648</memory>
    ...
  </vm>
  ...
</vms>
```

В теле ответа XML-представление используется по умолчанию, но для явного указания можно добавить в заголовок запроса параметр `Accept: application/xml`.

Для передачи тела запроса в формате XML, добавьте в заголовок запроса параметр `Content-type: application/xml`.

> В XML-представлении объектов атрибуты `id` и `href` являются единственными, которые представлены как атрибуты XML, остальные представлены как вложенные элементы XML.

6.2. JSON-представление

JSON-представление объекта состоит из документа JSON, содержащего пару `имя:значение` для каждого атрибута (включая `id` и `href`). Например, JSON-представление виртуальной машины выглядит следующим образом:

JSON

```
{
  "id": "123",
  "href": "/ovirt-engine/api/vms/123",
  "name": "myvm",
  "description": "My VM",
  "memory": 1073741824,
  ...
}
```

Представление коллекции объектов в формате JSON состоит из документа JSON, содержащего пару `имя:значение` (названную по типу объектов в единственном числе), которая, в свою очередь, содержит массив с представлениями объектов коллекции. Например, представление JSON для набора виртуальных машин выглядит следующим образом:

JSON

```
{
  "vm": [
    {
      "id": "123",
      "href": "/ovirt-engine/api/vms/123",
      "name": "myvm",
      "description": "My VM",
      "memory": 1073741824,
      ...
    },
    {
      "id": "456",
      "href": "/ovirt-engine/api/vms/456",
      "name": "yourvm",
      "description": "Your VM",
      "memory": 2147483648,
```

```
      ...
    },
  ]
}
```

Для получения тела ответа в формате JSON добавьте в заголовок запроса параметр `Accept: application/json`

Для передачи тела запроса в формате JSON, добавьте в заголовок запроса параметр `Content-type: application/json`.

7. Сервисы

Сервисы — это части сервера, отвечающие за извлечение, добавление обновлений, удаление и выполнение действий над объектами, поддерживаемыми API.

Существует два соответствующих вида сервисов:

- **Сервисы, управляющие коллекцией объектов** - эти сервисы отвечают за перечисление существующих объектов и добавление новых объектов. Например, сервис **Vms** отвечает за управление набором виртуальных машин, доступных в системе.
- **Сервисы, управляющие конкретным объектом** - эти сервисы отвечают за получение, обновление, удаление и выполнение действий в определенных объектах. Например, сервис **Vm** отвечает за управление конкретной виртуальной машиной.

Каждый сервис доступен по определенному пути внутри сервера. Например, сервис, управляющий набором виртуальных машин, доступных в системе, доступен по пути `/vms`, а сервис, управляющий виртуальной машиной **123**, доступен по пути `/vms/123`.

Все виды сервисов имеют набор **методов**, представляющих операции, которые они могут выполнять. Сервисы, управляющие коллекциями объектов, обычно имеют методы `list` и `add`. Сервисы, управляющие определенными объектами, обычно имеют методы `get`, `update` и `remove`. Кроме того, сервисы могут также иметь методы **action**, представляющие менее распространенные операции. Например, сервис **Vm** имеет метод **start**, который используется для запуска виртуальной машины.

Для более обычных методов существует прямое сопоставление между именем метода и именем метода HTTP:

Имя метода	HTTP-метод
`add`	POST
`get`	GET
`list`	GET

Имя метода	HTTP-метод
`update`	PUT
`remove`	DELETE

Путь, используемый в HTTP-запросе — это путь сервиса с префиксом `/ovirt-engine/api`.

Например, запрос для получения списка виртуальных машин с помощью метода `list` должен быть таким (используется HTTP-метод `GET` и путь `/vms`):

```
GET /ovirt-engine/api/vms
```

Для методов **action** HTTP-метод всегда `POST`, а имя метода добавляется в качестве суффикса к пути. Например, запрос на запуск виртуальной машины **123** должен выглядеть так (используется HTTP-метод `POST` и путь `/vms/123/start`):

```
POST /ovirt-engine/api/vms/123/start
```

Каждый метод имеет набор параметров.

Параметры делятся на две категории:

- **Основной параметр** - основной параметр соответствует объекту или коллекции, которые извлекаются, добавляются или обновляются. Это относится только к методам `add`, `get`, `list` и `update` и для каждого метода будет ровно один такой главный параметр.
- **Вторичные параметры** - остальные параметры.

Например, операция добавления виртуальной машины имеет три параметра: `vm`, `clone` и `clone_permissions`. Основным параметром является `vm`, так как он описывает добавляемый объект. Параметры `clone` и `clone_permissions` являются вторичными параметрами.

Основной параметр, используемый для ввода, должен быть включен в тело HTTP-запроса. Например, при добавлении виртуальной машины в тело запроса должен быть включен параметр `vm` типа `Vm`. Таким образом, полный запрос на добавление виртуальной машины, включая все детали HTTP, должен выглядеть следующим образом:

```
POST /ovirt-engine/api/vms HTTP/1.1
Host: myengine.example.com
Authorization: Bearer fqbR1ftzh8wBCviLxJcYuV5oSDI=
Content-Type: application/xml
Accept: application/xml
```

```
<vm>
  <name>myvm</name>
  <description>My VM</description>
  <cluster>
    <name>Default</name>
  </cluster>
  <template>
    <name>Blank</name>
  </template>
</vm>
```

При использовании для вывода основные параметры включаются в тело ответа. Например, при добавлении виртуальной машины в тело ответа будет включен параметр `vm`. Таким образом, полное тело ответа будет выглядеть так:

```
HTTP/1.1 201 Created
Content-Type: application/xml

<vm href="/ovirt-engine/api/vms/123" id="123">
  <name>myvm</name>
  <description>My VM</description>
  ...
</vm>
```

Вторичные параметры разрешены только для ввода (за исключением методов действий, которые описаны ниже), и они должны быть включены в качестве параметров запроса. Например, при добавлении виртуальной машины с параметром `clone`, установленным на `true`, полный запрос должен выглядеть так:

```
POST /ovirt-engine/api/vms?clone=true HTTP/1.1
Host: myengine.example.com
Authorization: Bearer fqbR1ftzh8wBCviLxJcYuV5oSDI=
Content-Type: application/xml
Accept: application/xml

<vm>
  <name>myvm</name>
  <description>My VM</description>
  <cluster>
    <name>Default</name>
  </cluster>
  <template>
    <name>Blank</name>
  </template>
</vm>
```

Методы **action** имеют только вторичные параметры. Их можно использовать для ввода и вывода, и они должны быть включены в тело запроса, обернутые элементом `action`. Например, метод **action**, используемый для запуска виртуальной машины, имеет параметр `vm`, описывающий, как следует запускать виртуальную машину, и параметр `use_cloud_init`, указывающий, следует ли использовать cloud-init для настройки гостевой операционной системы. Таким образом, полный запрос на запуск виртуальной машины **123** с помощью **cloud-init** при использовании XML будет выглядеть следующим образом:

```
POST /ovirt-engine/api/vms/123/start HTTP/1.1
Host: myengine.example.com
Authorization: Bearer fqbR1ftzh8wBCviLxJcYuV5oSDI=
Content-Type: application/xml
Accept: application/xml

<action>
  <use_cloud_init>true</use_cloud_init>
  <vm>
    <initialization>
      <nic_configurations>
        <nic_configuration>
          <name>eth0</name>
          <on_boot>true</on_boot>
          <boot_protocol>static</boot_protocol>
          <ip>
            <address>192.168.0.100</address>
            <netmask>255.255.255.0</netmask>
            <gateway>192.168.0.1</netmask>
          </ip>
        </nic_configuration>
      </nic_configurations>
      <dns_servers>192.168.0.1</dns_servers>
    </initialization>
  </vm>
</action>
```

7.1. Вложенные сервисы

Некоторые сервисы могут быть определены как самостоятельные или быть вложенными в другие сервисы. Параметры запросов к вложенным сервисам, как правило, соответствуют базовому применению.

Далее приводится список перечень дочерних и родительских сервисов.

- **/datacenters/{dc:id}**
    - **/clusters** и все запросы, описанные в группе **Кластеры**.

Применение сервисов **/clusters** в сервисе **/datacenters/{dc:id}** позволяет выполнять операции только с кластерами, относящимися к соответствующему центру данных.

Например, GET-запрос `/ovirt-engine/api/clusters` возвращает все кластеры, присутствующие в системе, а GET-запрос `/ovirt-engine/api/datacenters/{dc:id}/clusters` - только кластеры в центре данных с указанным ID.

- **/storagedomains** и все относящиеся к нему запросы, описанные в группе **Домены хранения** (сервисы **/storagedomains**), за исключением некоторых методов, описанных в группе **Центры данных**.

  Применение сервисов **/storagedomains** в сервисе **/datacenters/{dc:id}** позволяет выполнять операции только с доменами хранения, относящимися к соответствующему центру данных.

  Например, GET-запрос `/ovirt-engine/api/storagedomains` возвращает все существующие домены хранения, присутствующие в системе, а GET-запрос `/datacenters/{datacenter:id}/storagedomains` - только домены хранения в соответствующем центре данных.

- **/datacenters/{datacenter:id}/iscsibonds/{iscsibond:id}**
  - **/networks** и все относящиеся к нему запросы, описанные в группе **логические сети**, за исключением POST-запроса `/datacenters/{datacenter:id}/iscsibonds/{iscsibond:id}/networks`.

    Применение сервисов **/networks** в сервисе **/datacenters/{dc:id}/iscsibonds/{iscsi-bond:id}** позволяет выполнять операции только с сетями, относящимися к соответствующему iSCSI-бонду.

    Например, GET-запрос `/ovirt-engine/api/networks` возвращает все логические сети, присутствующие в системе, а GET-запрос `/datacenters/{datacenter:id}/iscsibonds/{iscsibond:id}/networks` - только сети в iSCSI-бонде с указанным ID.

  - **/storageserverconnections** и все относящиеся к нему запросы, описанные в группе **Домены хранения** (сервис **/storageconnections**), за исключением POST-запроса `/datacenters/{datacenter:id}/iscsibonds/{iscsibond:id}/storageserverconnections`.

    Применение сервисов **/storageserverconnections** в сервисе **/datacenters/{dc:id}/iscsibonds/{iscsibond:id}** позволяет выполнять операции только с соединениями доменов хранения, относящимися к соответствующему iSCSI-бонду.

    Например, GET-запрос `/ovirt-engine/api/storageconnections` возвращает все существующие соединения доменов хранения, присутствующие в системе, а GET-запрос

`/datacenters/{datacenter:id}/iscsibonds/{iscsibond:id}/storageserverconnections` - только соединения в iSCSI-бонде с указанным ID.

- **/groups**
  - **/roles** и все относящиеся к нему запросы, описанные в группе **Разрешения и роли**.

    Применение сервисов **/roles** в сервисе **/groups/{group:id}** позволяет выполнять операции только с ролями, назначенными указанной группе.

    Например, GET-запрос `/ovirt-engine/api/roles` возвращает все существующие роли, присутствующие в системе, а GET-запрос `/groups/{group:id}/roles` - только роли, назначенные указанной группе.
- **/users**
  - **/roles** и все относящиеся к нему запросы, описанные в группе **Разрешения и роли**.

    Применение сервисов **/roles** в сервисе **/users/{user:id}** позволяет выполнять операции только с ролями, назначенными указанному пользователю.

    Например, GET-запрос `/ovirt-engine/api/roles` возвращает все существующие роли, присутствующие в системе, а GET-запрос `/users/{user:id}/roles` - только роли, назначенные указанному пользователю.
- **/networks**
  - **/vnicprofiles** и все относящиеся к нему запросы, описанные в группе **Профили vNIC**.

    Применение сервисов **/vnicprofiles** в сервисе **/networks/{network:id}** позволяет выполнять операции только с профилями vNIC, принадлежащими указанной логической сети.

    Например, GET-запрос `/ovirt-engine/api/vnicprofiles` возвращает все существующие профили vNIC, присутствующие в системе, а GET-запрос `/networks/{network:id}/vnicprofiles` - только профили vNIC, принадлежащие указанной логической сети.
- **/storagedomains**
  - **/diskprofiles** и все относящиеся к нему запросы, описанные в группе **Диски и профили дисков**.

    Применение сервисов **/diskprofiles** в сервисе **/storagedomains/{storagedomain:id}** позволяет выполнять операции только с профилями дисков, принадлежащими указанному домену хранения.

    Например, GET-запрос `/ovirt-engine/api/diskprofiles` возвращает все существующие профили дисков, присутствующие в системе, а GET-запрос `/storagedomains/{storagedomain:id}/diskprofiles` - только профили, принадлежащие указанному домену хранения.

- **/disks** и все относящиеся к нему запросы, описанные в группе **Диски и профили дисков**.

  Применение сервисов **/disks** в сервисе **/storagedomains/{storagedomain:id}** позволяет выполнять операции только с образами дисков, принадлежащими указанному домену хранения.

  Например, GET-запрос `/ovirt-engine/api/disks` возвращает все существующие образы дисков, присутствующие в системе, а GET-запрос `/storagedomains/{storagedomain:id}/disks` - только образы, принадлежащие указанному домену хранения.

- **/storageconnections** и все относящиеся к нему запросы, описанные в группе **Домены хранения**.

  Применение сервисов **/storageconnections** в сервисе **/storagedomains/{storagedomain:id}** позволяет выполнять операции только с соединениями хранилищ, принадлежащими указанному домену хранения.

  Например, GET-запрос `/ovirt-engine/api/storageconnections` возвращает все существующие соединения хранилищ, присутствующие в системе, а GET-запрос `/storagedomains/{storagedomain:id}/storageconnections` - только соединения, принадлежащие указанному домену хранения.

8. Управление разрешениями

Управление разрешениями возможно с помощью сервиса **permissions**. Каждое разрешение содержит ссылки на пользователя или группу, роль и объект. Например, разрешения, назначенные конкретной виртуальной машине, можно получить, отправив такой запрос:

