Российская сервисная платформа виртуализации РУСТЭК

Руководство администратора

Релиз 2021.2.6

2023

Оглавление

1. Порядок создания объектов

Все объекты по умолчанию создаются в зоне видимости *Проекта* и принадлежат ему. *Администратор платформы* может создавать *Проекты* и *Пользователей*, а также присваивать им роли Администраторов в созданных *Проектах*. Также Администратор может отметить *Образы*, *Конфигурации* и *Сети* как "Общие", что сделает их видимыми на уровне всей платформы.

1.1. Начало работы

Порядок создания объектов во многом зависит от задачи, которая стоит перед пользователем. Рассмотрим самый простой сценарий: создание новой ВМ с доступом к интернету и возможностью подключения извне по `ssh`.

1.2. Создание ВМ

Первым делом выполните вход в портал. Учётная запись администратора по умолчанию — `admin`, её автоматически сгенерированный пароль хранится в файле `/etc/openstack/clouds.yml` на любом из узлов инсталляции. После авторизации и ввода валидной лицензии можно начинать создавать необходимые объекты.

1.2.1. Создание конфигурации

Конфигурация — это шаблон, по которому виртуальной машине будут предоставлены ресурсы. Она включает в себя такие параметры, как количество vCPU и объём оперативной памяти, которыми будет оснащён любой новый ВМ с этой конфигурацией.

Подробнее: в разделе 3. .

1.2.2. Создание образа

Образ виртуальной машины — это заранее подготовленный файл, содержащий в себе преднастроенную операционную систему.

Подробнее: в разделе 7. .

1.2.3. Создание сети

1.2.3.1. Сеть

Для коммуникации с внешним миром нужно создать сеть — в ПВ РУСТЭК под этим подразумевается сеть второго уровня модели OSI, обеспечивающая физическую связность. Существует два типа сетей: внутренние и внешние. Для пользователя они отличаются состоянием чекбокса «Внешняя», который доступен при создании сети. В нашем сценарии нужно создать сети обоих типов.

Подробнее: в разделе 6.4.

1.2.3.2. Подсеть

Подсеть — это пул IP-адресов и сопутствующих настроек, которые обеспечивают ВМ возможность взаимодействия с другими сетевыми устройствами на третьем уровне OSI. По умолчанию при создании подсети выбрана опция DHCP, которая позволит виртуальным машинам получать IP-адреса из доступного диапазона автоматически. Подсеть нужно создать для обеих созданных сетей: внутренней и внешней.

Подробнее: в разделе 6.5.

1.2.3.3. Роутер

Для того чтобы пакеты могли проходить из внутренних сетей во внешний мир и наоборот, нужно создать маршрутизатор — роутер, и подключить к нему созданные сети.

Подробнее: в разделе 6.7.

1.2.4. Создание ВМ

Основные необходимые объекты созданы, и теперь можно приступать непосредственно к созданию виртуальной машины. При создании потребуется указать конфигурацию, образ и сеть, а после того, как ВМ создастся и запустится, настроить к ней доступ.

Подробнее: в документе 2.6_Руководство пользователя.

1.2.5. Настройка доступа

1.2.5.1. Плавающий IP-адрес

Посредством плавающих IP-адресов в ПВ РУСТЭК реализован NAT, связывающий сети внешнего мира с внутренними виртуальными сетями. В нашем примере это необходимо для того, чтобы обеспечить доступ до операционной системы ВМ, связав адрес из внешней подсети, доступный извне, с адресом во внутренней.

Подробнее: в разделе 6.9.

1.2.5.2. Профили безопасности

Профили безопасности — это шаблоны настроек `iptables`, обеспечивающие фильтрацию трафика. Чтобы получить доступ к ВМ по `ssh`, нужно создать профиль безопасности с правилом, разрешающим порт `TCP 22`.

Подробнее: в разделе 6.8.

1.2.6. Готово!

Теперь можно проверить доступ к вашему первому ВМ, подключившись по `ssh` на плавающий IP-адрес, ассоциированный с внутренним адресом ВМ.

2. Агрегаты

Создание и управление логическими группами узлов

Для повышения производительности или в целях администрирования существует возможность разбить развертывание физических ресурсов на логические группы. ПВ РУСТЭК предоставляет следующие механизмы для разделения логических групп:

Агрегат узлов — это группировка физических узлов в логическую единицу на основе таких атрибутов, как характеристики оборудования или производительности. Физический узел можно назначить одному или нескольким агрегатам узлов. Различные Конфигурации (flavor) ВМ и/или Образы ВМ можно сопоставить с агрегатами узла - для этого необходимо назначать пары параметров ключ-значение для групп узлов, имеющих статус "физический узел". Каждый агрегат может иметь несколько пар ключ-значение. Одна и та же пара ключ-значение может быть назначена нескольким агрегатам. Эта информация может использоваться в планировщике для включения расширенного планирования, для настройки пулов ресурсов гипервизора или для определения логических групп для миграции.

Так же можно настроить множители веса для каждого агрегата, задав параметр `xxx_weight_multiplier` конфигурации в метаданных агрегата. Агрегаты можно использовать для управления балансировкой нагрузки, обеспечения физической изоляции или избыточности, группировки ВМ с общими атрибутами или отдельных классов оборудования. При создании агрегата узлов можно указать имя зоны. Это имя представляется пользователям облака как зона доступности, которую они могут в дальнейшем использовать.

Зона доступности — это облачное пользовательское представление агрегата узлов. Пользователь не может просматривать физические узлы в зоне доступности или просматривать метаданные зоны доступности. Пользователь может видеть только название зоны доступности. Каждый физический узел можно назначить только одной зоне доступности. Если при создании ВМ пользователь не укажет зону доступности, в которой будут создаваться ВМ, то её можно настроить по умолчанию.

Возможность управления агрегатами физических узлов ограничивается RBAC-политиками вида **os_compute_api:os-aggregates-*** служб Nova в файле /etc/nova/policy.yaml

2.1. Планирование агрегатов узлов

За включение механизма планировщика ВМ на агрегатах узлов с определенными атрибутами отвечает фильтр `AggregateInstanceExtraSpecsFilter` в конфиге `/etc/nova/nova.conf` раздела `[filter_scheduler]` параметра `enabled_filters`:

```
[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
enabled_filters                                                    =
AvailabilityZoneFilter,ComputeFilter,ImagePropertiesFilter,ServerGroupAntiAffinityF
ilter,ServerGroupAffinityFilter,AggregateInstanceExtraSpecsFilter
```

Чтобы этот фильтр работал должным образом, нужно создать дополнительные метаданные у Конфигурации, указав префикс с `aggregate_instance_extra_specs:`имя. Т.е. фильтр сопоставит метаданные Конфигурации вида:

`aggregate_instance_extra_specs:cpu_info:vendor=Intel`

с метаданными агрегата вида:

`cpu_info:vendor=Intel`.

За включение механизма обработки запросов зоны доступности отвечает фильтр `AvailabilityZoneFilter`.

2.2. Создание, редактирование и удаление агрегата узлов

Администратор платформы ПВ РУСТЭК может создать необходимое количество агрегатов. Агрегаты создаются как через портал в подразделе **Агрегаты** раздела **Конфигурация**, так и через CLI.

Для создания агрегата в портале нужно:

- на панели инструментов нажать кнопку ;
- заполнить открывшуюся форму **Создание агрегата**;
- нажать кнопку **Создать**.

Для добавления физических узлов в агрегат в портале нужно:

- выбрать агрегат;
- на панели инструментов нажать кнопку ;
- добавить физические узлы в открывшейся форме **Редактирование агрегата**;
- нажать кнопку **Сохранить**.

Удаление агрегата в портале производится аналогично с помощью кнопки .

Невозможно удалить агрегат, к которому привязаны узлы. Сначала следует удалить узлы из привязки к агрегату, а затем удалить агрегат.

Для выполнения действий над агрегатом в CLI используют следующие команды:

```
# openstack aggregate create <aggregate_name>
# openstack aggregate set \
 --property <key=value> \
 --property <key=value> \
 <aggregate_name>
# openstack aggregate add host \
 <aggregate_name> \
 <compute_host_name>
# openstack aggregate remove host \
 <aggregate_name> \
 <compute_host_name>
# openstack aggregate delete <aggregate_name>
```

2.3. Пример создания агрегата узлов с процессорами AMD

Для примера создадим агрегат в CLI с именем `amd-cpu-agg`, назначим ему метаданные `amd_cpu=true` и добавим в этот агрегат два узла — node1 и node2, которые используют CPU AMD:

```
# openstack aggregate create amd-cpu-agg
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | nova                       |
| created_at        | 2021-12-22T07:31:13.013466 |
| deleted           | False                      |
| deleted_at        | None                       |
| id                | 1                          |
| name              | amd-cpu-agg                |
| updated_at        | None                       |
+-------------------+----------------------------+

# openstack aggregate set --property amd_cpu=true 1
+-------------------+----------------------------+
```

```
| Field             | Value                       |
+-------------------+-----------------------------+
| availability_zone | nova                        |
| created_at        | 2021-12-22T07:31:13.000000 |
| deleted           | False                       |
| deleted_at        | None                        |
| hosts             | []                          |
| id                | 1                           |
| name              | amd-cpu-agg                 |
| properties        | amd_cpu='true'              |
| updated_at        | None                        |
+-------------------+-----------------------------+

# openstack aggregate add host 1 node1
+-------------------+-------------------------------------------------------------+
| Field             | Value                                                       |
+-------------------+-------------------------------------------------------------+
| availability_zone | nova                                                        |
| created_at        | 2021-12-22T07:31:13.000000                                  |
| deleted           | False                                                       |
| deleted_at        | None                                                        |
| hosts             | [u'node1']                                                  |
| id                | 1                                                           |
| metadata          | {u'amd_cpu': u'true', u'availability_zone': u'nova'}        |
| name              | amd-cpu-agg                                                 |
| updated_at        | None                                                        |
+-------------------+-------------------------------------------------------------+

# openstack aggregate add host 1 node2
+-------------------+-------------------------------------------------------------+
| Field             | Value                                                       |
+-------------------+-------------------------------------------------------------+
| availability_zone | nova                                                        |
| created_at        | 2021-12-22T07:31:13.000000                                  |
| deleted           | False                                                       |
| deleted_at        | None                                                        |
| hosts             | [u'node1', u'node2']                                        |
| id                | 1                                                           |
| metadata          | {u'amd_cpu': u'true', u'availability_zone': u'nova'}        |
| name              | amd-cpu-agg                                                 |
| updated_at        | None                                                        |
+-------------------+-------------------------------------------------------------+
```

В результате будет создан агрегат `amd-cpu-agg` с ID 1, метаданными и двумя узлами.