```
GET /ovirt-engine/api/vms/123/permissions
```

Тело ответа будет выглядеть так:

```xml
<permissions>
  <permission id="456" href="/ovirt-engine/api/vms/123/permissions/456">
    <user id="789" href="/ovirt-engine/api/users/789"/>
    <role id="abc" href="/ovirt-engine/api/roles/abc"/>
    <vm id="123" href="/ovirt-engine/api/vms/123"/>
  </permission>
  ...
</permissions>
```

Разрешение добавляется к объекту отправкой POST-запроса с представлением разрешения в этот сервис. Для каждого нового разрешения требуется роль и пользователь.

Сервис **permissions** может использоваться следующим образом:

- Самостоятельно (`/ovirt-engine/api/permissions`): при таком использовании предполагается управление правами на уровне всей системы.
- Как вложенный сервис: при таком использовании предполагается управление правами для конкретного объекта.

Поскольку вложенное использование сервиса **permissions** идентично самостоятельному, в разделе Запросы отсутствует описание его применения на уровне отдельных объектов. Описание самостоятельного применения смотрите в группе **Разрешения и роли**.

Вложенное использование сервиса **permissions** допустимо по следующим путям:

- **/clusters/{cluster:id}** (т.е. `/clusters/{cluster:id}/permissions` , далее будут описаны только пути к родительским объектам) - управление правами на уровне кластера.
- **/cpuprofiles/{profile:id}** - управление правами на уровне профиля ЦП.
- **/datacenters/{datacenter:id}** - управление правами на уровне центра данных.
- **/datacenters/{datacenter:id}/quotas/{quota:id}** - управление правами на уровне квоты центра данных.
- **/diskprofiles/{diskprofile:id}** - управление правами на уровне профиля диска.
- **/disks/{disk:id}** - управление правами на уровне диска (включая возможность использования с сервисом **disks** в конкретном домене хранения).
- **/groups/{group:id}** - управление правами группы.
- **/hosts/{host:id}** - управление правами на уровне хоста.
- **/macpools/{macpool:id}** - управление правами на уровне пула MAC-адресов.
- **/networks/{network:id}** - управление правами на уровне логической сети (включая возможность использования с сервисом **networks** в конкретном центре данных).
- **/storagedomains/{storagedomain:id}** - управление правами на уровне домена хранения.
- **/templates/{template:id}** - управление правами на уровне шаблона ВМ.
- **/users/{user:id}** - управление правами пользователя.
- **/vmpools/{pool:id}** - управление правами на уровне пула ВМ.
- **/vms/{vm:id}** - управление правами на уровне ВМ.
- **/vnicprofiles/{profile:id}** - управление правами на уровне профиля vNIC (включая возможность использования с сервисом **vnicprofiles** в конкретной сети).

9. Поиск

Метод `list` некоторых сервисов имеет параметр `search`, который можно использовать для указания критериев поиска. При использовании сервер будет возвращать только те объекты коллекции, которые удовлетворяют этим критериям. Например, следующий запрос вернет только виртуальную машину с именем **myvm**:

```
GET /ovirt-engine/api/vms?search=name%3Dmyvm
```

9.1. Параметр 'max'

Используйте параметр `max`, чтобы ограничить количество возвращаемых объектов. Например, следующий запрос вернет только одну виртуальную машину, независимо от того, сколько их доступно в системе:

```
GET /ovirt-engine/api/vms?max=1
```

Поисковый запрос без параметра `max` вернет все объекты. Указание параметра `max` рекомендуется для снижения влияния запросов на общую производительность системы.

9.2. Чувствительность к регистру

По умолчанию запросы не чувствительны к регистру. Например, следующий запрос вернет виртуальные машины с именами **myvm**, **MyVM** и **MYVM**:

```
GET /ovirt-engine/api/vms?search=name%3Dmyvm
```

Необязательный логический параметр `case_sensitive` можно использовать для изменения этого поведения. Например, чтобы получить именно виртуальную машину с именем **myhost**, а не **MyHost** или **MYHOST**, отправьте такой запрос:

```
GET /ovirt-engine/api/vms?search=name%3D=myvm&case_sensitive=true
```

9.3. Синтаксис поиска

Параметры `search` используют тот же синтаксис, что и язык запросов zVirt:

```
(criteria) [sortby (element) asc|desc]
```

Предложение `sortby` является необязательным и требуется только для упорядочивания результатов.

Таблица 1. Примеры поисковых запросов

Коллекция	Критерии	Результат
`hosts`	`vms.status=up`	Возвращает список всех хостов, на которых запущены виртуальные машины.
`vms`	`domain=example.com`	Возвращает список всех виртуальных машин, работающих в указанном домене.
`vms`	`users.name=mary`	Возвращает список всех виртуальных машин, принадлежащих пользователям с именем пользователя `mary`.
`events`	`severity > normal sortby time`	Возвращает список всех событий с серьезностью выше, чем `normal` и отсортированных по значению атрибута `time`.
`events`	`severity > normal sortby time desc`	Возвращает список всех событий с серьезностью выше, чем `normal`и отсортированных по значению их атрибута `time` в порядке убывания.

Значение параметра `search` должно быть представлено в URL кодировке для перевода зарезервированных символов, таких как операторы и пробелы. Например, знак равенства (=) должен быть закодирован как `%3D`:

```
GET /ovirt-engine/api/vms?search=name%3Dmyvm
```

9.4. Подстановочные знаки

Звездочка (*) может использоваться как часть значения, чтобы указать, что любая строка соответствует, включая пустую строку. Например, следующий запрос вернет все виртуальные машины с именами, начинающимися с **myvm**, например **myvm**, **myvm2**, **myvma** или **myvm-webserver**:

`GET /ovirt-engine/api/vms?search=name%3Dmyvm*`

9.5. Пагинация

Некоторые среды zVirt содержат большие коллекции объектов. Получение всех их одним запросом нецелесообразно и снижает производительность. Чтобы разрешить извлечение их страница за страницей, параметр `search` поддерживает необязательную опцию `page`. Это, в сочетании с параметром `max`, является основой для разделения на страницы. Например, чтобы получить первую страницу виртуальных машин с размером страницы 10 виртуальных машин, отправьте запрос следующим образом:

```
GET /ovirt-engine/api/vms?search=page%201&max=10
```

Параметр поиска представлен в URL кодировке, фактическое значение параметра `search` до кодирования равно `page 1`, что указывает на запрос первой страницы.

Увеличьте значение `page`, чтобы получить следующую страницу:

```
GET /ovirt-engine/api/vms?search=page%202&max=10
```

Опцию `page` можно использовать вместе с другими опциями внутри параметра `search`. Например, следующий запрос вернет вторую страницу виртуальных машин, но с сортировкой по имени:

```
GET /ovirt-engine/api/vms?search=sortby%20name%20page%202&max=10
```

API не сохраняет состояния между различными запросами, поскольку все запросы независимы друг от друга. В результате, если между вашими запросами происходит изменение статуса, результаты страницы могут быть несогласованными.

Например, если вы запросили определенную страницу из списка виртуальных машин, а до того, как успели запросить следующую, были созданы или удалены виртуальные машины, то в результатах нового запроса могут отсутствовать некоторые из них или содержаться дубликаты.

10. Ссылки

API возвращает указания на связанные объекты в виде **ссылок**. Например, данные о виртуальной машине содержат ссылки на присоединённые к ней диски и сетевые карты:

XML
```xml
<vm id="123" href="/ovirt-engine/api/vms/123">
  ...
  <link rel="diskattachments" href="/ovirt-engine/api/vms/123/diskattachments"/>
  <link rel="nics" href="/ovirt-engine/api/vms/123/nics"/>
  ...
</vm>
```

Полное описание этих связанных объектов можно получить, отправив отдельные запросы:

```
GET /ovirt-engine/api/vms/123/diskattachments
GET /ovirt-engine/api/vms/123/nics
```

Однако в некоторых ситуациях приложению, использующему API, удобнее извлекать связанную информацию в том же запросе. Это полезно, например, когда дополнительные

циклы передачи данных по сети приводят к неприемлемым издержкам или когда множественные запросы усложняют код приложения неприемлемым образом. Для этих случаев использования API предоставляет параметр `follow`, который позволяет приложению извлекать связанную информацию, используя только один запрос.

Значение параметра `follow` представляет собой список строк, разделенных запятыми. Каждая из этих строк является путем связанного объекта. Например, чтобы получить присоединённые диски и сетевые карты в приведенном выше примере, запрос должен быть таким:

```
GET /ovirt-engine/api/vms/123?follow=disk_attachments,nics
```

Это вернет ответ, подобный следующему:

```xml
<vm id="123" href="/ovirt-engine/api/vms/123">
  ...
  <disk_attachments>
    <disk_attachment id="456" href="/ovirt-
engine/api/vms/123/diskattachments/456">
      <active>true</active>
      <bootable>true</bootable>
      <interface>virtio_scsi</interface>
      <pass_discard>false</pass_discard>
      <read_only>false</read_only>
      <uses_scsi_reservation>false</uses_scsi_reservation>
      <disk id="789" href="/ovirt-engine/api/disks/789"/>
    </disk_attachment>
    ...
  </disk_attachments>
  <nics>
    <nic id="234" href="/ovirt-engine/api/vms/123/nics/234">
      <name>eth0</name>
      <interface>virtio</interface>
      <linked>true</linked>
      <mac>
        <address>00:1a:4a:16:01:00</address>
      </mac>
      <plugged>true</plugged>
    </nic>
    ...
  </nics>
  ...
</vm>
```

Путь к связанному объекту может быть одним словом, как в предыдущем примере, или последовательностью слов, разделенных точками, для запроса вложенных данных. Например, в предыдущем примере использовался параметр `disk_attachments` для

получения полного описания присоединённых дисков, но каждый блок `disk_attachments` содержит нераскрытую ссылку на диск (в примере выше: `<disk id="789" href="/ovirt-engine/api/disks/789"/>`). Для того, чтобы также перейти по ссылкам на диски, можно использовать следующий запрос:

```
GET /ovirt-engine/api/vms/123?follow=disk_attachments.disk
```

Это приведет к следующему ответу:

```xml
<vm id="123" href="/ovirt-engine/api/vms/123">
  <disk_attachments>
    <disk_attachment id="456" href="/ovirt-
engine/api/vms/123/diskattachments/456">
      <active>true</active>
      <bootable>true</bootable>
      <interface>virtio_scsi</interface>
      <pass_discard>false</pass_discard>
      <read_only>false</read_only>
      <uses_scsi_reservation>false</uses_scsi_reservation>
      <disk id="789" href="/ovirt-engine/api/disks/789">
        <name>mydisk</name>
        <description>My disk</description>
        <actual_size>0</actual_size>
        <format>raw</format>
        <sparse>true</sparse>
        <status>ok</status>
        <storage_type>image</storage_type>
        <total_size>0</total_size>
        ...
      </disk>
    </disk_attachment>
    ...
  </disk_attachments>
  ...
</vm>
```

Путь можно сделать настолько глубоким, насколько это необходимо. Например, чтобы также получить статистику дисков:

```
GET /ovirt-engine/api/vms/123?follow=disk_attachments.disk.statistics
```

Можно комбинировать несколько элементов пути и несколько путей. Например, чтобы получить присоединённые диски и сетевые карты со статистикой:

```
GET /ovirt-engine/api/vms/123?
follow=disk_attachments.disk.statistics,nics.statistics
```

Почти все операции, извлекающие объекты, поддерживают этот параметр `follow`, но обязательно ознакомьтесь со справочной документацией, так как некоторые операции могут не поддерживать его или давать рекомендации по его использованию для достижения наилучшей производительности.

Использование параметра `follow` перемещает нагрузку со стороны клиента на сторону сервера. Когда вы запрашиваете дополнительные данные, сервер должен получить и объединить их с основными данными. Это потребляет ЦП и память на стороне сервера и в большинстве случаев требует дополнительных запросов к базе данных. Это может отрицательно сказаться на производительности сервера, особенно в крупномасштабных средах. Обязательно протестируйте приложение в реалистичной среде и используйте параметр `follow` только тогда, когда это оправдано.

11. Обработка ошибок

Некоторые ошибки требуют дополнительного объяснения, помимо стандартного кода состояния HTTP. Например, API сообщает о неудачном обновлении состояния объекта или действии с помощью элемента `fault` в тексте ответа. Элемент `fault` содержит атрибуты `reason` и `detail`. Например, когда сервер получает запрос на создание виртуальной машины без обязательного атрибута `name`, он вернёт следующую строку ответа HTTP:

```
HTTP/1.1 400 Bad Request
```

И следующее тело ответа:

```
<fault>
  <reason>Incomplete parameters</reason>
  <detail>Vm [name] required for add</detail>
</fault>
```

Таблица 2. Интерпретация основных кодов

Код	Статус	Описание
200	OK	Указывает на успешное выполнение **GET**, **PUT** и **DELETE** запросов. **PUT** и **DELETE** запросы возвращают код **200** только в случае синхронного выполнения (параметр **async** либо не указан, либо имеет значение **false**).
201	Created	Указывает на успешное выполнение **POST**-запроса

Код	Статус	Описание
202	Accepted	Указывает на успешное создание задачи путем отправки асинхронного **PUT** или **DELETE** запроса (параметр **async** имеет значение **true**). Возврат данного кода не означает, что операция выполнена успешно. Его получение указывает только на тот факт, что запрос успешно принят.
400	Bad Request	Указывает на некорректный запрос. Как правило, является результатом ошибок в теле PUT и POST запросов. Например, код 400 возвращается при передаче в теле запроса неверного параметра (`naem` вместо `name`) или его значения (`internal` вместо `internal-authz` при указании домена аутентификации), а также при отсутствии в теле требуемого параметра.
401	Unauthorized	Указывает на необходимость авторизации для выполнения запроса. Как правило возникает в следующих случаях: • Указан некорректный метод авторизации (например, **NoAuth**). • Истек срок действия токена авторизации. • Пользователь, использованный для авторизации, не имеет необходимых прав на выполнение запроса.
404	Not Found	Указывает на то, что объект, к которому производится запрос, не найден. Как правило возникает при некорректно указанном ID объекта.
405	Method Not Allowed	Указывает на то, что запрошенный метод недоступен для объекта. Например, **DELETE** запрос к **/groups** (обращение к сервису, указывающему на все группы, а не к конкретной группе **/groups/{group:id}**)
409	Conflict	Указывает на невозможность выполнения операции. Может возникать при невыполнении требований для совершения операции, например: • Передача в теле **POST** запроса ссылки на несуществующий объект (например, код **409** будет возвращен при создании iSCSI-бонда с указанием несуществующей в центре данных логической сети). • Запрос метода, который требует предварительного выполнения дополнительных операции (например, для сброса эмулируемой машины в кластере необходимо выключить все хосты). • Операция не имеет смысла (например, попытка передать роль master на домен хранения, который уже является мастер-доменом).

Введение в REST API

Менеджер управления включает **Representational State Transfer (REST) API**. API предоставляет разработчикам программного обеспечения и системным администраторам контроль над своей средой zVirt за пределами стандартного веб-интерфейса. API полезен для интеграции функций среды zVirt с пользовательскими сценариями или внешними приложениями, которые обращаются к API через стандартный протокол передачи гипертекста (HTTP).

Преимущества API

- Широкая клиентская поддержка. Любой язык программирования, платформа или система с поддержкой протокола HTTP могут использовать API.
- Самоописательный — клиентские приложения требуют минимальных знаний об инфраструктуре виртуализации, так как многие детали обнаруживаются во время выполнения.
- Модель на основе ресурсов. Модель REST на основе ресурсов обеспечивает естественный способ управления платформой виртуализации.

Это дает разработчикам и администраторам возможность

- Интеграции с корпоративными ИТ-системами.
- Интеграции со сторонним программным обеспечением для виртуализации.
- Выполнения автоматизированного обслуживания или проверки ошибок.
- Автоматизации повторяющихся задач в среде zVirt с помощью сценариев.

Эта документация служит справочником по API zVirt. Он предназначен для предоставления разработчикам и администраторам инструкций и примеров, которые помогут использовать функциональные возможности их среды zVirt через API либо напрямую, либо с использованием предоставленных SDK.

1. REST

Representational State Transfer (REST) — это архитектура проектирования, ориентированная на ресурсы для конкретной службы и их представления. Представление ресурсов — это ключевая абстракция информации, которая соответствует одному конкретному управляемому элементу на сервере. Клиент отправляет запрос элементу сервера, расположенному по универсальному идентификатору ресурса (URI), и выполняет операции, используя стандартные методы HTTP, такие как `GET`, `POST`, `PUT` и `DELETE`. Это обеспечивает связь между клиентом и сервером без сохранения состояния, при которой

каждый запрос действует независимо от любого другого запроса и содержит всю информацию, необходимую для выполнения запроса.

2. Предварительные требования к API

Предварительные требования для использования zVirt API:

- Сетевая установка zVirt Engine, включающая API.
- Клиент или программная библиотека, которая инициирует и получает HTTP-запросы от сервера API. Например:
  - SDK oVirt для Python.
  - SDK oVirt для Ruby.
  - SDK oVirt для Java.
  - Инструмент командной строки cURL.
  - RESTClient — отладчик для веб-служб RESTful.
- Знание протокола передачи гипертекста (HTTP), протокола, используемого для взаимодействия с REST API. Инженерная рабочая группа Интернета публикует запрос комментариев (RFC) с объяснением протокола передачи гипертекста по адресу http://www.ietf.org/rfc/rfc2616.txt.
- Знание расширяемого языка разметки (XML) или нотации объектов JavaScript (JSON), которые API использует для создания представлений ресурсов. W3C предоставляет полную спецификацию XML на http://www.w3.org/TR/xml. ECMA International предоставляет бесплатную публикацию по JSON на http://www.ecma-international.org.

Примеры использования REST API

Примеры в этом разделе показывают, как использовать REST API для настройки базовой среды zVirt и создания виртуальной машины. В дополнение к стандартным предварительным требованиям для этих примеров требуется следующее:

- Установленная и настроенная среда zVirt, имеющая доступ к сети.
- Файл ISO, содержащий операционную систему виртуальной машины, которую вы хотите установить. В этой главе в качестве примера установки ISO используется CentOS 7.

В примерах используется curl для демонстрации запросов API с помощью клиентского приложения. Вы можете использовать любое приложение, которое отправляет HTTP-запросы.

Заголовки HTTP-запросов в этом примере опускают заголовки `Host` и `Authorization`. Однако эти поля являются обязательными и требуют данных, специфичных для вашей установки zVirt.

В curl примерах используется имя пользователя **api-user@internalsso**, пароль **mypassword**, расположение сертификата **/etc/pki/ovirt-engine/ca.pem** и имя хоста **myengine.example.com**. Вы должны заменить их правильными значениями для вашей среды.

zVirt генерирует уникальный идентификатор атрибута `id` для каждого ресурса. Идентификационные коды в этом примере будут отличаться от идентификационных кодов в вашей среде zVirt.

Во многих примерах некоторые атрибуты результатов, возвращаемых API, для краткости опущены.

1. Доступ к точке входа API

Следующий запрос извлекает представление основной точки входа для API версии 4:

```
GET /ovirt-engine/api HTTP/1.1
Version: 4
Accept: application/xml
```

Тот же запрос, но с использованием префикса URL `/v4` вместо заголовка `Version`:

```
GET /ovirt-engine/api/v4 HTTP/1.1
Accept: application/xml
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--request GET \
--header 'Version: 4' \
--header 'Accept: application/xml' \
--user 'api-user@internalsso:mypassword' \
https://myengine.example.com/ovirt-engine/api
```

Результатом является объект типа **Api**:

XML

```
<api>
    <link href="/ovirt-engine/api/clusters" rel="clusters"/>
    <link href="/ovirt-engine/api/clusters?search={query}"
rel="clusters/search"/>
    ...
    <product_info>
        <instance_id>6fdd60c6-fdfd-11ed-94ee-00163e232a46</instance_id>
        <name>zVirt Engine</name>
        <version>
            <build>9</build>
            <full_version>4.4.9.5-1.1587.zvirt.el7</full_version>
            <major>4</major>
            <minor>4</minor>
            <revision>0</revision>
        </version>
    </product_info>
    <special_objects>
        <blank_template href="..." id="..."/>
        <root_tag href="..." id="..."/>
    </special_objects>
    <summary>
        <hosts>
            <active>3</active>
            <total>3</total>
        </hosts>
        <storage_domains>
            <active>2</active>
            <total>3</total>
        </storage_domains>
        <users>
            <active>2</active>
            <total>2</total>
        </users>
        <vms>
            <active>1</active>
            <total>1</total>
        </vms>
    </summary>
```