Далее добавим метаданные `amd_cpu=true` области `aggregate_instance_extra_specs` требуемой конфигурации ВМ:

```
# openstack --os-cloud rustack_system flavor set \
    --property aggregate_instance_extra_specs:amd_cpu=true vtiny

# openstack flavor show vtiny
+----------------------------+-------------------------------------------------+
| Field                      | Value                                           |
+----------------------------+-------------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                           |
| OS-FLV-EXT-DATA:ephemeral  | 0                                               |
```

```
| access_project_ids         | None                                          |
| description                | None                                          |
| disk                       | 0                                             |
| id                         | 1496330f-4847-4a8b-8bbb-8351f83783f4          |
| name                       | vtiny                                         |
| os-flavor-access:is_public | True                                          |
| properties                 | aggregate_instance_extra_specs:amd_cpu='true' |
| ram                        | 128                                           |
| rxtx_factor                | 1.0                                           |
| swap                       |                                               |
| vcpus                      | 1                                             |
+----------------------------+-----------------------------------------------+
```

Теперь, при создании пользователем ВМ с конфигурацией из примера **vtiny,** планировщик будет использовать узлы node1 и node2.

2.4. Создание зон доступности

Зона доступности (AZ) — это видимая для пользователя логическая абстракция разделения узлов без знания физической инфраструктуры. Зоны доступности определяются путем добавления метаданных к агрегату. Добавление к агрегату делает агрегат видимым с точки зрения пользователя и позволяет планировать экземпляры ВМ для определенного набора узлов.

Зоны доступности могут использоваться, например, для логического разделения инфраструктуры: разбивка TOR-свитчей по разным зонам, разбивка физических узлов по разным стойкам или ИБП и т.д.

Однако, несмотря на их сходство, есть несколько дополнительных различий, которые следует учитывать при сравнении зон доступности и агрегатов узлов:

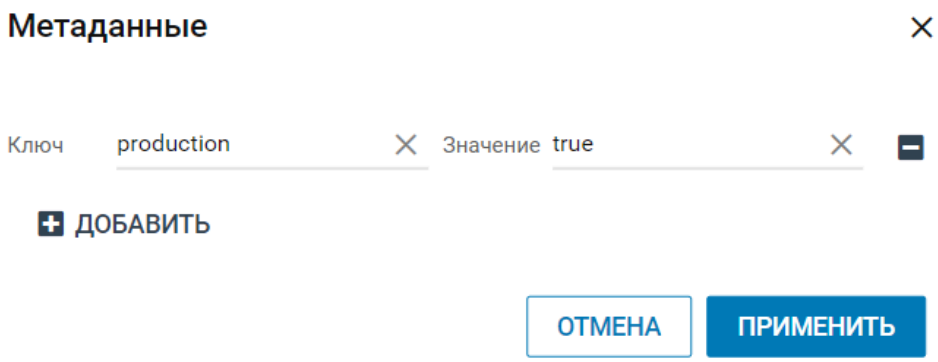
- Узел может быть частью нескольких агрегатов, но только в одной зоне доступности.
- Узел является частью **зоны доступности по умолчанию**, даже если он не принадлежит агрегату. Имя этой зоны доступности по умолчанию можно настроить с помощью `default_availability_zone` параметра конфигурации. В ПВ РУСТЭК зона доступности по умолчанию называется **nova**.

Имена зон доступности НЕ ДОЛЖНЫ содержать `:` (двоеточие), т.к. оно используются администраторами для указания узлов, на которых запускаются ВМ.

Создание зоны доступности выполняется путем связывания метаданных с агрегатом узлов. Это можно сделать в подразделе **Агрегаты** раздела **Конфигурация** портала, используя ключ метаданных **availability_zone** или в CLI.

Для связывания метаданных с агрегатом в портале нужно:

- выбрать агрегат;
- на панели инструментов нажать кнопку  ;
- ввести пару ключ-значение в открывшуюся форму **Метаданные**;
- нажать кнопку **Применить**.



Ввести пару ключ-значение можно также при создании агрегата.

Для связывания метаданных с агрегатом в CLI используют следующие команды:

```
    # openstack --os-cloud rustack_system aggregate create --zone az1 my-aggregate
    # openstack --os-cloud rustack_system aggregate add host my-aggregate my-host
    # openstack aggregate set \
        --property availability_zone=my-availability-zone2 my-aggregate2
    #   openstack   --os-cloud   rustack_system   aggregate   unset   --property
availability_zone my-aggregate
```

Хотя узел можно добавить к нескольким агрегатам узлов, его невозможно добавить в несколько зон доступности. Попытка добавить узел в несколько агрегатов узлов, связанных с разными зонами доступности, приведет к ошибке.

Посмотреть доступные зоны и создать ВМ, используя конкретную зону, можно с помощью команд:

```
    # openstack availability zone list --compute
    +-----------+-------------+
    | Zone Name | Zone Status |
    +-----------+-------------+
    | az1       | available   |
    | nova      | available   |
    +-----------+-------------+

    # openstack server create --volume disk1 --network net1 --flavor tiny --
availability-zone az1 test-server
```

Функционал выбора зоны доступности при создании ВМ доступен через API и CLI, но не доступен в портале.

Миграция между разными зонами доступности не поддерживается.

2.5. Создание изолированного агрегата для определенного проекта

В платформе можно создать агрегат, доступный только для определенных проектов. Только проекты, которые вы назначите агрегату, смогут запускать ВМ на узлах агрегата.

Изоляция по проектам использует службу Placement для фильтрации узлов агрегатов.

За включение механизма изоляции планировщика ВМ на агрегатах узлов с определенными атрибутами отвечают переменные **limit_tenants_to_placement_aggregate** и **placement_aggregate_required_for_tenants** в конфиге `/etc/nova/nova.conf` раздела `[scheduler]`:

- **limit_tenants_to_placement_aggregate** — включает механизм изоляции проектов по агрегатам;
- **placement_aggregate_required_for_tenants** — по умолчанию имеет значение False, что позволяет проектам, не назначенным какому-либо агрегату создавать ВМ на любых узлах любых агрегатов. Если агрегаты используются для ограничения некоторых проектов, но не всех - то значение должно быть False. Если все проекты должны быть ограничены с помощью агрегатов, то значение должно быть True, для того чтобы проекты не получали неограниченное создание ВМ на любом доступном узле.

Механизм изоляции использует метаданные ключ **filter_tenant_id<суффикс>** и значение **ID** требуемого проекта, назначенного на агрегаты. Суффиксом **<суффикс>** может служить любое значение, удобное администратору.

При изоляции агрегата для одного проекта, суффикс в ключе **filter_tenant_id** можно не использовать.

2.6. Пример изоляции агрегатов для определенных проектов

Настройка изоляции агрегатов доступна по API, CLI и из портала.

Настройка агрегатов в CLI:

```
# openstack --os-cloud rustack_system project list -f value
dd928b8021064a1583b26dc97b19f299 admin
0d7062ff91024ae799d8dd023ae69b45 user1
c9dce7ffc1c84c9bbb59755061eb8295 user2
8b4a40a9a22c47389a282103da4bda7e user3
...SKIP...

# openstack --os-cloud rustack_system aggregate set \
 --property filter_tenant_id0=c9dce7ffc1c84c9bbb59755061eb8295 \
 --property filter_tenant_id0d706=0d7062ff91024ae799d8dd023ae69b45 \
 --property filter_tenant_id=dd928b8021064a1583b26dc97b19f299 \
 my-aggregate1

# openstack --os-cloud rustack_system aggregate set \
 --property filter_tenant_id=8b4a40a9a22c47389a282103da4bda7e \
 my-aggregate2
```

3. Конфигурации

Конфигурация — это шаблон ресурсов, который определяет профиль виртуального оборудования для ВМ. При создании ВМ пользователь должен выбрать определенную конфигурацию, подходящую по параметрам для создаваемой ВМ.

Конфигурация может указывать количество следующих ресурсов, которое физический узел должен будет выделить для ВМ:

- Количество виртуальных ЦП;
- Количество оперативной памяти, в МБ.

Администратор платформы указывает проекты, которые могут использовать Конфигурацию, сделав ее общедоступной для всех проектов или частной для определенных проектов или доменов. Конфигурации могут использовать метаданные для указания необходимости аппаратной поддержки экземпляров ВМ, квот, QoS и т.д. Администратор платформы также может использовать метаданные, чтобы найти подходящие агрегаты узлов для размещения ВМ. Чтобы запланировать ВМ на агрегате узлов, нужно использовать специальный формат метаданных, указав префикс `extra_specs` ключа с `aggregate_instance_extra_specs:пространство имен`. Дополнительные сведения см. в разделе 2. .

> Возможность управления Конфигурациями ограничивается RBAC-политиками вида **os_compute_api:os-flavor-*** служб Nova в файле `/etc/nova/policy.yaml`

3.1. Создание конфигураций

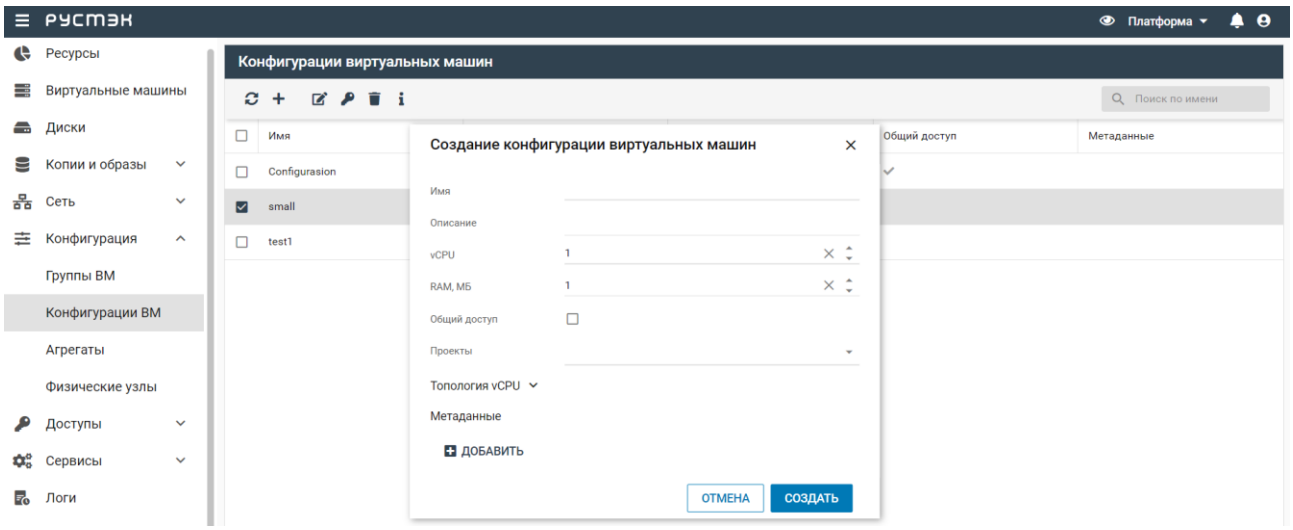
Вы можете создавать и управлять различными вариантами Конфигураций для достижения определенных функций или поведения, например:

- Указывать количество виртуальных ЦП;
- Указывать количество оперативной памяти и емкость основного диска по умолчанию в соответствии с потребностями ВМ;
- Добавлять метаданные, чтобы установить определенную скорость ввода-вывода для ВМ или ограничивать запуск ВМ на определенном агрегате узлов.

Администратор платформы ПВ РУСТЭК может создать необходимое количество Конфигураций. Конфигурации создаются как через портал, так и через CLI.

Конфигурация создается в подразделе **Конфигурации ВМ** раздела меню **Конфигурация** портала следующим образом:

- нажать кнопку **Создать** на панели инструментов;
- заполнить открывшуюся форму **Создание конфигурации виртуальных машин**;
- нажать кнопку **Создать**.



После создания публичной (с общим доступом) Конфигурации невозможно изменить ее настройки доступа.

В CLI Конфигурация создаётся с помощью команд:

```
# openstack --os-cloud rustack_system flavor create \
--ram <size_mb> \
--vcpus <no_vcpus> \
<flavor_name>
# openstack --os-cloud rustack_system flavor set \
--property <key=value> \
--property <key=value> \
<flavor_name>
# openstack --os-cloud rustack_system flavor unset \
--property <key> \
<flavor>
# openstack --os-cloud rustack_system flavor list
# openstack --os-cloud rustack_system flavor show <flavor>
# openstack --os-cloud rustack_system flavor delete <flavor>
```

3.2. Примеры создания конфигураций

Для некоторых рабочих нагрузок выгодна настраиваемая топология ЦП. Например, в некоторых операционных системах или ПО может потребоваться другой тип лицензирования в зависимости от количества сокетов ЦП.

Создадим конфигурацию, при использовании которой ВМ будет иметь два сокета ЦП с процессорами, имеющими по два ядра:

```
# openstack --os-cloud rustack_system flavor set \
--vcpus 4 --ram 1024 --public \
--property hw:cpu_sockets=2 \
--property hw:cpu_cores=2 \
processor_topology_flavor
```

Аналогично, создадим Конфигурацию для одного процессора и одного потока:

```
# openstack --os-cloud rustack_system flavor create \
```

```
--vcpus 1 --ram 1024 --public \
--property hw:cpu_cores=1 \
--property hw:cpu_threads=1 \
processor_topology_flavor
```

При указании параметров нужно учесть, что их значения должны удовлетворять условию: **cpu_sockets × cpu_cores × cpu_threads = количество виртуальных ЦП в Конфигурации**. Значения, не указанные явно, в этой формуле принимаются за 1. Конфигурация не удовлетворяющая этому условию будет неработоспособна и приведет к ошибке при создании ВМ.

Примеры:

2 Сокета × 4 Ядра × 2 Потока = 16 Виртуальных ЦП

1 Сокет × 3 ядра × 3 потока = 9 Виртуальных ЦП

8 Сокетов × 2 Ядра × 4 Потока = 64 Виртуальных ЦП

3.3. CPU-pinning

Для некоторых задач требуется работа ВМ в режиме реального или почти реального времени, что невозможно из-за задержек, создаваемых политикой ЦП по умолчанию. В этом случае полезно контролировать, какие ЦП узла привязаны к виртуальным ЦП ВМ. Этот механизм известен как закрепление ЦП или cpu-pinning. ВМ с закрепленными ЦП не может использовать ЦП другой ВМ с закрепленными ЦП, тем самым предотвращается конфликт физических ресурсов между такими ВМ.

Nova обрабатывает процессоры узла, используемые для незакрепленных ВМ иначе, чем те, которые используются ВМ с закрепленными ЦП. Первые отслеживаются при размещении с использованием типа ресурса **VCPU** и могут использовать переподписку физических ресурсов ЦП, а вторые отслеживаются с использованием типа ресурса **PCPU**. По умолчанию nova будет сообщать обо всех ЦП узла как о VCPU. Однако, это поведение можно настроить с помощью параметров конфига:

- **compute/cpu_shared_set** – указать, какие ЦП узла следует использовать для инвентаризации **VCPU** и запуска незакрепленных ВМ;
- **compute/cpu_dedicated_set** – указать, какие ЦП узла следует использовать для инвентаризации **PCPU** и запуска закрепленных ВМ.

Рассмотрим физический узел с 24 физическими ядрами ЦП с включенным Hyperthreading (HT). Администратору нужно зарезервировать первое физическое ядро ЦП и его поток для обработки процессов узла (не для использования ВМ). Кроме того, администратору нужно использовать 8 физических ядер ЦП узла и их потоки для выделения закрепленным ВМ. Остальные 15 физических ядер ЦП узла и их потоки будут использоваться для выделения незакрепленным ВМ с переподпиской 8:1. Нумерация ядер начинается с цифры ноль. Так как используется HT, то количество ядер умножается на два.

На вычислительном узле потребуется внести изменения в конфиг `/etc/nova/nova.conf` и перезагрузить службу nova-compute на вычислительном узле:

```
[DEFAULT]
cpu_allocation_ratio=8.0

[compute]
cpu_dedicated_set=2-17
cpu_shared_set=18-47
```

Проверим результат - узнаем ID гипервизора с именем 2021-comp1.node.rustack.example.com и посмотрим на предоставляемые им ресурсы:

```
# openstack --os-cloud rustack_system hypervisor list --matching comp1 -c ID -f value
3479dcf8-10b5-4350-9735-a6c3d3bcb934

# openstack --os-cloud rustack_system resource provider inventory list 3479dcf8-10b5-4350-
9735-a6c3d3bcb934
+----------------+------------------+----------+----------+----------+-----------+------
-+
| resource_class | allocation_ratio | min_unit | max_unit | reserved | step_size | total
|
+----------------+------------------+----------+----------+----------+-----------+------
-+
| VCPU           |              8.0 |        1 |       30 |        0 |         1 |    30
|
| MEMORY_MB      |              1.0 |        1 |   160160 |     3750 |         1 |160160
|
| DISK_GB        |              1.0 |        1 |       39 |        0 |         1 |    39
|
| PCPU           |              1.0 |        1 |       16 |        0 |         1 |    16
|
+----------------+------------------+----------+----------+----------+-----------+------
-+
```

Значения PCPU и VCPU объединяются при расчете квоты по ядрам.

Политики закрепления ЦП настраиваются с помощью метаданных `hw:cpu_policy`. Существует три политики: выделенная (**dedicated**), смешанная (**mixed**) и общая (**shared**). Общая (shared) политика используется по умолчанию и указывает на то, что у ВМ нет закрепленных ЦП.

Ни одна ВМ с закрепленными ЦП не может использовать ядра другой ВМ с закрепленными ЦП, что предотвращает конфликт ресурсов между ними.

```
# openstack --os-cloud rustack_system flavor create \
--vcpus 2 --ram 1024 --public \
--property hw:cpu_policy=dedicated \
cpu_pinned

# virsh vcpuinfo instance-00000021
VCPU:           0
CPU:            2
State:          running
CPU time:       5.2s
CPU Affinity:   --y-

VCPU:           1
CPU:            0
State:          running
CPU time:       1.4s
CPU Affinity:   y---
```

Смешанная (mixed) политика указывает на то, что ВМ использует закрепленные ЦП вместе с незакрепленными ЦП. ЦП, закрепленные за ВМ, можно указать в метаданных `hw:cpu_dedicated_mask`.

Пример настройки метаданных для использования смешанной политики ЦП. 4 виртуальные ЦП, первые 2 из которых — закрепленные ЦП:

```
# openstack --os-cloud rustack_system flavor set \
--property hw:cpu_policy=mixed \
```

```
--property hw:cpu_dedicated_mask=0-1 \
mixed_policy
```

Для отделения ВМ с закрепленными ЦП от ВМ с незакрепленными ЦП следует использовать агрегаты узлов. Поскольку ВМ с незакрепленными ЦП не учитывают требования к выделению ЦП у закрепленных при запуске на одних узлах.

Особое внимание нужно уделять узлам с процессорами, поддерживающие различные SMT технологии. Потоки в ЦП с многопоточностью имеют ряд общих компонентов, и конкуренция за эти компоненты может повлиять на производительность. Чтобы настроить использование потоков, необходимо указать специальную политику потоков ЦП.

Для рабочих нагрузок, в которых совместное использование повышает производительность, используйте SMT потоки:

```
# openstack --os-cloud rustack_system flavor set \
--property hw:cpu_policy=dedicated \
--property hw:cpu_thread_policy=require \
cpu_smt_pinned
```

Для рабочих нагрузок, где на производительность влияет конкуренция за ресурсы, используйте узлы без потоков или узлы без технологии SMT:

```
# openstack --os-cloud rustack_system flavor set \
--property hw:cpu_policy=dedicated \
--property hw:cpu_thread_policy=isolate \
cpu_no-smt_pinned
```

Наконец, для рабочих нагрузок, при которых влияние SMT на производительность минимально, можно использовать потоки, если они доступны. Если узлы с многопоточностью будут недоступны — ВМ запустится на любых других узлах. Это значение в политике потоков по умолчанию, но его можно установить явно:

```
# openstack --os-cloud rustack_system flavor set \
--property hw:cpu_policy=dedicated \
--property hw:cpu_thread_policy=prefer \
cpu_prefer-smt_pinned
```

3.4. Специальные характеристики узлов (trait)

С помощью метаданных Конфигураций вида **trait:** возможно запускать ВМ на узлах, которые имеют определенные характеристики аппаратного обеспечения, например, узлы, ЦП которых поддерживает SMT, или наборы команд AVX. Синтаксис метаданных принимает вид **trait:<название_характеристики>=required**, если характеристика требуется или в противоположном случае **trait:<название_характеристики>=forbidden**.