```
    <time>2023-06-02T18:09:00.587+03:00</time>
    ...
</api>
```

Если не используются ни заголовок, ни префикс URL, сервер автоматически выберет версию. Версия по умолчанию - 4. Вы можете изменить версию по умолчанию с помощью параметра конфигурации `ENGINE_API_DEFAULT_VERSION`:

```
echo "ENGINE_API_DEFAULT_VERSION=3" > \
/etc/ovirt-engine/engine.conf.d/99-set-default-version.conf

systemctl restart ovirt-engine
```

Изменение этого параметра влияет на всех пользователей API, не указывающих версию явно.

Точка входа предоставляет пользователю ссылки (`link`) на коллекции в среде виртуализации. Атрибут `rel` каждой ссылки на коллекцию предоставляет контрольную точку для каждой ссылки. На следующем шаге в этом примере исследуется коллекция центра данных, доступная по ссылке `datacenters` .

Точка входа также содержит другие данные, такие как `product_info` , `special_objects` и `summary` .

2. Список центров данных

zVirt при установке создает центр данных **Default**. В этом примере центр данных **Default** используется в качестве основы для виртуальной среды.

Следующий запрос извлекает представление центров данных:

```
GET /ovirt-engine/api/datacenters HTTP/1.1
Accept: application/xml
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--request GET \
--header 'Version: 4' \
--header 'Accept: application/xml' \
--user 'api-user@internalsso:mypassword' \
https://myengine.example.com/ovirt-engine/api/datacenters
```

Результатом будет список объектов типа **DataCenter**:

```xml
<data_centers>
    <data_center href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46" id="6e1b9a78-fdfd-11ed-afe5-00163e232a46">
        <actions>
            <link href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46/setmaster" rel="setmaster"/>
            <link href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46/cleanfinishedtasks" rel="cleanfinishedtasks"/>
        </actions>
        <name>Default</name>
        <description>The default Data Center</description>
        <link href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46/storagedomains" rel="storagedomains"/>
        <link href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46/permissions" rel="permissions"/>
        <link href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46/clusters" rel="clusters"/>
        ...
        <local>false</local>
        <quota_mode>disabled</quota_mode>
        <status>up</status>
        <storage_format>v5</storage_format>
        <supported_versions>
            <version>
                <major>4</major>
                <minor>6</minor>
            </version>
        </supported_versions>
        <version>
            <major>4</major>
            <minor>6</minor>
        </version>
    </data_center>
</data_centers>
```

Обратите внимание на `id` вашего центра данных **Default**. Он идентифицирует этот центр данных по отношению к другим ресурсам вашей виртуальной среды.

Центр данных также содержит ссылку на сервис **AttachedStorageDomains**, который управляет доменами хранения, подключенными к центру данных:

```xml
<link href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46/storagedomains" rel="storagedomains"/>
```

Этот сервис используется для присоединения доменов хранения из основной коллекции `storagedomains`, что будет рассмотрено в этом примере позже.

3. Получение списка кластеров

zVirt при установке создает кластер **Default**. В этом примере кластер **Default** используется для группировки ресурсов в среде zVirt.

Следующий запрос извлекает представление коллекции кластеров:

```
GET /ovirt-engine/api/clusters HTTP/1.1
Accept: application/xml
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--request GET \
--header 'Version: 4' \
--header 'Accept: application/xml' \
--user 'api-user@internalsso:mypassword' \
https://myengine.example.com/ovirt-engine/api/clusters
```

Результатом будет список объектов типа **Cluster**:

XML

```
<clusters>
    <cluster href="/ovirt-engine/api/clusters/6e1d0a3e-fdfd-11ed-ad51-
00163e232a46" id="6e1d0a3e-fdfd-11ed-ad51-00163e232a46">
        <name>Default</name>
        <description>The default server cluster</description>
        <comment></comment>
        <link href="/ovirt-engine/api/clusters/6e1d0a3e-fdfd-11ed-ad51-
00163e232a46/permissions" rel="permissions"/>
        <link href="/ovirt-engine/api/clusters/6e1d0a3e-fdfd-11ed-ad51-
00163e232a46/networks" rel="networks"/>
        ...
        <cpu>
            <architecture>x86_64</architecture>
            <type>Intel Icelake Server Family</type>
        </cpu>
        <version>
            <major>4</major>
            <minor>6</minor>
        </version>
        <virt_service>true</virt_service>
        <data_center href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-
afe5-00163e232a46" id="6e1b9a78-fdfd-11ed-afe5-00163e232a46"/>
    </cluster>
</clusters>
```

Обратите внимание на `id` своего кластера **Default**. Он идентифицирует этот кластер по отношению к другим ресурсам вашей виртуальной среды.

Кластер **Default** ассоциируется с центрами данных **Default** посредством связи с использованием атрибутов `id` и `href` ссылки `data_center`:

```XML
<data_center href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-afe5-00163e232a46" id="6e1b9a78-fdfd-11ed-afe5-00163e232a46"/>
```

Ссылка `networks` — это ссылка на сервис **DataCenterNetworks**, которая управляет сетями, связанными с этим кластером. В следующем разделе коллекция сетей рассматривается более подробно.

4. Список логических сетей

zVirt при установке по умолчанию создает сеть **ovirtmgmt**. Эта сеть действует как сеть управления Менеджера управления для доступа к хостам.

Эта сеть связана с кластером **Default** и входит в состав центра данных **Default**. В этом примере сеть **ovirtmgmt** используется для подключения виртуальных машин.

Следующий запрос извлекает список логических сетей:

```
GET /ovirt-engine/api/networks HTTP/1.1
Accept: application/xml
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--request GET \
--header 'Version: 4' \
--header 'Accept: application/xml' \
--user 'api-user@internalsso:mypassword' \
https://myengine.example.com/ovirt-engine/api/networks
```

Результатом будет список объектов типа **Network**:

```XML
<networks>
    <network href="/ovirt-engine/api/networks/00000000-0000-0000-0000-000000000009" id="00000000-0000-0000-0000-000000000009">
        <name>ovirtmgmt</name>
        <description>Management Network</description>
        <comment></comment>
        <link href="/ovirt-engine/api/networks/00000000-0000-0000-0000-
```

```
000000000009/permissions" rel="permissions"/>
        <link href="/ovirt-engine/api/networks/00000000-0000-0000-0000-
000000000009/vnicprofiles" rel="vnicprofiles"/>
        <link href="/ovirt-engine/api/networks/00000000-0000-0000-0000-
000000000009/networklabels" rel="networklabels"/>
        <mtu>0</mtu>
        <port_isolation>false</port_isolation>
        <stp>false</stp>
        <usages>
            <usage>vm</usage>
        </usages>
        <vdsm_name>ovirtmgmt</vdsm_name>
        <data_center href="/ovirt-engine/api/datacenters/6e1b9a78-fdfd-11ed-
afe5-00163e232a46" id="6e1b9a78-fdfd-11ed-afe5-00163e232a46"/>
    </network>
</networks>
```

Сеть **ovirtmgmt** подключена к центру данных **Default** через связь, использующую `id` центра данных.

Сеть **ovirtmgmt** также подключена к кластеру **Default** посредством связи в подколлекции сетей кластера.

5. Список хостов

В этом примере извлекается список хостов и отображается хост с именем **host1-kf**, зарегистрированным в среде виртуализации:

```
GET /ovirt-engine/api/hosts HTTP/1.1
Accept: application/xml
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--request GET \
--header 'Version: 4' \
--header 'Accept: application/xml' \
--user 'api-user@internalsso:mypassword' \
https://myengine.example.com/ovirt-engine/api/hosts
```

Результатом будет список объектов типа **Host**:

XML

```
<hosts>
    <host href="/ovirt-engine/api/hosts/5650f5c0-6c01-4967-bce0-76465be8fbf1"
id="5650f5c0-6c01-4967-bce0-76465be8fbf1">
```

```
        <name>host1-kf</name>
        <comment></comment>
        <link href="/ovirt-engine/api/hosts/5650f5c0-6c01-4967-bce0-
76465be8fbf1/nics" rel="nics"/>
        ...
        <address>host1-kf</address>
        <cpu>
            <name>Intel Xeon Processor (Icelake)</name>
            <speed>3000</speed>
            <topology>
                <cores>1</cores>
                <sockets>6</sockets>
                <threads>1</threads>
            </topology>
            <type>Intel Icelake Server Family</type>
        </cpu>
        <memory>16777216000</memory>
        <os>
            <type>RHEL</type>
            <version>
                <build>2</build>
                <full_version>7.3.2 - 40.el7</full_version>
                <major>7</major>
                <minor>3</minor>
            </version>
        </os>
        <port>54321</port>
        <status>up</status>
        <cluster href="/ovirt-engine/api/clusters/6e1d0a3e-fdfd-11ed-ad51-
00163e232a46" id="6e1d0a3e-fdfd-11ed-ad51-00163e232a46"/>
    </host>
</hosts>
```

Обратите внимание на `id` вашего хоста. Он идентифицирует этот хост по отношению к другим ресурсам вашей виртуальной среды.

Этот хост является членом кластера **Default**, а доступная подколлекция `nics` показывает, что этот хост подключен к сети **ovirtmgmt**.

6. Создание домена хранения данных на базе NFS

Домен хранения данных на базе NFS — это экспортированный общий ресурс NFS, подключенный к центру данных и предоставляющий хранилище для виртуализированных гостевых образов. Для создания нового домена хранения требуется запрос `POST`, содержащий представление домена хранения, отправленный по URL-адресу коллекции доменов хранения.

Вы можете включить опцию очистки после удаления по умолчанию в домене хранения. Чтобы настроить это, укажите `wipe_after_delete` в запросе POST. Этот параметр можно изменить после создания домена, но это не изменит свойства очистки после удаления уже существующих дисков.

Запрос должен быть таким:

```
POST /ovirt-engine/api/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

А тело запроса должно выглядеть следующим образом:

XML

```xml
<storage_domain>
  <name>mydata</name>
  <type>data</type>
  <description>My data</description>
  <storage>
    <type>nfs</type>
    <address>mynfs.example.com</address>
    <path>/exports/mydata</path>
  </storage>
  <host>
    <name>myhost</name>
  </host>
</storage_domain>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<storage_domain>
  <name>mydata</name>
  <description>My data</description>
  <type>data</type>
  <storage>
    <type>nfs</type>
    <address>mynfs.example.com</address>
    <path>/exports/mydata</path>
  </storage>
  <host>
    <name>myhost</name>
```

```
  </host>
</storage_domain>
' \
https://myengine.example.com/ovirt-engine/api/storagedomains
```

Сервер использует хост **myhost** для создания домена хранения данных типа NFS **mydata** с именем пути экспорта `mynfs.example.com:/exports/mydata`. API также возвращает следующее представление только что созданного ресурса домена хранения типа **StorageDomain**:

XML

```xml
<storage_domain href="/ovirt-engine/api/storagedomains/005" id="005">
  <name>mydata</name>
  <description>My data</description>
  <available>42949672960</available>
  <committed>0</committed>
  <master>false</master>
  <status>unattached</status>
  <storage>
    <address>mynfs.example.com</address>
    <path>/exports/mydata</path>
    <type>nfs</type>
  </storage>
  <storage_format>v5</storage_format>
  <type>data</type>
  <used>9663676416</used>
</storage_domain>
```

7. Создание домена ISO на базе NFS

Домен ISO на базе NFS — это смонтированная общая папка NFS, подключенная к центру данных и обеспечивающая хранилище для DVD/CD-ROM ISO и файлов образов виртуальных гибких дисков (VFD). Для создания нового домена хранения требуется запрос POST с включенным представлением домена хранения, отправленный по URL-адресу коллекции доменов хранения:

Запрос должен быть таким:

```
POST /ovirt-engine/api/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

А тело запроса должно выглядеть следующим образом:

XML

```xml
<storage_domain>
  <name>myisos</name>
```

```
  <description>My ISOs</description>
  <type>iso</type>
  <storage>
    <type>nfs</type>
    <address>mynfs.example.com</address>
    <path>/exports/myisos</path>
  </storage>
  <host>
    <name>myhost</name>
  </host>
</storage_domain>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<storage_domain>
  <name>myisos</name>
  <description>My ISOs</description>
  <type>iso</type>
  <storage>
    <type>nfs</type>
    <address>mynfs.example.com</address>
    <path>/exports/myisos</path>
  </storage>
  <host>
    <name>myhost</name>
  </host>
</storage_domain>
' \
https://myengine.example.com/ovirt-engine/api/storagedomains
```

Сервер использует хост **myhost** для создания домена ISO на базе NFS, который называется **myisos** с путем экспорта `mynfs.example.com:/exports/myisos`. API также возвращает следующее представление только что созданного ресурса домена хранения типа **StorageDomain**:

```
<storage_domain href="/ovirt-engine/api/storagedomains/006" id="006">
  <name>myiso</name>
  <description>My ISOs</description>
  <available>42949672960</available>
  <committed>0</committed>
  <master>false</master>
```

```
  <status>unattached</status>
  <storage>
    <address>mynfs.example.com</address>
    <path>/exports/myisos</path>
    <type>nfs</type>
  </storage>
  <storage_format>v5</storage_format>
  <type>iso</type>
  <used>9663676416</used>
</storage_domain>
```

8. Прикрепление доменов хранения к центру данных

В следующем примере домены хранения **mydata** и **myisos** подключаются к центру данных **Default**.

Чтобы подключить домен хранения данных **mydata**, отправьте запрос следующего вида:

```
POST /ovirt-engine/api/datacenters/001/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

С таким телом запроса:

XML

```
<storage_domain>
  <name>mydata</name>
</storage_domain>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<storage_domain>
  <name>mydata</name>
</storage_domain>
' \
https://myengine.example.com/ovirt-engine/api/datacenters/001/storagedomains
```

Чтобы подключить домен ISO **myisos**, отправьте запрос следующего вида:

```
POST /ovirt-engine/api/datacenters/001/storagedomains HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

С таким телом запроса:

```xml
<storage_domain>
  <name>myisos</name>
</storage_domain>
```

Тот же запрос с помощью команды curl:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<storage_domain>
  <name>myisos</name>
</storage_domain>
' \
https://myengine.example.com/ovirt-engine/api/datacenters/001/storagedomains
```

9. Создание виртуальной машины

В следующем примере в кластере **Default** на основе шаблона `Blank` создается виртуальная машина с именем **myvm**. Запрос также определяет память виртуальной машины в размере 512 МБ и устанавливает в качестве загрузочного устройства виртуальный жесткий диск.

Запрос должен содержать объект типа **Vm**, описывающий создаваемую виртуальную машину:

```
POST /ovirt-engine/api/vms HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

Тело запроса должно выглядеть следующим образом:

```xml
<vm>
  <name>myvm</name>
```

```
  <description>My VM</description>
  <cluster>
    <name>Default</name>
  </cluster>
  <template>
    <name>Blank</name>
  </template>
  <memory>536870912</memory>
  <os>
    <boot>
      <devices>
        <device>hd</device>
      </devices>
    </boot>
  </os>
</vm>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<vm>
  <name>myvm</name>
  <description>My VM</description>
  <cluster>
    <name>Default</name>
  </cluster>
  <template>
    <name>Blank</name>
  </template>
  <memory>536870912</memory>
  <os>
    <boot>
      <devices>
        <device>hd</device>
      </devices>
    </boot>
  </os>
</vm>
' \
https://myengine.example.com/ovirt-engine/api/vms
```

Тело ответа будет объектом типа **Vm**:

```xml
<vm href="/ovirt-engine/api/vms/007" id="007">
  <name>myvm</name>
  <link href="/ovirt-engine/api/vms/007/diskattachments" rel="diskattachments"/>
  <link href="/ovirt-engine/api/vms/007/nics" rel="nics"/>
  ...
  <cpu>
    <architecture>x86_64</architecture>
    <topology>
      <cores>1</cores>
      <sockets>1</sockets>
      <threads>1</threads>
    </topology>
  </cpu>
  <memory>1073741824</memory>
  <os>
    <boot>
      <devices>
        <device>hd</device>
      </devices>
    </boot>
    <type>other</type>
  </os>
  <type>desktop</type>
  <cluster href="/ovirt-engine/api/clusters/002" id="002"/>
  <status>down</status>
  <original_template href="/ovirt-engine/api/templates/000" id="00"/>
  <template href="/ovirt-engine/api/templates/000" id="000"/>
</vm>
```

10. Создание сетевой карты виртуальной машины

В следующем примере создается виртуальный сетевой интерфейс для подключения примера виртуальной машины к сети **ovirtmgmt**.

Запрос должен быть таким:

```
POST /ovirt-engine/api/vms/007/nics HTTP/1.1
Content-Type: application/xml
Accept: application/xml
```

Тело запроса должно содержать объект типа **Nic**, описывающий создаваемую сетевую карту:

```xml
<nic>
  <name>mynic</name>
```

```
  <description>My network interface card</description>
</nic>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<nic>
  <name>mynic</name>
  <description>My network interface card</description>
</nic>
' \
https://myengine.example.com/ovirt-engine/api/vms/007/nics
```

11. Создание диска виртуальной машины

В следующем примере создается диск с возможностью копирования при записи объемом 8 ГБ для примера виртуальной машины.

Запрос должен быть таким:

```
POST /ovirt-engine/api/vms/007/diskattachments HTTP/1.1
Content-Type: application/xml
Accept: application/xml
```

Тело запроса должно быть объектом типа **DiskAttachment**, описывающим диск и способ его подключения к виртуальной машине:

XML

```
<disk_attachment>
  <bootable>false</bootable>
  <interface>virtio</interface>
  <active>true</active>
  <disk>
    <description>My disk</description>
    <format>cow</format>
    <name>mydisk</name>
    <provisioned_size>8589934592</provisioned_size>
    <storage_domains>
      <storage_domain>
        <name>mydata</name>
```

```
      </storage_domain>
    </storage_domains>
  </disk>
</disk_attachment>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<disk_attachment>
  <bootable>false</bootable>
  <interface>virtio</interface>
  <active>true</active>
  <disk>
    <description>My disk</description>
    <format>cow</format>
    <name>mydisk</name>
    <provisioned_size>8589934592</provisioned_size>
    <storage_domains>
      <storage_domain>
        <name>mydata</name>
      </storage_domain>
    </storage_domains>
  </disk>
</disk_attachment>
' \
https://myengine.example.com/ovirt-engine/api/vms/007/diskattachments
```

Атрибут `storage_domains` указывает API хранить диск в домене хранения данных **mydata**.

12. Прикрепление ISO-образа к виртуальной машине

Загрузочный носитель для следующего примера виртуальной машины требует ISO-образ компакт-диска или DVD для установки операционной системы. В этом примере используется образ CentOS 7.

Для использования виртуальными машинами образы ISO должны быть доступны в домене ISO **myisos**. Вы можете использовать сервисы **ImageTransfers** для создания передачи образа и **ImageTransfer** для загрузки ISO-образа.

После загрузки ISO-образа можно использовать API для запроса списка файлов из домена хранения ISO:

```
GET /ovirt-engine/api/storagedomains/006/files HTTP/1.1
Accept: application/xml
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request GET \
--header 'Version: 4' \
--header 'Accept: application/xml' \
https://myengine.example.com/ovirt-engine/api/storagedomains/006/files
```

Сервер возвращает следующий список объектов типа **File**, по одному для каждого доступного образа ISO:

```xml
<files>
  <file href="..." id="CentOS-7-x86_64-Minimal.iso">
    <name>CentOS-7-x86_64-Minimal.iso</name>
  </file>
  ...
</files>
```

Пользователю API необходимо прикрепить **CentOS-7-x86_64-Minimal.iso** к примеру виртуальной машины. Прикрепление ISO-образа эквивалентно использованию кнопки **Сменить CD** на администрирования или пользовательском портале.

Запрос должен быть таким:

```
PUT /ovirt-engine/api/vms/007/cdroms/00000000-0000-0000-0000-000000000000
HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

Тело запроса должно быть объектом типа **Cdrom**, содержащим внутренний атрибут `file` для указания идентификатора образа ISO:

```xml
<cdrom>
  <file id="CentOS-7-x86_64-Minimal.iso"/>
</cdrom>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request PUT \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<cdrom>
  <file id="CentOS-7-x86_64-Minimal.iso"/>
</cdrom>
' \
https://myengine.example.com/ovirt-engine/api/vms/007/cdroms/00000000-0000-0000-0000-000000000000
```

Дополнительные сведения см. в документации сервиса **VmCdrom**, управляющего компакт-дисками виртуальных машин.

13. Запуск виртуальной машины

Виртуальная среда готова, а виртуальная машина содержит все необходимые для работы компоненты. В этом примере виртуальная машина запускается с помощью метода `start`.