Пример создания конфигурации для запуска ВМ на узлах, ЦП которых поддерживают набор инструкций AVX2:

```
# openstack --os-cloud rustack_system flavor set \
--property trait:HW_CPU_X86_AVX2=required \
cpu_req_avx2
```

Список доступных характеристик (trait) для узла можно получить командой:

```
    # openstack --os-cloud rustack_system hypervisor list --matching comp1 -c ID -f
value
    3479dcf8-10b5-4350-9735-a6c3d3bcb934

    # openstack --os-cloud rustack_system resource provider trait list -f value
3479dcf8-10b5-4350-9735-a6c3d3bcb934
    HW_CPU_X86_AVX512F
    HW_CPU_X86_AVX512CD
    HW_CPU_X86_AVX512BW
    HW_CPU_X86_AVX512DQ
    HW_CPU_X86_AVX512VL
    HW_CPU_X86_INTEL_VMX
    ...SKIP...
```

3.5. Квоты и QoS

С помощью метаданных Конфигураций вида **quota:** возможно определять различные квоты для ВМ, например, квоты CPU-времени выполнения на гипервизоре для данной ВМ. Все ВМ по-умолчанию создаются с квотой выполнения 1024. Если в метаданные передать `quota:cpu_shares = 512`, то все ВМ с этим профилем будут получать в два раза меньше процессорного времени на гипервизоре (аналог `cpu_shares` для `systemd`) при конкуренции с другими ВМ за ресурсы ЦП.

Также может быть полезно включить механизм сетевого шейпинга (QoS) для интерфейсов ВМ через метаданные.

Поддерживаемые опции:

- `quota:vif_outbound_average` = число в килобайтах;
- `quota:vif_outbound_burst` = число в килобайтах;
- `quota:vif_inbound_peak` = число в килобайтах;
- `quota:vif_inbound_average` = число в килобайтах;
- `quota:vif_inbound_burst` = число в килобайтах;
- `quota:vif_outbound_peak` = число в килобайтах.

Метаданные для ограничения дискового ввода-вывода:

`disk_read_bytes_sec`
`disk_read_iops_sec`
`disk_write_bytes_sec`
`disk_write_iops_sec`
`disk_total_bytes_sec`
`disk_total_iops_sec`

Используя квоты дискового ввода-вывода, можно задать максимальную скорость записи на основной диск 10 МБ в секунду для пользователя виртуальной машины. Например:

```
    # openstack --os-cloud rustack_system flavor set \
    --property quota:disk_write_bytes_sec=10485760 \
    disk_quotas
```

4. Планировщик Nova

ПВ РУСТЭК использует службу `nova-scheduler`, чтобы определить, на каком физическом узле или агрегате узлов следует разместить экземпляр ВМ. Когда служба `nova-compute` получает запрос на запуск или миграцию ВМ, она использует указанные в запросе спецификации, конфигурацию и образ для поиска подходящего узла. Например, в конфигурации с помощью метаданных можно указывать различные признаки, по которым будет выбран узел для запуска экземпляра ВМ: тип диска, расширение набора инструкций ЦП, выделенный GPU и многое другое. Служба `nova-scheduler` запускается на всех узлах с ролью "Управление ВМ". В контексте фильтров термин «узел» означает физический узел с ролью "Физический узел", на котором запущена служба `nova-compute`. Так же планировщик Nova при выборе решения опирается на службу Placement (uwsgi.placement-api), которая так же запускается на всех узлах с ролью "Управление ВМ".

Служба планировщика Nova для определения узла, на котором нужно запустить или переместить ВМ, использует компоненты в следующем порядке :

1. **Предварительные фильтры** службы Placement. Служба планировщика вычислений использует службу Placement для фильтрации набора узлов-кандидатов на основе определенных атрибутов. Например, сервис Placement автоматически исключает отключенные физические узлы.
2. **Фильтры.** Используются службой планировщика Nova для определения начального набора физических узлов, на которых следует запускать экземпляр ВМ.
3. **Веса**. Служба планировщика отдает приоритет отфильтрованным физическим узлам с помощью системы взвешивания. Наибольший вес имеет наивысший приоритет.

4.1. Предварительная фильтрация с помощью Placement

Служба `nova-compute` взаимодействует со службой Placement при создании экземпляров ВМ и управлении ими. Служба Placement отслеживает инвентаризацию и использование поставщиков ресурсов, таких как физические узлы, общие пулы хранения или пулы распределения IP-адресов, а также их доступные количественные ресурсы, такие как виртуальные ЦП. Любая служба, которой необходимо управлять выбором и анализом потребления ресурсов, может использовать Placement.

Placement также отслеживает доступность качественных ресурсов у их поставщиков, например, тип диска хранения, которым обладает поставщик ресурсов.

Служба Placement применяет предварительные фильтры к набору физических узлов-кандидатов на основе инвентаризаций и специальных характеристик узлов (trait) поставщика ресурсов. Можно создавать предварительные фильтры на основе следующих критериев:

- поддерживаемые типы образов;
- специальные характеристики узлов (trait);
- проекты или пользователи;
- зона доступности.

Доступные на платформе характеристики можно вывести с помощью команды:

```
# openstack --os-cloud rustack_system trait list
```

Для вывода характеристик определенного узла, зная его `uuid`, можно воспользоваться командой:

```
# openstack --os-cloud rustack_system resource provider trait list <uuid-узла>
```

4.2. Фильтрация по характеристикам поставщика ресурсов

У каждого поставщика ресурсов есть набор специальных характеристик. Это качественные аспекты поставщика ресурсов, например, тип диска хранения или расширение набора инструкций ЦП.

Физический узел сообщает о своих возможностях сервису Placement в виде специальных характеристик. Таким образом, экземпляр ВМ может указать, какие из этих характеристик ему необходимы, а какие не должны присутствовать у поставщика ресурсов. Планировщик физических ресурсов может использовать эти признаки для определения подходящего физического узла или агрегата узлов для размещения экземпляра ВМ.

Чтобы позволить пользователям создавать ВМ на узлах с определенными характеристиками, можно определить вариант Конфигурации, требующий или запрещающий определенную характеристику. Также можно создать образ, требующий или запрещающий ту или иную характеристику.

4.3. Создание образа, требующего или запрещающего характеристику поставщика ресурсов

Чтобы запланировать ВМ на узле или агрегате узлов, который имеет какую-либо обязательную характеристику, можно добавить эту характеристику в дополнительную метадату образа. Например, чтобы запланировать экземпляры на узле или агрегате, который поддерживает AVX-512, нужно добавить следующую характеристику к образу:

```
# openstack --os-cloud rustack_system image set \
--property trait:HW_CPU_X86_AVX512BW=required \
trait-image
```

Чтобы отфильтровать узлы, которые имеют запрещенную характеристику, нужно добавить ее в метадату образа. Например, чтобы запретить планирование ВМ на узле или агрегате, который поддерживает с множественным подключением дисков, нужно добавить метадату к образу:

```
# openstack --os-cloud rustack_system image set \
--property trait:COMPUTE_VOLUME_MULTI_ATTACH=forbidden \
trait-image
```

4.4. Фильтры Nova

Используемые фильтры определены в конфиге `/etc/nova/nova.conf` службы `nova-scheduler` и применяются в порядке написания:

- **AvailabilityZoneFilter** — фильтрует узлы по зоне доступности. Фильтр пропускает узлы, соответствующие зоне доступности;
- **ComputeFilter** — пропускает любой узел, служба `nova-compute` которого включена и работает;
- **ImagePropertiesFilter** — фильтрует узлы на основе архитектуры ЦП, типа гипервизора и режима виртуальной машины;
- **ServerGroupAntiAffinityFilter** — фильтр реализует антиаффинность для группы ВМ. Сначала вы должны создать группу ВМ с политикой антиаффинности, затем при создании новой ВМ укажите подсказку планировщика «`group=<uuid>`», где `<uuid>` — это UUID созданной вами группы ВМ. Это приведет к тому, что ВМ будет добавлена в группу. Когда ВМ будет запланирована, политика антиаффинности будет применена ко всем ВМ в этой группе;
- **ServerGroupAffinityFilter** — фильтр работает как **ServerGroupAntiAffinityFilter** с противоположным эффектом.
- **AggregateInstanceExtraSpecsFilter** — фильтр отвечает за включение механизма планировщика ВМ на агрегатах узлов с определенными атрибутами

```
[filter_scheduler]
available_filters = nova.scheduler.filters.all_filters
enabled_filters = AvailabilityZoneFilter,ComputeFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter,AggregateInstanceExtraSpecsFilter
```

4.5. Веса

Взвешиваются только узлы, прошедшие все фильтры. Вес любого узла начинается с 0. Взвешивание упорядочивает эти узлы, добавляя или вычитая из значения веса узла, назначенного предыдущим взвешивающим. Веса могут быть отрицательными. Экземпляр ВМ будет назначен на один из узлов с наибольшим весом.

Планировщик определяет вес каждого физического узла, выполняя следующие задачи:

- Планировщик нормализует каждый вес до значения от 0,0 до 1,0;
- Планировщик умножает нормализованный вес на множитель веса.

Планировщик вычисляет весовую нормализацию для каждого типа ресурсов, используя нижнее и верхнее значения доступности ресурсов для физических узлов-кандидатов:

- узлам с наименьшей доступностью ресурса (minval) присваивается 0;
- узлам с наивысшей доступностью ресурса (maxval) присваивается 1;
- узлам с доступностью ресурсов в диапазоне minval-maxval назначается нормализованный вес, рассчитываемый по следующей формуле: (node_resource_availability − minval) / (maxval − minval).

Если все физические узлы имеют одинаковую доступность ресурса, то все они нормализуются до 0.

Планировщик фильтров взвешивает узлы на основе параметра конфигурации `[filter_scheduler]weight_classes`, по умолчанию это `nova.scheduler.weights.all_weighers`, который выбирает следующие взвешиватели:

- **RAMWeigher** — вычисляет вес на основе доступной оперативной памяти на физическом узле. Сортировка производится по наибольшему весу. Если множитель `[filter_scheduler]ram_weight_multiplier` отрицательный, больший вес будет у узла с наименьшим объемом доступной оперативной памяти.
- **CPUWeigher** — вычисляет вес на основе доступных виртуальных ЦП на физическом узле. Сортировка производится по наибольшему выигрышному весу. Если множитель `[filter_scheduler]cpu_weight_multiplier` отрицательный, больший вес будет у узла с наименьшим количеством доступных ЦП.
- **PCIWeigher** — вычисляет вес на основе количества устройств PCI на узле и количества устройств PCI, запрошенных ВМ. Например, при наличии трех узлов — одного с одним устройством PCI, одного с множеством устройств PCI и одного без устройств PCI — nova должна расставлять приоритеты для них по-разному в зависимости от требований ВМ. Если ВМ запрашивает одно устройство PCI, то предпочтение следует отдавать первому из узлов. Точно так же, если ВМ запрашивает несколько устройств PCI, то предпочтение отдается второму из этих узлов. Наконец, если ВМ не запрашивает PCI-устройство, предпочтение следует отдавать последнему из этих узлов. Чтобы взвешивание работало должен быть включен хотя бы один из фильтров: **PciPassthroughFilter** или **NUMATopologyFilter.**
- **ServerGroupSoftAffinityWeigher** — вычисляет вес на основе количества экземпляров, работающих в одной и той же группе серверов. Наибольший вес определяет предпочтительный узел для нового экземпляра ВМ. Для множителя допускается только положительное значение.
- **ServerGroupSoftAntiAffinityWeigher** — вычисляет вес противоположно **ServerGroupSoftAffinityWeigher.**
- **BuildFailureWeigher** - вычисляет вес по количеству недавних неудачных попыток загрузки ВМ. Если найдено значение с ключом `build_failure_weight_multiplier` в метаданных агрегата, это значение будет выбрано в качестве множителя веса. Если метаданных агрегата для узла найдено более одного значения, будет использовано минимальное значение.

5. Диски и СХД

5.1. Бэкенды

5.1.1. Описание работы компонентов

Служба блочного хранилища Cinder устанавливается на узлы ПВ РУСТЭК с ролью "Управление дисками" и управляет администрированием, безопасностью, планированием и жизненным циклом всех дисков ВМ. Диски используются в качестве основной формы постоянного хранилища для экземпляров ВМ.

Метод предоставления и использования хранилища определяется драйвером хранилища. При установке в ПВ РУСТЭК можно выбрать из двух основных типов драйверов: NFS и OCFS. Также существует множество драйверов для всевозможных СХД, как программных, так и аппаратных, которые при необходимости нужно настраивать вручную. Матрица совместимости доступна по адресу: `https://wiki.openstack.org/wiki/CinderSupportMatrix`

Компоненты Cinder, являющиеся частью инсталляции ПВ РУСТЭК:

Компонент	Описание
uwsgi.cinder-api	Точка входа для запросов в сервис по протоколу HTTP. Приняв запрос, сервис проверят полномочия на выполнение запроса и переправляет брокеру сообщений для доставки другим службам. Использует конфигурационный файл /etc/cinder/cinder.conf. Логи пишутся в файл /var/log/cinder/api.log. Работает в режиме «active-active», то есть запущен одновременно на всех узлах с ролью "Управление дисками".
cinder-scheduler	Сервис-планировщик принимает запросы от брокера сообщений и определяет, какой узел с работающим сервисом cinder-volume должен обработать запрос. Использует конфигурационный файл /etc/cinder/cinder.conf. Логи пишутся в файл /var/log/cinder/cinder-scheduler.log. Работает в режиме «active-active», то есть запущен одновременно на всех узлах с ролью "Управление дисками".
cinder-volume	Сервис отвечает за взаимодействие с бэкендом. Получает запросы от Сервиса-планировщика и транслирует непосредственно в хранилище. Cinder позволяет одновременно использовать несколько бэкендов. При этом при помощи фильтров CapacityFilter и CapacityWeigher или дисковых типов можно управлять тем, какой бэкенд выберет планировщик. Использует конфигурационный файл /etc/cinder/cinder.conf. Логи пишутся в файл /var/log/cinder/cinder-volume.log Работает в режиме «active-passive», то есть запущен только на одном из узлов с ролью "Управление дисками". Управление запуском службы производится сервисом keepalived.

Проверить состояние всех сервисов Cinder можно командой `openstack volume service list` на любом из узлов платформы:

```
[root@controller1 ~]# openstack volume service list
+------------------+-------------------------+------+---------+-------+----------------------------+
| Binary           | Host                    | Zone | Status  | State | Updated At                 |
+------------------+-------------------------+------+---------+-------+----------------------------+
| cinder-scheduler | ohost2.node.example.com | nova | enabled | up    | 2022-12-16T15:11:33.870505 |
| cinder-scheduler | ohost3.node.example.com | nova | enabled | up    | 2022-12-16T15:11:34.162981 |
| cinder-scheduler | ohost1.node.example.com | nova | enabled | up    | 2022-12-16T15:11:35.033050 |
| cinder-backup    | ohost2.node.example.com | nova | enabled | up    | 2022-12-16T15:11:40.114567 |
| cinder-volume    | ohost1@ocfs2            | nova | enabled | up    | 2022-12-16T15:11:39.233331 |
| cinder-backup    | ohost3.node.example.com | nova | enabled | up    | 2022-12-16T15:11:39.346477 |
| cinder-backup    | ohost1.node.example.com | nova | enabled | up    | 2022-12-16T15:11:40.690475 |
+------------------+-------------------------+------+---------+-------+----------------------------+
```

Для развертывания кластерной ФС OCFS2 вы должны установить адаптеры (HBA) на всех узлах с ролью "Управление дисками" и "физический узел" в случае использования Fibre Channel (FC) или осуществить подключение через iSCSI к этим узлам.

Выбрать и сконфигурировать тип дискового хранилища по умолчанию возможно из РУСТЭК.Конфигуратора. Дополнительные типы подключаются с помощью конфигурационных файлов, которые располагаются на кластерной ФС Glusterfs и доступны на всех узлах платформы по пути /etc/openstack/cinder_backends/.

```
# ls -l /etc/openstack/cinder_backends/
-rw-r----- 1 cinder cinder 664 Dec 14 14:19 netapp-nfs.conf
-rw-r----- 1 cinder cinder 356 Nov 23 19:55 ocfs2.conf
```

5.1.2. Тип дискового хранилища NFS

Самый простой тип хранилища, который использует шару на NFS-сервере для хранения дисков ВМ. Для использования необходим доступный NFS-сервер, имеющий:

- права 0755 на экспортируемый каталог;
- опции`no_subtree_check,no_root_squash`;
- отключенную на стороне сервера опцию `-g` или `--manage-gids` (см. man rpc.mountd, опция включена по умолчанию в Debian, отключена в Gentoo, CentOS).

В случае NFS шары монтируются в директорию на узле с запущенным cinder-volume группы "Управление дисками":

```
# ls -ld /mnt/cinder/*
drwxrwxr-x 3 cinder cinder 4096 Nov 21 14:26
/mnt/cinder/462c060aeacd50ad6d0bcbd2d6e322a7
```

Точкой монтирования управляет переменная nfs_mount_point_base в конфиге бэкенда /etc/openstack/cinder_backends/nfs.conf. По-умолчанию для NFS значение переменной /mnt/cinder.

На узлах группы "физический узел" NFS-шара монтируется по требованию при создании ВМ:

```
# ls -ld /var/lib/nova/mnt/*
drwxr-xr-x 2 cinder cinder 4096 Dec 14 14:21
/var/lib/nova/mnt/7174f653526ecce0bca485b9d3c5eaf7
```

Параметры, используемые в бэкенде NFS:

- **volume_backend_name = nfs** — параметр используется для случаев настройки multi-storage бэкендов: позволяет объединить несколько бэкендов под одним именем и в случае NFS имеет значение nfs по умолчанию;
- **nfs nas_secure_file_permissions = true** — параметр устанавливает более безопасные права доступа к файлам на NAS для ограничения широкого доступа. Если установлено значение

True, тома создаются с разрешениями для пользователя и группа cinder (660), по умолчанию (true);

- **nas_secure_file_operations = true** — параметр разрешает NAS работать в безопасной среде, где доступ root к шаре запрещен. Если установлено значение False, доступ пользователя root разрешен, что небезопасно. Значение по умолчанию: true;
- **volume_clear = zero** — метод заполнения удаляемых дисков;
- **volume_driver = cinder.volume.drivers.nfs.NfsDriver** — используемый драйвер;
- **max_over_subscription_ratio = 1.0** — переподписка;
- **nfs_shares_config = /etc/cinder/nfs_shares** — конфиг к файлу с описанием точки и параметров монтирования NFS шар с сервера;
- **nfs_sparsed_volumes = true** — параметр создания дисков в виде разреженных файлов. Если установлено значение False, диск создается как обычный файл. В таком случае создание тома занимает много времени, по умолчанию (true);
- **nfs_qcow2_volumes = false** — параметр создания qcow2 или raw дисков. Значение по умолчанию: false (создаются raw диски);
- **nfs_snapshot_support = true** — параметр включения поддержки снапшотов на дисках с NFS шары. Значение по умолчанию: true;
- **nfs_mount_point_base = /mnt/cinder** — точка монтирования шары с NFS сервера на узлах с ролями "Управления дисками". Значение по умолчанию: (/mnt/cinder)
  backup_use_temp_snapshot = true.

Дополнительные настройки:

- **reserved_percentage = 0** — процент от свободного места, которое будет зарезервировано на бэкенде;
- **image_volume_cache_enabled = false** — активация кэширования образов Glance на данном бэкенде;
- **image_volume_cache_max_size_gb = 600** — размер в Гб кэша для образов Glance на данном бэкенде;
- **image_volume_cache_max_count = 50** — количество образов в кэше Glance на данном бэкенде.

На NFS не работают server_clone, volume_backup и volume_snapshot.

5.1.3. Тип дискового хранилища OCFS2

Параметры, используемые в бэкенде OCFS2:

- **volume_backend_name = ocfs2** — параметр используется для случаев настройки multi-storage бэкендов: позволяет объединить несколько бэкендов под одним именем и в случае NFS имеет значение nfs по умолчанию;
- **nfs nas_secure_file_permissions = true** — параметр устанавливает более безопасные права доступа к файлам на NAS для ограничения широкого доступа. Если установлено значение True, тома создаются с разрешениями для пользователя и группа cinder (660), по умолчанию (true);
- **nas_secure_file_operations = true** — параметр разрешает NAS работать в безопасной среде, где доступ root к шаре запрещен. Если установлено значение False, доступ пользователя root разрешен, что небезопасно. Значение по умолчанию: true;
- **volume_clear = zero** — метод заполнения удаляемых дисков;
- **volume_clear_ionice = -c3** — метод заполнения удаляемых дисков volume_clear_ionice = -c3 – параметры для изменения приоритета ввода-вывода процессом ionice, используемого для обнуления тома после удаления, например «-c3» для приоритет "idle";
- **volume_driver = crutches.cinder.volume.drivers.sharedfs.SharedFS** — используемый драйвер;
- **sharedfs_shares_config = /etc/cinder/ocfs_shares** — конфиг к файлу с описанием точки и параметров монтирования NFS шар с сервера;
- **sharedfs_sparsed_volumes = true** — параметр создания дисков в виде разреженных файлов. Если установлено значение False, диск создается как обычный файл. В таком случае создание тома занимает много времени. Значение по умолчанию: true;
- **sharedfs_reflink_cmd = cp --reflink** — команда для использования CoW функционала.