Запрос должен быть таким:

```
POST /ovirt-engine/api/vms/007/start HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

Тело запроса должно выглядеть следующим образом:

```xml
<action>
  <vm>
    <os>
      <boot>
        <devices>
          <device>cdrom</device>
        </devices>
      </boot>
    </os>
  </vm>
</action>
```

Тот же запрос с помощью команды `curl`:

```
curl \
--cacert '/etc/pki/ovirt-engine/ca.pem' \
--user 'api-user@internalsso:mypassword' \
--request POST \
--header 'Version: 4' \
--header 'Content-Type: application/xml' \
--header 'Accept: application/xml' \
--data '
<action>
  <vm>
    <os>
      <boot>
        <devices>
          <device>cdrom</device>
        </devices>
      </boot>
    </os>
  </vm>
</action>
' \
https://myengine.example.com/ovirt-engine/api/vms/007/start
```

В теле запроса в качестве загрузочного устройства виртуальной машины задается CD-ROM только для этой загрузки. Это позволяет виртуальной машине установить операционную систему из прикрепленного ISO-образа. Загрузочное устройство возвращается к диску для всех будущих загрузок.

Запросы к сервисам REST API oVirt

Архитектура среды виртуализации zVirt

Среда виртуализации zVirt может быть развернута как в архитектуре **Hosted Engine**, так и в **Standalone**. Рекомендуемая архитектура - **Hosted Engine**.

1. Архитектура Hosted Engine

Архитектура развертывания **Hosted Engine** предполагает развертывание баз данных и сервисов Менеджера управления внутри специальной виртуальной машины - **ВМ HostedEngine**. **ВМ HostedEngine** запускается на специальных хостах (**хостах с ролью HostedEngine**), управляемых этим Менеджером.

Основное преимущество архитектуры **Hosted Engine** состоит в том, что отсутствует необходимость в отдельном хосте с ролью менеджера управления (Standalone). Кроме того, **ВМ HostedEngine** может работать в режиме высокой доступности.

Минимальная конфигурация среды включает:

- Одна виртуальная машина с Менеджером управления, размещенная на хостах с ролью **Hosted Engine**.
- Один (или два для режима высокой доступности **ВМ HostedEngine**) хост.
- Внешняя система хранения данных (СХД) или локальное хранилище для размещения домена хранения данных (хранилища). Хранилище должно быть доступно всем хостам виртуализации.

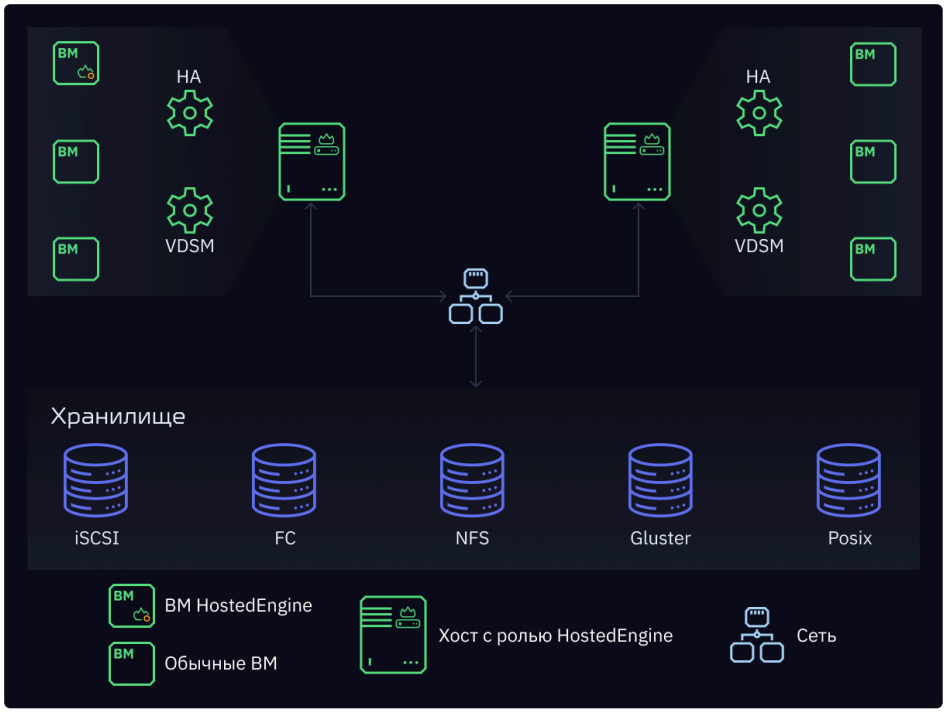

Рисунок 1. Архитектура виртуализации zVirt Hosted Engine

2. Архитектура Standalone

Архитектура развертывания Standalone предполагает развертывание баз данных и сервисов Менеджера управления на физическом сервере - **Standalone-хосте**, с установленной средой исполнения zVirt Node.

Технически также возможно использование в качестве **Standalone-хоста** виртуальной машины, работающей в существующей среде виртуализации.

Техническая поддержка не оказывается в случаях использования сторонних систем виртуализации.

Рекомендуем использовать физические сервера, либо KVM виртуализацию.

Минимальная установка менеджера в архитектуре **Standalone** включает:

- Один хост для менеджера управления - Standalone-хост.
- Один хост-гипервизор (для обеспечения высокой доступности виртуальных машин рекомендуется добавить в среду как минимум два хоста).
- Внешняя система хранения данных (СХД) или локальное хранилище для размещения домена хранения данных (хранилища). Хранилище должно быть доступно всем хостам виртуализации.

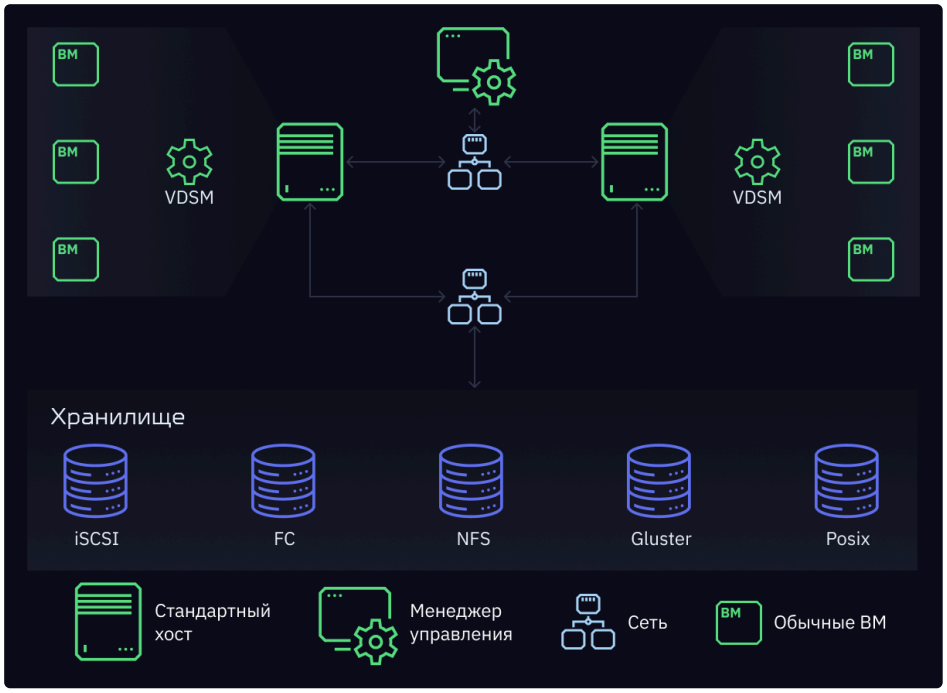

Рисунок 2. Архитектура виртуализации zVirt Standalone:

Существует также вариант развертывания **Standalone All-in-One**. В этой архитектуре **Standalone-хост** после развертывания на нем Менеджера управления, добавляется в среду как гипервизор. В самой минимальной конфигурации накопители **Standalone-хоста** также можно использовать для создания доменов хранения.

Несмотря на значительную экономию оборудования при такой архитектуре, настоятельно не рекомендуется использовать её для продуктивных сред, поскольку в этом случае **Standalone-хост** является единой точкой отказа для всей инсталляции.

Пояснения по планированию инфраструктуры

В этой главе описаны преимущества, ограничения и доступные опции для различных компонентов среды виртуализации zVirt.

1. Хосты

Все хосты в кластере должны иметь одинаковый тип процессора. Процессоры Intel и AMD не могут находится в одном кластере.

2. Центр данных

Каждый центр данных должен иметь как минимум один домен хранения данных. Также центр данных может иметь не более одного домена хранения **Экспорт**. Домены типа **Экспорт** и **ISO** устарели, но при необходимости их можно создать.

Домен хранения может состоять либо из блочных устройств (iSCSI или Fibre Channel), либо из файловой системы (POSIX).

По умолчанию домены GlusterFS и локальные домены хранения поддерживают размер блока 4K. Размер блока 4K может обеспечить более высокую производительность, особенно при использовании больших файлов, а также необходим при использовании инструментов, требующих совместимости с 4K, таких как VDO.

В настоящее время zVirt не поддерживает блочное хранилище с размером блока 4K. Вы должны настроить блочное хранилище в режиме (512b block).

Типы хранилищ, описанные в следующих разделах, поддерживаются для использования в качестве доменов хранения данных. Домены хранения **Экспорт** и **ISO** поддерживают только файловые типы хранения. Домен **ISO** поддерживает локальное хранение при использовании в локальном центре данных.

2.1. NFS

zVirt поддерживает NFS версий 3 и 4. Для производственных рабочих нагрузок требуется сервер NFS корпоративного уровня, если NFS не используется только в качестве домена хранения **ISO**. Когда корпоративная NFS развернута на 10GbE, разделена с помощью VLAN,

а отдельные службы настроены на использование определенных портов, она является одновременно быстрой и безопасной.

При расширении NFS хранилища zVirt сразу же распознает изменение размера хранилища данных. Никакой дополнительной настройки на хостах или менеджере управления не требуется. Это дает NFS небольшое преимущество перед блочным хранилищем с точки зрения масштабирования и эксплуатации.

2.2. iSCSI

Для производственных рабочих нагрузок требуется сервер iSCSI корпоративного уровня. Если корпоративный iSCSI развернут на 10GbE, разделен на виртуальные локальные сети и использует аутентификацию CHAP, он является одновременно быстрым и безопасным. iSCSI также может использовать многоканальность (multipathing) для повышения высокой доступности.

zVirt поддерживает 1800 логических томов на блочный домен хранения. Разрешается использовать не более 300 LUN.

2.3. Fibre Channel

Fibre Channel является одновременно быстрым и безопасным, и его следует использовать, если он уже используется в целевом центре данных. Его преимущество заключается в низкой нагрузке на процессор по сравнению с iSCSI и NFS. Fibre Channel также может использовать многоканальность (multipathing) для повышения высокой доступности.

zVirt поддерживает 1500 логических томов на блочный домен хранения. Разрешается использовать не более 300 LUN.

2.4. Fibre Channel over Ethernet

Чтобы использовать Fibre Channel over Ethernet (FCoE) необходимо включить ключ `fcoe` в менеджере управления и установить пакет `vdsm-hook-fcoe` на хосты.

zVirt поддерживает 1500 логических томов на блочный домен хранения. Разрешается использовать не более 300 LUN.

2.5. Gluster Storage

Gluster Storage (GS) - это POSIX-совместимая файловая система с открытым исходным кодом. Три или более серверов конфигурируются как кластер Gluster Storage, вместо сетевых устройств хранения данных (NAS) или массива сети хранения данных (SAN).

Gluster Storage следует использовать по 10GbE и разделять с помощью виртуальных локальных сетей.

2.6. Гиперконвергентная инфраструктура

zVirt поддерживает гиперконвергентный вариант развертывания с помощью Gluster. Вместо того чтобы подключать zVirt к внешнему хранилищу Gluster, существует возможность объединить zVirt и Gluster в одной инфраструктуре, что позволяет снизить эксплуатационные расходы и накладные расходы.

2.7. POSIX-совместимые файловые системы

Другие POSIX-совместимые файловые системы могут использоваться в качестве доменов хранения в zVirt, если они являются кластерными файловыми системами, такими как Global File System 2 (GFS2), и поддерживают разреженные файлы и прямой ввод-вывод. Файловая система Common Internet File System (CIFS), например, не поддерживает прямой ввод/вывод, что делает ее несовместимой с zVirt.

2.8. Локальное хранилище

Локальное хранилище создается на отдельном хосте, используя собственные ресурсы хоста. Когда вы настраиваете хост на использование локального хранилища, он автоматически добавляется в новый локальный центр данных и кластер, в который не могут быть добавлены другие хосты. Виртуальные машины, созданные в кластере с одним хостом, не могут быть перемещены, ограждены или запланированы.

Для хостов локальное хранилище всегда должно быть определено в файловой системе, отдельной от `/` (root). Используйте отдельный логический том или диск.

3. Пояснения по сети

При планировании и настройке сетей в среде zVirt настоятельно рекомендуется ознакомиться с концепциями сетей и их использованием. Прочитайте руководства производителя сетевого оборудования для получения дополнительной информации об управлении сетями.

Логические сети могут поддерживаться с помощью физических устройств, таких как сетевые карты, или логических устройств, таких как `bond`. Bonding улучшает высокую доступность и обеспечивает повышенную отказоустойчивость, поскольку все объединенные сетевые карты должны выйти из строя, чтобы сам `bond` вышел из строя. Режимы объединении 1, 2, 3 и 4 могут использоваться для сетей виртуальных машин. Режимы 0, 5 и

6 не предназначены для сети виртуальных машин. Система виртуализации по умолчанию использует режим 4.

Нет необходимости иметь одно устройство для каждой логической сети, поскольку несколько логических сетей могут совместно использовать одно устройство с помощью тегов виртуальных локальных сетей (VLAN) для изоляции сетевого трафика. Чтобы использовать эту функцию, тегирование VLAN должно поддерживаться на уровне коммутатора.

Ограничения на количество логических сетей в zVirt:

- Количество логических сетей, подключенных к хосту, ограничено количеством доступных сетевых устройств в сочетании с максимальным количеством виртуальных локальных сетей (VLAN), которое составляет 4096.
- Количество сетей, которые могут быть присоединены к хосту за одну операцию, в настоящее время ограничено 50.
- Количество логических сетей в кластере ограничено количеством логических сетей, которые могут быть присоединены к хосту, поскольку сетевое взаимодействие должно быть одинаковым для всех хостов в кластере.
- Количество логических сетей в центре данных ограничено только количеством содержащихся в нем кластеров в сочетании с количеством логических сетей, разрешенных для каждого кластера.

Будьте особенно внимательны при изменении свойств сети управления **ovirtmgmt**. Неправильные изменения свойств сети **ovirtmgmt** могут привести к тому, что хосты станут недоступными.

4. Поддержка серверов каталога

Во время установки менеджер zVirt по умолчанию создает пользователя **admin** в домене **internal**. Эта учетная запись предназначена для использования при первоначальной настройке среды и для устранения неполадок. Вы можете создать дополнительных пользователей во внутреннем домене с помощью утилиты ovirt-aaa-jdbc-tool. Учетные записи пользователей, созданные в локальных доменах, называются локальными пользователями.

Вы также можете подключить внешний сервер каталогов к системе виртуализации и использовать его в качестве внешнего домена. Учетные записи пользователей, созданные во внешних доменах, называются пользователями каталога.

Поддерживается использование более одного сервера каталогов.

Для использования с zVirt поддерживаются следующие серверы каталогов:

- Active Directory

- Identity Management (IdM - на основе IPA)
- Red Hat Directory Server 9 (RHDS 9 - основан на 389DS)
- OpenLDAP
- IBM Security (Tivoli) Directory Server

Пользователь с правами на чтение всех пользователей и групп должен быть создан в сервере каталогов специально для использования в качестве сервисной учётной записи для менеджера управления zVirt. Не используйте пользователя с правами администратора на сервере каталогов в качестве сервисной учётной записи для менеджера управления zVirt.

5. Инфраструктурные пояснения

5.1. Локальный или удаленный хостинг

Следующие компоненты могут быть размещены как на менеджере управления, так и на удаленном хосте. Размещение всех компонентов на менеджере управления проще и требует меньше ресурсов для обслуживания. Перемещение компонентов на удаленный хост требует больших ресурсов, но может повысить производительность как менеджера управления, так и `Data Warehouse`.

База данных и служба Data Warehouse:

- Чтобы разместить Data Warehouse на менеджере управления, выберите `Yes`, когда появится соответствующий запрос от утилиты `engine-setup`.
- Чтобы разместить Data Warehouse на удаленной машине, выберите `No`, когда появится соответствующий запрос от утилиты `engine-setup`.

Вы также можете разместить службу Data Warehouse и базу данных отдельно друг от друга.

Websocket proxy:

- Чтобы разместить прокси-сервер на менеджере управления, выберите `Yes`, когда появится соответствующий запрос от утилиты `engine-setup`.

5.2. Только удаленный хостинг

Следующие компоненты должны быть размещены на удаленной машине:

DNS

Использование службы DNS внутри ВМ в той же среде виртуализации не рекомендуется.

Хранилище

За исключением локального хранилища, служба хранения не должна находиться на одной машине с менеджером управления или любым хостом.

Управление идентификацией

IdM (ipa-server) несовместим с пакетом `mod_ssl`, который требуется для менеджера управления.

6. Моментальные снимки

Моментальные снимки — это функция, которая позволяет администратору создавать точки восстановления операционной системы, приложений и данных виртуальной машины на определенный момент времени.

Моментальные снимки сохраняют данные, присутствующие в образе жесткого диска виртуальной машины, в виде тома COW и позволяют восстановить данные, существовавшие на момент создания моментального снимка.

При создании моментального снимка, происходит добавление новой дельта-копии диска виртуальной машины (слоя `COW` поверх текущего), после чего все данные будут записываться в эту дельту (новый слой `COW`). Таким образом, чем больше данных записывается после создания моментального снимка, тем больше времени потребуется для фиксации и консолидации их обратно в родительский образ, поэтому не рекомендуется использовать в продуктивной среде моментальные снимки, особенно в высоконагруженных виртуальных машинах.

Важно понимать, что образ жесткого диска виртуальной машины представляет собой цепочку из одного или нескольких слоёв. С точки зрения виртуальной машины эти слои выглядят как один образ диска. Виртуальная машина работает только с диском и не имеет доступа к слоям, из которых состоит диск.

Каждый моментальный снимок создается для того, чтобы администратор мог отменить изменения в виртуальной машине, внесенные после создания моментального снимка. Снимки обеспечивают функциональность, аналогичную функции **точки восстановления**.

Моментальные снимки - это не резервные копии, а фиксация состояния образа виртуальной машины в определенный момент времени, к которому можно вернуться при необходимости. Не полагайтесь на моментальные снимки как на полноценный функционал резервного копирования.

Не храните продолжительное время моментальные снимки виртуальных машины. Как только убедитесь, что возврат к состоянию на момент создания моментального снимка больше не требуется - удалите моментальные снимки.

Ограничьте количество моментальных снимков. Создание нескольких снимков подряд может снизить производительность виртуальной машины и хоста, так как `qemu` придется

просматривать каждый образ в цепочке моментальных снимков, чтобы считать новый файл из базового образа (base_image).

Перед созданием моментальных снимков на виртуальной машине должны быть установлены гостевые дополнения.

Три основные операции со снимками:

- Создание, которое включает в себя первый снимок, созданный для виртуальной машины.
- Предварительный просмотр, который включает предварительный просмотр моментального снимка, чтобы определить, следует ли восстанавливать данные на момент времени, когда был сделан снимок.
- Удаление, которое включает удаление точки восстановления, которая больше не требуется.

Моментальные снимки дисков виртуальных машин, помеченных как `общие`, и те, которые основаны на `прямом подключении LUN`, не поддерживаются. Моментальный снимок любой другой виртуальной машины, которая не клонируется или не мигрирует, может быть сделан во время работы, приостановки или после остановки.

Рекомендации по планированию инфраструктуры

1. Общие рекомендации

- Сразу после завершения развертывания создайте полную резервную копию и сохраните ее в отдельном месте. После этого регулярно создавайте резервные копии.
- Избегайте запуска любой службы, от которой зависит менеджер управления, в качестве виртуальной машины в той же среде виртуализации. Если так делается, необходимо тщательно спланировать это, чтобы минимизировать время простоя, если виртуальная машина, содержащая эту службу, выйдет из строя.
- Убедитесь, что хост или виртуальная машина, на котором будет производится развертывание менеджера управления, имеет достаточную энтропию. Значения ниже `200` могут привести к сбою при развертывании менеджера. Чтобы проверить значение энтропии, выполните команду `cat /proc/sys/kernel/random/entropy_avail`. Чтобы увеличить энтропию, установите пакет `rng-tools`.
- Вы можете автоматизировать развертывание хостов и виртуальных машин с помощью `PXE`, `Kickstart`, `CloudForms`, `Ansible` или их комбинации. Обратите внимание, что установка в архитектуре **Hosted engine** с помощью `PXE` не поддерживается.
- Используйте протокол сетевого времени (NTP) на всех хостах и виртуальных машинах в среде для синхронизации времени. Аутентификация и сертификаты особенно чувствительны к разнице во времени.

  В zVirt поддерживается только сервер `chrony`.
- Документируйте всё, чтобы все, кто работает с окружением, знали о его текущем состоянии и необходимых процессах.

2. Рекомендации по безопасности

- Не отключайте функции безопасности (такие как HTTPS, SELinux и межсетевой экран) на хостах или виртуальных машинах.
- Создайте индивидуальные учетные записи администраторов, вместо того чтобы допускать использование одной учетной записи администратора несколькими сотрудниками. Это также полезно для отслеживания действий.
- Ограничьте доступ к хостам и создайте отдельные учетные записи. Не используйте одну учётную запись с правами `root` на всех хостах виртуализации.

- На хостах виртуализации должны использоваться только пакеты и службы, необходимые для виртуализации, производительности, безопасности и мониторинга. Хосты не должны иметь дополнительных пакетов, таких как анализаторы, компиляторы или других компонентов, которые добавляют риск для безопасности и стабильности.

3. Рекомендации по работе с сетью

- Объединяйте сетевые интерфейсы, особенно на продуктивных хостах. Объединение улучшает общую доступность, а также пропускную способность сети.
- Стабильная сетевая инфраструктура использующая DNS и DHCP.
- Если объединения сетевых интерфейсов (bonding) будут использоваться совместно с другим сетевым трафиком, необходимо обеспечить надлежащее качество обслуживания (QoS) для хранилища и другого сетевого трафика.
- Для оптимальной производительности и упрощения поиска и устранения неисправностей используйте виртуальные локальные сети для разделения различных типов трафика и оптимального использования сетей 10 GbE или 40 GbE.
- Если базовые коммутаторы поддерживают `jumbo frames`, установите `MTU` на максимальный размер (например, 9000), который поддерживают базовые коммутаторы. Эта настройка обеспечивает оптимальную пропускную способность, более высокую пропускную способность и меньшее использование ЦП для большинства приложений. `MTU` по умолчанию определяется минимальным размером, поддерживаемым базовыми коммутаторами. Если у вас включен `LLDP`, вы можете увидеть `MTU`, поддерживаемый хостом, в подсказках сетевой карты в окне `Установка сетей хоста`.

  

  Если вы измените параметры `MTU` сети, вы должны распространить эти изменения на работающие виртуальные машины в сети: "Горячее" отключение и повторное подключение **vNIC** каждой виртуальной машины, которая должна применить настройки **MTU**, или перезапуск виртуальных машин. В противном случае эти интерфейсы выйдут из строя при миграции виртуальной машины на другой хост.

- Сети 1 GbE следует использовать только для трафика управления. Используйте 10 GbE или 40 GbE для виртуальных машин и хранилищ на базе Ethernet.
- Если на хост добавляются дополнительные физические интерфейсы для использования хранилища, снимите флажок `Сеть BM`, чтобы VLAN назначалась непосредственно физическому интерфейсу.

4. Рекомендации по хостам

- Стандартизируйте хосты в одном кластере. Это включает в себя использование одинаковых моделей оборудования и версий микропрограммного обеспечения.

Смешивание различного серверного оборудования в одном кластере может привести к нестабильной производительности от хоста к хосту.

- Настройте устройства ограждения во время развертывания. Устройства ограждения необходимы для обеспечения высокой доступности.
- Используйте отдельные аппаратные коммутаторы для ограждения трафика. Если мониторинг и ограждение проходят через один коммутатор, этот коммутатор становится единой точкой отказа для обеспечения высокой доступности.

4.1. Рекомендации по настройке сетей хоста

- Всегда используйте менеджер управления для изменения сетевой конфигурации хостов в кластерах. В противном случае вы можете создать неподдерживаемую конфигурацию.
- Если ваша сетевая инфраструктура сложная, вам может потребоваться настроить сеть хоста вручную перед добавлением хоста в среду виртуализации.
- Настроить сеть можно с помощью `Cockpit`. В качестве альтернативы можно использовать `nmtui` или `nmcli`.
- Если сеть не требуется для развертывания архитектуры **Hosted Engine** или для добавления хоста в менеджер управления, настройте сеть на **Портале администрирования после добавления** хоста в менеджер управления.
- Используйте следующие соглашения об именовании:
  - Устройства VLAN: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
  - Интерфейсы VLAN: physical_device.VLAN_ID (например, eth0.23, eth1.128, enp3s0.50).
  - Интерфейсы bond: bondnumber (например, bond0, bond1).
  - VLAN на объединенных интерфейсах: bondnumber.VLAN_ID (например, bond0.50, bond1.128).
- Используйте объединение сетей (bonding). `Teaming` не поддерживается в zVirt и приведет к ошибкам.
- Используйте рекомендуемые режимы объединения:
  - `Режим 0` (round-robin) - передача пакетов через сетевые интерфейсы в последовательном порядке. Пакеты передаются в цикле, который начинается с первого доступного сетевого интерфейса на хосте и заканчивается последним доступным сетевым интерфейсом на хосте. Все последующие циклы начинаются с первой доступной карты сетевого интерфейса. `Режим 0` обеспечивает отказоустойчивость и распределяет нагрузку между всеми сетевыми интерфейсными картами в связке. Обратите внимание, что `режим 0` не может использоваться в сочетании с `bridge` и поэтому не совместим с логическими сетями виртуальных машин.

    - `Режим 1` (active-backup) - переводит все сетевые интерфейсы в резервное состояние, в то время как один сетевой интерфейс остается активным. В случае отказа активного сетевого интерфейса один из резервных интерфейсов заменяет сбойный интерфейс в качестве единственного активного сетевого интерфейса в бонде. MAC-адрес соединения в `режиме 1` виден только на одном порту, чтобы предотвратить путаницу, которая может возникнуть, если MAC-адрес соединения изменится на MAC-адрес активной сетевой интерфейсной карты. `Режим 1` обеспечивает отказоустойчивость и поддерживается в zVirt.
    - `Режим 2` (XOR) - выбирает сетевой интерфейс, через который будут передаваться пакеты, на основе результата операции XOR над MAC-адресами источника и назначения по модулю количества сетевых интерфейсов. Этот расчет гарантирует, что для каждого используемого MAC-адреса назначения будет выбрана одна и та же карта сетевого интерфейса. `Режим 2` обеспечивает отказоустойчивость и балансировку нагрузки и поддерживается в zVirt.
    - `Режим 3` (broadcast) - передает все пакеты всем сетевым интерфейсам. `Режим 3` обеспечивает отказоустойчивость и поддерживается в zVirt.
    - `Режим 4` (IEEE 802.3ad) - создает группы агрегации, в которых интерфейсы имеют одинаковые настройки скорости и дуплекса. `Режим 4` использует все сетевые интерфейсы в активной группе агрегации в соответствии со спецификацией `IEEE 802.3ad` и поддерживается в zVirt.
    - `Режим 5` (adaptive transmit load balancing) - обеспечивает распределение исходящего трафика с учетом нагрузки на каждый сетевой интерфейс в связке и то, что текущий сетевой интерфейс получает весь входящий трафик. Если сетевой интерфейс, назначенный для приема трафика, выходит из строя, роль приема входящего трафика возлагается на другой сетевой интерфейс. `Режим 5` нельзя использовать в сочетании с bridge, поэтому он не совместим с логическими сетями виртуальных машин.
    - `Режим 6` (adaptive load balancing) - объединяет `режим 5` (adaptive transmit load balancing) с балансировкой нагрузки при приеме для трафика IPv4 без каких-либо специальных требований к коммутатору. Для балансировки принимаемой нагрузки используется согласование ARP. `Режим 6` нельзя использовать в сочетании с `bridge`, поэтому он не совместим с логическими сетями виртуальных машин.
- Если сеть `ovirtmgmt` не используется виртуальными машинами, сеть может использовать любой поддерживаемый режим объединения.
- Если сеть `ovirtmgmt` используется виртуальными машинами, сеть должна использовать режимы объединения `1`, `2`, `3` или `4`.
- По умолчанию в zVirt используется `режим объединения 4 Dynamic Link Aggregation`. Если ваш коммутатор не поддерживает протокол `Link Aggregation Control Protocol (LACP)`, используйте `режим 1 Active-Backup`.

Пример 1. Настройка VLAN на физической сетевой карте (в примере используется `nmcli`, но вы можете использовать любой инструмент)