Дополнительные настройки:

- **reserved_percentage = 0** — процент от свободного места, которое будет зарезервировано на бэкенде;
- **image_volume_cache_enabled = false** — активация кэширования образов Glance на данном бэкенде;
- **image_volume_cache_max_size_gb = 600** — размер в Гб кэша для образов Glance на данном бэкенде;
- **image_volume_cache_max_count = 50** — количество образов в кэше Glance на данном бэкенде.

5.1.4. Подключение дополнительных типов дискового хранилища

Для подключения дополнительного типа хранилища необходимо создать конфиг в директории /etc/openstack/cinder_backends/ и добавить значение переменной volume_backend_name в переменную enabled_backends файла /etc/cinder/cinder.conf.

Пример конфига к netapp-nfs драйверу:

```
[netapp-nfs1]
backend_host = hw-n2
volume_backend_name=netapp-nfs
volume_driver=cinder.volume.drivers.netapp.common.NetAppDriver
netapp_server_hostname=10.0.0.1
netapp_server_port=80
nas_secure_file_permissions = true
nas_secure_file_operations = true
netapp_transport_type=http
netapp_storage_protocol=nfs
netapp_storage_family=ontap_cluster
netapp_login=cinder-api-user
netapp_password=P@ssw0rd123
netapp_vserver=SVM_NFS_01
nfs_shares_config=/etc/openstack/cinder/netapp_shares
max_over_subscription_ratio=1.0
reserved_percentage=5
nfs_sparsed_volumes = true
nfs_qcow2_volumes = false
nfs_mount_options=lookupcache=pos
nfs_mount_point_base = /mnt/cinder-netapp
```

Подключенный бэкенд в /etc/cinder/cinder.conf вместе с дефолтным ocfs2:

```
# grep enabled_backends /etc/cinder/cinder.conf
enabled_backends = ocfs2,netapp-nfs
```

РУСТЭК.Конфигуратор при прогоне не изменяет дополнительные конфигурационные файлы бэкендов, отличные от **/etc/openstack/cinder_backends/nfs.conf** и **/etc/openstack/cinder_backends/ocfs2.conf.**
Также РУСТЭК.Конфигуратор не удаляет созданные дополнительные типы дисков.

5.1.5. Директории хранения дисков

В случае использования OCFS2, LUN монтируются на всех узлах с ролями "Управление дисками" и "физический узел" в директорию /mnt/ocfs2-<wwid>:

```
# ls -ld /mnt/ocfs2*
```

```
    drwxr-xr-x    3    cinder    cinder    3896    Dec    16    23:38    /mnt/ocfs2-
360000000000000000e00000000020001
    drwxr-xr-x    2    cinder    cinder    3896    Dec    16    23:38    /mnt/ocfs2-
360000000000000000e00000000030001
```

В случае использования NFS диски ВМ монтируются по пути /mnt/cinder/<внутренний_хеш_шары> на узле с запущенным cinder-volume и в /var/lib/nova/mnt/<внутренний_хеш_шары> на узле с работающей ВМ.

5.1.6. Переподписка (oversubscription)

При использовании NFS драйвера для задания коэффициента переподписки используется опция max_over_subscription_ratio, которая задаётся в конфигурационном файле бэкенда /etc/openstack/cinder_backends/nfs.conf в секции [nfs]. Также для корректной работы переподписки опция nfs_sparsed_volumes должна иметь значение true.

После этого изменения нужно перезапустить работающую службу cinder-volume на узлах с ролью "Управление дисками". Для этого можно воспользоваться командой:

```
    # ansible -i /var/lib/rustack-ansible/inventory.yml cinder -a "rc-service -s
cinder-volume restart"
```

По умолчанию коэффициент max_over_subscription_ratio имеет значение 1.0. Это означает, что выделенный объём для дисков не может превышать общий физический объём хранилища. Если хранилище объёмом 100 Гб, то значит на нём нельзя создать тонких дисков объёмом больше, чем на 100 Гб.

Если изменить коэффициент max_over_subscription_ratio на значение 10.5, то выделенный объём для дисков может превысить общий физический объём хранилища в 10.5 раз. Тогда в хранилище объёмом 100 Гб можно создать тонких дисков суммарным объёмом 1050 Гб.

```
[nfs]
backend_host = 2021-cont1
volume_backend_name = nfs
nas_secure_file_permissions = true
nas_secure_file_operations = true
volume_clear = zero
volume_clear_ionice = -c3
volume_driver = cinder.volume.drivers.nfs.NfsDriver
max_over_subscription_ratio = 10.0
nfs_shares_config = /etc/cinder/nfs_shares
nfs_sparsed_volumes = true
nfs_qcow2_volumes = false
nfs_snapshot_support = true
nfs_mount_point_base = /mnt/cinder
backup_use_temp_snapshot = true
```

5.1.7. Формат файлов диска

В зависимости от выбранного формата файлов диска в конфиге бэкенда /etc/openstack/cinder_backends/ будут заполнены различные опции, относящиеся к формату файлов хранимых дисков.

Типы используемых хранимых дисков на бэкендах NFS и OCFS2:

- raw — "толстые" файлы тома, место на бэкенде полностью выделяется во время создания. Это приводит к увеличению задержек (зависит от скорости хранилища) при создании диска;
- qcow — файлы в формате qcow2;
- sparse — "разреженные (тонкие)" файлы тома, место на бэкенде выделяется не полностью и разреженные файлы увеличиваются по мере необходимости. Настройка применяется для файлов формата raw;

В случае инсталляции с бэкендом NFS:

При использовании RAW файлов рекомендуется активировать nfs_sparsed_volumes, т.к. NFS не поддерживает вызов fallocate() и файлы будут создаваться посредством dd.

В случае инсталляции с бэкендом OCFS2:

Невозможно использование дисков в формате qcow, т.к. это приводит к большому падению производительности при записи на диск. Образы в формате qcow будут автоматически сконвертированы в raw на этапе создания диска.

При использовании RAW файлов имеет смысл активировать sharedfs_sparsed_volumes, т.к. OCFS2 поддерживает вызов fallocate() и файлы будут создаваться посредством вызова fallocate почти мгновенно.

5.1.8. Безопасное удаление дисков ВМ

Для соответствия требованиям в части информационной безопасности в платформе ПВ РУСТЭК существует несколько методик удаления дисков. Для настройки используется параметр **volume_clear** в настройках бэкенда.

Параметр может принимать следующие значения:

- none — файл диска удаляется посредством команды rm;
- zero — файл диска перезаписывается нолями в один проход посредством dd и затем удаляется посредством команды rm.

Следует учитывать, что при использовании volume_clear=zero удаление дисков занимает продолжительное время. В случае перезаписи sparse дисков нолями перезаписывается полный размер диска.

5.2. Дисковые типы

5.2.1. Тип диска

По умолчанию в системе созданы три типа диска: glance_nfs , nfs или ocfs2, в зависимости от выбранного типа в РУСТЭК.Конфигураторе при развертывании платформы, и __DEFAULT__.

- **glance_nfs** — сервисный тип диска, использующийся сервисом Glance для сохранения образов на хранилище NFS.
- **nfs** — тип диска, использующийся сервисом Cinder для хранения дисков ВМ на хранилище NFS.
- **ocfs2** — тип диска, использующийся сервисом Cinder для хранения дисков ВМ на блочных устройствах (LUN), поданных на узлы по прокотолам iSCSI или FC.
- **__DEFAULT__** — тип диска по умолчанию. Если не указан конкретный тип диска, то будет выбран подходящий на основе использующихся фильтров и весов. Подробнее в разделе 5.3.

```
# openstack volume type list
+--------------------------------------+-------------+-----------+
| ID                                   | Name        | Is Public |
+--------------------------------------+-------------+-----------+
| 0c548032-f19d-4928-85a8-5c94be9a6be0 | glance_nfs  | False     |
| fe86726d-dede-43f5-a1d4-2d188a91773d | nfs         | True      |
| a93daa57-2b72-4bed-9b5b-b58421498b6c | __DEFAULT__ | True      |
+--------------------------------------+-------------+-----------+
```

5.2.2. Создание нового типа диска

Создание нового типа диска может понадобится для того, чтобы привязать его к новому бэкенду, либо чтобы добавить к типу диска QoS (смотрите раздел 5.4).

Создание нового типа диска реализуется командой `openstack volume type create`:

```
# openstack volume type create qos_iops100
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| description | None                                 |
| id          | ced2c227-1b31-4ff9-b16d-8d99d8a14ef1 |
| is_public   | True                                 |
| name        | qos_iops100                          |
+-------------+--------------------------------------+
```

Команда `volume type set` связывает созданный тип диска `qos_iops100` с бэкендом. Имя бэкенда хранится в файле /etc/cinder/cinder.conf. Используется опция enabled_backends (если бэкендов несколько, то выбирается требуемый):

```
$ grep enabled_backends /etc/cinder/cinder.conf
enabled_backends = nfs
$ openstack volume type set --property volume_backend_name=nfs qos_iops100
```

Командой `volume type list --long` проверяется успешность проведённых операций:

```
$ openstack volume type list --long
+--------------------------------------+-------------+-----------+--------------------+-------------
--------------------------------------+
| ID                                   | Name        | Is Public | Description        | Properties
|
+--------------------------------------+-------------+-----------+--------------------+-------------
--------------------------------------+
| ced2c227-1b31-4ff9-b16d-8d99d8a14ef1 | qos_iops100 | True      | None               |
volume_backend_name='nfs'                          |
| 01f7403c-cc00-4271-ad12-c50043663936 | glance_nfs  | False     | None               | multiattach='<is>
True', volume_backend_name='nfs' |
| e80579d1-9eb2-4c0f-8375-95f200448e66 | nfs         | True      | None               |
volume_backend_name='nfs'                          |
| a93daa57-2b72-4bed-9b5b-b58421498b6c | __DEFAULT__ | True      | Default Volume Type |
|
+--------------------------------------+-------------+-----------+--------------------+-------------
--------------------------------------+
```

В данном примере типы nfs и qos_iops100 имеют одинаковый бэкенд.

Вывод команды `volume type list --long` при создании типа диска для нового бэкенда:

```
$ openstack volume type list --long
+--------------------------------------+-------------+-----------+--------------------+-------------
--------------------------------------+
| ID                                   | Name        | Is Public | Description        | Properties
|
+--------------------------------------+-------------+-----------+--------------------+-------------
--------------------------------------+
| 4c27a9b9-8a0e-491e-bbb6-39e5683aae2a | nfs_extra   | True      | None               |
volume_backend_name='nfs2'                         |
| 01f7403c-cc00-4271-ad12-c50043663936 | glance_nfs  | False     | None               | multiattach='<is>
True', volume_backend_name='nfs' |
| e80579d1-9eb2-4c0f-8375-95f200448e66 | nfs         | True      | None               |
volume_backend_name='nfs'                          |
| a93daa57-2b72-4bed-9b5b-b58421498b6c | __DEFAULT__ | True      | Default Volume Type |
|
```

```
      +--------------------------------------+-------------+-----------+--------------------+------------
  ------------------------------------+
```

В данном примере типы nfs и nfs_extra имеют различные бэкенды.

5.2.3. Удаление типа диска

Для удаления типа диска сначала нужно удалить все диски, использующие данный тип диска, а потом выполнить команду `openstack volume type delete`:

```
      # openstack volume type delete qos_iops100
```

5.3. Планировщик Cinder

5.3.1. Описание

Cinder-scheduler — это сервис, который отвечает за выбор оптимального узла - бэкенда, привязанного к дисковому типу, для размещения блочных устройств хранения (далее диски) в облачной инфраструктуре. Узел выбирается на основе различных критериев, таких как доступная емкость узла, количество созданных дисков и т.д. Для этого сервис использует различные фильтры и веса.

При создании диска Cinder-scheduler выбирает узел с наилучшими показателями на основе заданных фильтров и весов, чтобы обеспечить эффективное использование ресурсов и минимизировать возможные проблемы, такие как перегрузка узла или неравномерное распределение нагрузки.

5.3.2. Фильтры

5.3.2.1. AvailabilityZoneFilter

AvailabilityZoneFilter фильтрует узлы по зонам доступности. Это позволяет избежать создания диска в зоне доступности, отличающейся от той, в которой будет располагаться ВМ, для которой создается диск.

5.3.2.2. CapabilitiesFilter

CapabilitiesFilter фильтрует узлы на основе их отчетных возможностей. Игнорируется с драйверами NFS и OCFS2. При использовании СХД, для работы с которой есть отдельный драйвер, узлы могут сообщать о своих возможностях, таких как типы подключённых дисков, количество и типы сетевых интерфейсов и т.д. Фильтр по возможностям позволяет выбирать узлы, которые имеют требуемые возможности для конкретного запроса на создание блочного устройства хранения.

5.3.2.3. CapacityFilter

CapacityFilter фильтрует узлы по доступной емкости. Это позволяет выбирать узлы, которые имеют достаточно свободной емкости для размещения нового диска.

5.3.3. Веса

5.3.3.1. AllocatedCapacityWeigher

AllocatedCapacityWeigher вычисляет вес для каждого узла на основании уже выделенной емкости. Узел с наименьшим значением получает наибольший вес. Если в конфигурационном файле `/etc/cinder/cinder.conf` определить параметр `allocated_capacity_weight_multiplier` и присвоить ему положительное значение — поведение изменится на противоположное, диски будет размещаться консолидировано.

5.3.3.2. CapacityWeigher

CapacityWeigher вычисляет вес для каждого узла на основе доступной емкости. Узлы с большей доступной емкостью получают более высокий вес и, следовательно, имеют больший приоритет при выборе узла для размещения диска.

5.3.3.3. VolumeNumberWeigher

VolumeNumberWeigher вычисляет вес для каждого узла на основе количества созданных на нем дисков. Узлы с меньшим количеством созданных объемов получают более высокий вес, чтобы уменьшить вероятность перегрузки узла.

5.3.4. Использующиеся веса и фильтры

Настроить фильтры и веса можно в файле **/etc/cinder/cinder.conf** в разделе **[DEFAULT]**. Порядок фильтров в файле конфигурации определяет порядок их применения при выборе узла для размещения диска.

По умолчанию в платформе используются следующие фильтры и веса:

```
    [DEFAULT]
    ...
    scheduler_default_filters                                                    =
AvailabilityZoneFilter,CapabilitiesFilter,CapacityFilter
    scheduler_default_weighers = CapacityWeigher
    ...
```

5.4. Quality of Service (QoS)

5.4.1. Описание

QoS позволяет устанавливать ограничения IOPS для виртуальных дисков, предотвращая исчерпание ресурсов хранилища. Лимиты IOPS могут быть определены числом IOPS или числом байтов в секунду (IOPS имеет приоритет). Можно использовать различные опции ограничения:

- read_iops_sec;
- write_iops_sec;
- total_iops_sec;
- read_bytes_sec;
- write_bytes_sec;
- total_bytes_sec;
- read_iops_sec_max;
- write_iops_sec_max;
- total_iops_sec_max;
- read_bytes_sec_max;
- write_bytes_sec_max;
- total_bytes_sec_max;
- size_iops_sec.

5.4.2. Создание

Создаём новую спецификацию QoS. Задаём тип бэкенда, опцию в формате ключ=значение и имя:

```
    $ openstack volume qos create --consumer front-end --property read_iops_sec=100
my_qos
    +------------+--------------------------------------+
    | Field      | Value                                |
    +------------+--------------------------------------+
    | consumer   | front-end                            |
```

```
| id         | 98b4aba0-b545-4524-a8d5-f150577b16b5 |
| name       | my_qos                               |
| properties | read_iops_sec='100'                  |
+------------+--------------------------------------+
```

С одной спецификацией QoS можно связывать несколько опций. Это можно сделать сразу при создании спецификации, указав несколько параметров **property**:

```
$ openstack volume qos create --consumer front-end --property read_iops_sec=100
--property write_iops_sec=100 my_qos2
+------------+-------------------------------------------+
| Field      | Value                                     |
+------------+-------------------------------------------+
| consumer   | front-end                                 |
| id         | 0a86a985-e781-40a9-9f71-4201d14ca99b      |
| name       | my_qos2                                   |
| properties | read_iops_sec='100', write_iops_sec='100' |
+------------+-------------------------------------------+
```

5.4.3. Редактирование

Опции можно добавить после создания QoS, используя команду set с указанием имени спецификации QoS. Для задания нового значения уже существующей опции достаточно повторно применить команду set с новым значением. Список всех спецификаций QoS можно узнать в выводе команды list.

```
$ openstack volume qos list
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
| ID                                   | Name    | Consumer  | Associations | Properties                                |
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
| 0a86a985-e781-40a9-9f71-4201d14ca99b | my_qos2 | front-end |             | read_iops_sec='100', write_iops_sec='100'
|
| 98b4aba0-b545-4524-a8d5-f150577b16b5 | my_qos  | front-end |             | read_iops_sec='100'                       |
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
$ openstack volume qos set --property write_iops_sec=100 my_qos
$ openstack volume qos list
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
| ID                                   | Name    | Consumer  | Associations | Properties                                |
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
| 0a86a985-e781-40a9-9f71-4201d14ca99b | my_qos2 | front-end |             | read_iops_sec='100', write_iops_sec='100'
|
| 98b4aba0-b545-4524-a8d5-f150577b16b5 | my_qos  | front-end |             | read_iops_sec='100', write_iops_sec='100'
|
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
$ openstack volume qos set --property write_iops_sec=150 my_qos
$ openstack volume qos list
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
| ID                                   | Name    | Consumer  | Associations | Properties                                |
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
| 0a86a985-e781-40a9-9f71-4201d14ca99b | my_qos2 | front-end |             | read_iops_sec='100', write_iops_sec='100'
|
| 98b4aba0-b545-4524-a8d5-f150577b16b5 | my_qos  | front-end |             | read_iops_sec='100', write_iops_sec='150'
|
+--------------------------------------+---------+-----------+-------------+--------------------------------------
--+
```

Опция удаляется командой `unset`:

```
$ openstack volume qos unset --property write_iops_sec my_qos
$ openstack volume qos list
```

```
+--------------------------------------+---------+----------+-------------+--------------------------------------
--+
| ID                                   | Name    | Consumer | Associations | Properties                           |
+--------------------------------------+---------+----------+-------------+--------------------------------------
--+
| 0a86a985-e781-40a9-9f71-4201d14ca99b | my_qos2 | front-end |             | read_iops_sec='100', write_iops_sec='100'
|
| 98b4aba0-b545-4524-a8d5-f150577b16b5 | my_qos  | front-end |             | read_iops_sec='100'                  |
+--------------------------------------+---------+----------+-------------+--------------------------------------
--+
```

QoS может быть привязана к существующему типу диска, либо с ее использованием может быть создан новый тип диска. Предпочтительнее использование второго варианта. Создадим новый тип диска:

```
$ openstack volume type create qos_iops100
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| description | None                                 |
| id          | f16b20a0-6786-420b-ba5c-8eaf20bd60d2 |
| is_public   | True                                 |
| name        | qos_iops100                          |
+-------------+--------------------------------------+
```

Свяжем созданный тип диска qos_iops100 с бэкендом. Имя бэкенда доступно к просмотру в файле /etc/cinder/cinder.conf (если бэкендов несколько, то следует выбрать один из них).

```
$ grep enabled_backends /etc/cinder/cinder.conf
volume_backend_name = nfs
$ openstack volume type set --property volume_backend_name=nfs qos_iops100
$ openstack volume type list --long
+--------------------------------------+-------------+-----------+--------------------+------------------------------
----------------------+
| ID                                   | Name          | Is Public | Description           | Properties
|
+--------------------------------------+-------------+-----------+--------------------+------------------------------
----------------------+
| f16b20a0-6786-420b-ba5c-8eaf20bd60d2 | qos_iops100 | True      | None                  | volume_backend_name='nfs'
|
| 0c548032-f19d-4928-85a8-5c94be9a6be0 | glance_nfs  | False     | None                   | multiattach='<is> True',
volume_backend_name='nfs' |
| fe86726d-dede-43f5-a1d4-2d188a91773d | nfs         | True      | None                  | volume_backend_name='nfs'
|
|   a93daa57-2b72-4bed-9b5b-b58421498b6c   |   __DEFAULT__   |   True                |   Default   Volume   Type   |
|
+--------------------------------------+-------------+-----------+--------------------+------------------------------
----------------------+
```