```
nmcli connection add type vlan con-name vlan50 ifname eth0.50 dev eth0 id 50
nmcli con mod vlan50 +ipv4.dns 8.8.8.8 +ipv4.addresses 123.123.0.1/24
+ivp4.gateway 123.123.0.254
```

Пример 2. Настройка VLAN поверх объединения (в примере используется `nmcli`, но вы можете использовать любой инструмент)

```
nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=active-backup,miimon=100" ipv4.method disabled ipv6.method ignore
nmcli connection add type ethernet con-name eth0 ifname eth0 master bond0
slave-type bond
nmcli connection add type ethernet con-name eth1 ifname eth1 master bond0
slave-type bond
nmcli connection add type vlan con-name vlan50 ifname bond0.50 dev bond0 id
50
nmcli con mod vlan50 +ipv4.dns 8.8.8.8 +ipv4.addresses 123.123.0.1/24
+ivp4.gateway 123.123.0.254
```

Не отключайте сервис `firewalld` (межсетевой экран).

5. Рекомендации по развертыванию архитектуры Hosted Engine

Создайте отдельный центр данных и кластер для BM HostedEngine и других служб уровня инфраструктуры, если ваша инфраструктура может позволить это. Хотя виртуальная машина с менеджером управления может работать на хостах в обычном кластере, отделение от остальных виртуальных машин помогает упростить план резервного копирования и управление производительностью, доступностью и безопасностью.

Домен хранения, предназначенный для BM HostedEngine, создается во время развертывания. Не используйте этот домен хранения для других виртуальных машин.

Если ожидается большая нагрузка на хранилище, разделите сети миграции, управления и хранения, чтобы уменьшить влияние на работоспособность BM HostedEngine.

Все хосты способные поддерживать работу BM HostedEngine должны иметь одинаковое семейство процессоров, чтобы виртуальная машина могла безопасно мигрировать между ними. Если вы планируете создание кластера с хостами, имеющими различные семейства процессоров, необходимо начинать установку с самого раннего семейства.

Если ВМ HostedEngine выключается или нуждается в миграции, на хосте должно быть достаточно памяти, чтобы виртуальная машина могла перезапуститься или мигрировать на него.

6. Рекомендации по переподписке

6.1. Переподписка vCPU (Overcommitting vCPU)

Гипервизор KVM поддерживает переподписку vCPU. Переподписка vCPU подразумевает назначение виртуальным машинам большего количества виртуальных процессоров, чем доступно физически на хосте.

Цель переподписки vCPU состоит в оптимизации использования ресурсов, поскольку большинство процессов в виртуальных машинах редко нуждаются в постоянном использовании всех выделенных ресурсов. Если каждая виртуальная машина использует ресурсы частично, а потребность в ресурсах меняется со временем, то можно безопасно перераспределить доступные физические ядра между большим числом виртуальных машин.

Чрезмерная переподписка может негативно сказываться на производительности. Если работающие виртуальные машины одновременно будут использовать все выделенные ресурсы, возникнет конкуренция за физические ядра, что приведет к задержкам и снижению производительности. Поэтому перед применением переподписки стоит тщательно оценить рабочую нагрузку и учесть возможные риски.

Если выделить одной виртуальной машине большее количество ядер, чем доступно на физическом процессоре, это также приведет к снижению производительности, поскольку приложения в гостевой ОС получают меньше процессорного времени, чем требуется. Для повышения производительности рекомендуется назначать каждой ВМ количество vCPU, достаточное для выполнения программ внутри гостевой ОС этой ВМ.

При использовании виртуальных машин с поддержкой симметричной многопроцессорной обработки (SMP) возникает дополнительная нагрузка на процессор. Переподписка CPU увеличивает эту нагрузку, так как использование тайм-слотов для распределения ресурсов между гостевыми ОС замедляет межпроцессорное взаимодействие внутри каждой гостевой ОС. Эта дополнительная нагрузка возрастает с увеличением числа vCPU у гостевых машин или повышением коэффициента переподписки.

Переподписка vCPU будет более эффективной, когда на одном хосте размещать несколько ВМ с небольшим количеством vCPU на каждую, по сравнению с общим числом физических ядер хоста. Гипервизор KVM способен безопасно поддерживать работу гостевых машин с нагрузкой менее 100% при соотношении 5 vCPU к одному физическому ядру на одном хосте (коэффициент переподписки 5:1). Гипервизор будет равномерно распределять нагрузку, переключаясь между виртуальными машинами.

Не стоит размещать больше 10 vCPU на каждое физическое ядро (коэффициент переподписки 10:1).

В средах с переподпиской снижается стабильность приложений, которые используют 100% доступной памяти или ресурсов процессора. Не используйте переподписку памяти или vCPU в продуктивной среде без проведения тестирования, так как коэффициент переподписки vCPU и количество SMP зависят от типа рабочей нагрузки.

6.2. Переподписка памяти (Overcommitting memory)

Виртуальные машины, работающие на гипервизоре KVM, не имеют выделенных блоков физической оперативной памяти. Вместо этого каждая гостевая виртуальная машина функционирует как процесс Linux, где ядро Linux хоста выделяет память только по запросу. Менеджер памяти хоста перемещает память виртуальной машины между собственной физической памятью и пространством подкачки.

Для использования переподписки памяти стоит выделять достаточное пространство подкачки на хосте для размещения всех виртуальных машин и достаточный объем памяти для процессов хоста. Операционная система хоста требует минимум 4 ГБ памяти плюс 4 ГБ пространства подкачки.

Переподписка не работает со всеми виртуальными машинами, но может быть эффективна при запуске нескольких идентичных ВМ с KSM.

Переподписка не является решением для общих проблем с памятью. Чтобы избежать недостатка памяти, рассмотрите возможность выделения меньшего объема памяти для каждой ВМ, добавления дополнительного физического объема памяти на хост или использования пространства подкачки.

Виртуальная машина будет работать медленнее при частом использовании подкачки. Помимо этого, переподписка может привести к исчерпанию памяти (OOM), что вызовет остановку важных системных процессов ядром Linux. Перед использованием переподписки стоит провести тестирование.

Переподписка памяти не поддерживается при назначении устройств хоста виртуальной машине. Это связано с тем, что при назначении устройств вся память виртуальной машины должна быть предварительно статически распределена, чтобы обеспечить прямой доступ к памяти (DMA) с назначенным устройством.