Затем необходимо связать тип диска со спецификацией QoS, используя команду associate. В качестве первого параметра задается имя спецификации QoS, а второго — имя типа диска:

```
$ openstack volume qos list
+--------------------------------------+---------+----------+-------------+--------------------------------------
--+
| ID                                   | Name    | Consumer | Associations | Properties                           |
+--------------------------------------+---------+----------+-------------+--------------------------------------
--+
| 0a86a985-e781-40a9-9f71-4201d14ca99b | my_qos2 | front-end |             | read_iops_sec='100', write_iops_sec='100'
|
| 98b4aba0-b545-4524-a8d5-f150577b16b5 | my_qos  | front-end |             | read_iops_sec='100'                  |
+--------------------------------------+---------+----------+-------------+--------------------------------------
--+
$ openstack volume type list
+--------------------------------------+-------------+-----------+
| ID                                   | Name        | Is Public |
+--------------------------------------+-------------+-----------+
| f16b20a0-6786-420b-ba5c-8eaf20bd60d2 | qos_iops100 | True      |
| a269b89c-ce1d-4d2d-bd17-828f93fe14ec | nfs2        | True      |
| 0c548032-f19d-4928-85a8-5c94be9a6be0 | glance_nfs  | False     |
| fe86726d-dede-43f5-a1d4-2d188a91773d | nfs         | True      |
+--------------------------------------+-------------+-----------+
```

```
$ openstack volume qos associate my_qos qos_iops100
```

Опция `list` позволяет просмотреть типы диска, привязанные к QoS (столбец **Associations):**

```
$ openstack volume qos list
+--------------------------------------+---------+-----------+--------------+---------------------------------------
--+
| ID                                   | Name    | Consumer  | Associations | Properties                                |
+--------------------------------------+---------+-----------+--------------+---------------------------------------
--+
| 0a86a985-e781-40a9-9f71-4201d14ca99b | my_qos2 | front-end |              | read_iops_sec='100', write_iops_sec='100'
|
| 98b4aba0-b545-4524-a8d5-f150577b16b5 | my_qos  | front-end | qos_iops100  | read_iops_sec='100'                       |
+--------------------------------------+---------+-----------+--------------+---------------------------------------
--+
```

Отключить QoS для конкретного типа диска можно командой `disassociate:`

```
$ openstack volume qos disassociate --volume-type qos_iops100 my_qos
```

Если QoS привязан к нескольким типам дисков, то с помощью ключа `--all` можно отключить все типы дисков, использующие данный QoS:

```
$ openstack volume qos disassociate --all my_qos
```

5.4.4. Удаление

Для удаления спецификации QoS необходимо сначала отключить её ото всех типов диска, и после этого выполнить команду `delete:`

```
$ openstack volume qos disassociate --all my_qos
$ openstack volume qos delete my_qos
```

5.5. Операции с дисками

5.5.1. Конвертация типа диска и миграция дисков между бэкендами

Конвертация и миграция не работают для диска, подключенного к выключенной виртуальной машине.

Если к типу диска привязана спецификация QoS, то конвертировать в этот тип диска или из этого типа диска можно только тогда, когда исходный диск имеет статус *Доступен*. Поэтому если он подключен к ВМ, сначала его необходимо отключить. Отключение загрузочного диска приведено в разделе "Отключение и подключение основного загрузочного диска ВМ".

5.5.1.1. Конвертация без миграции и с миграцией

Возможны два варианта конвертации в зависимости от привязки типов диска к бэкендам.

Если типы диска привязаны к одному бэкенду, то при конвертации между этими типами сам файл диска физически останется расположен на прежнем бэкенде. В следующем примере типы nfs и qos_iops100 привязаны к одному бэкенду — nfs:

```
$ openstack volume type list --long
```

```
      +--------------------------------------+-------------+-----------+--------------------+-----------------------------
----------------------+
      | ID                                   | Name          | Is Public | Description           | Properties
|
      +--------------------------------------+-------------+-----------+--------------------+-----------------------------
----------------------+
      | ced2c227-1b31-4ff9-b16d-8d99d8a14ef1 | qos_iops100 | True       | None                  | volume_backend_name='nfs'
|
      | 01f7403c-cc00-4271-ad12-c50043663936 | glance_nfs   | False      | None                   | multiattach='<is> True',
volume_backend_name='nfs' |
      | e80579d1-9eb2-4c0f-8375-95f200448e66 | nfs          | True       | None                  | volume_backend_name='nfs'
|
      |   a93daa57-2b72-4bed-9b5b-b58421498b6c   |   __DEFAULT__   |   True                  |   Default   Volume   Type   |
|
      +--------------------------------------+-------------+-----------+--------------------+-----------------------------
----------------------+
```

Если типы диска привязаны к разным бэкендам, то при конвертации диск мигрирует на другой бэкенд. В следующем примере типы nfs и nfs_extra привязаны к разным бэкендам — nfs и nfs2 соответственно:

```
      $ openstack volume type list --long
      +--------------------------------------+-------------+-----------+--------------------+-----------------------------
----------------------+
      | ID                                   | Name          | Is Public | Description           | Properties
|
      +--------------------------------------+-------------+-----------+--------------------+-----------------------------
----------------------+
      | 4c27a9b9-8a0e-491e-bbb6-39e5683aae2a | nfs_extra    | True       | None                 | volume_backend_name='nfs2'
|
      | 01f7403c-cc00-4271-ad12-c50043663936 | glance_nfs   | False      | None                   | multiattach='<is> True',
volume_backend_name='nfs' |
      | e80579d1-9eb2-4c0f-8375-95f200448e66 | nfs          | True       | None                  | volume_backend_name='nfs'
|
      |   a93daa57-2b72-4bed-9b5b-b58421498b6c   |   __DEFAULT__   |   True                  |   Default   Volume   Type   |
|
      +--------------------------------------+-------------+-----------+--------------------+-----------------------------
----------------------+
```

Механизм конвертации и миграции для этих вариантов одинаков. Требуются имя или ID диска, который нужно конвертировать.

```
      $ openstack volume list --long
      +--------------------------------------+---------------+--------+------+------+----------+-------------------------
-----+--------------------+
      | ID                                   | Name          | Status | Size | Type | Bootable | Attached to                |
Properties         |
      +--------------------------------------+---------------+--------+------+------+----------+-------------------------
-----+--------------------+
      | 29c14074-03c5-4313-8b07-4168f9dec3b3 | Disk for test1 | in-use |    1 | nfs  | true     | Attached to test1 on /dev/vda
| attached_mode='rw' |
      +--------------------------------------+---------------+--------+------+------+----------+-------------------------
-----+--------------------+
```

Для конвертации между разными типами дисков используется следующая команда:

```
    $ openstack volume set --retype-policy on-demand --type nfs_extra "Disk for
test1"
```

Опцией `type` задаётся тип диска, в который конвертируется диск. Последнее значение без опции — это имя исходного диска. Вместо имени возможно использование ID диска.

Спустя некоторое время, в зависимости от размера диска, проверяется успешность миграции и конвертации диска в новый тип:

```
    $ openstack volume show "Disk for test1" -c migration_status -c type
    +------------------+-----------+
    | Field            | Value     |
    +------------------+-----------+
    | migration_status | success   |
```

```
| type              | nfs_extra |
+-------------------+-----------+
```

Следует учитывать, что после конвертации ID диска в имени файла на хранилище изменится. Для его просмотра используется команда:

```
$ openstack volume show "Disk for test1" -c os-vol-mig-status-attr:name_id
+---------------------------------+--------------------------------------+
| Field                           | Value                                |
+---------------------------------+--------------------------------------+
| os-vol-mig-status-attr:name_id | 54599370-b693-4ac8-91b8-387c91803996 |
+---------------------------------+--------------------------------------+
```

5.5.1.2. Миграция без конвертации

Миграция без конвертации между бэкендами возможна только в том случае, когда к одному типу диска привязано несколько бэкендов.

Просмотр имени или ID диска определяется командой:

```
$ openstack volume list
+--------------------------------------+---------------+--------+------+--------------------------------+
| ID                                   | Name          | Status | Size | Attached to                    |
+--------------------------------------+---------------+--------+------+--------------------------------+
| 1fc383bf-a5a5-4fd3-b250-931f3c0a9385 | Disk for test1 | in-use |    1 | Attached to test1 on /dev/vda  |
+--------------------------------------+---------------+--------+------+--------------------------------+
```

Просмотр хранилища, в котором расположен мигрируемый диск:

```
$ openstack volume show "Disk for test1" -c os-vol-host-attr:host
+-----------------------+-----------------+
| Field                 | Value           |
+-----------------------+-----------------+
| os-vol-host-attr:host | node01@nfs2#nfs |
+-----------------------+-----------------+
```

Диск расположен на бэкенде node01@nfs2 (символы, начиная с решётки, не учитываются). Имя второго бэкенда, на который может выполняться миграция диска, указано в столбце Host - это node01@nfs:

```
$ openstack volume service list --service cinder-volume -c Binary -c Host
+---------------+-------------+
| Binary        | Host        |
+---------------+-------------+
| cinder-volume | node01@nfs2 |
| cinder-volume | node01@nfs  |
+---------------+-------------+
```

Следующей командой выполняется миграция:

```
$ openstack volume migrate --host node01@nfs "Disk for test1"
```

Опция `host` задаёт имя бэкенда, на который планируется мигрировать диск. Последнее значение без опции – это имя исходного диска. Вместо имени можно использовать ID диска.

Спустя некоторое время, в зависимости от размера диска, необходимо проверить успешность миграции диска на бэкенд node01@nfs:

```
$ openstack volume show "Disk for test1" -c migration_status -c os-vol-host-attr:host
+----------------------+---------------+
| Field                | Value         |
+----------------------+---------------+
| migration_status     | success       |
| os-vol-host-attr:host | node01@nfs#nfs |
+----------------------+---------------+
```

Следует учитывать, что после миграции ID диска, указываемое в имени файла на хранилище, изменится. Оно доступно к просмотру после выполнения команды:

```
$ openstack volume show "Disk for test1" -c os-vol-mig-status-attr:name_id
+--------------------------------+--------------------------------------+
| Field                          | Value                                |
+--------------------------------+--------------------------------------+
| os-vol-mig-status-attr:name_id | c5433157-3360-49e0-b194-ec1215ed6839 |
+--------------------------------+--------------------------------------+
```

5.5.1.3. Подключение к ВМ блочного устройства из ОС гипервизора

Подключение напрямую блочного устройства из ОС гипервизора в настоящий момент доступно через CLI. Для начала необходимо установить: на каком физическом узле запущена и какое служебное имя (id) в гипервизоре присвоено ВМ. Эти сведения прописаны в параметрах **OS-EXT-SRV-ATTR:hypervisor_hostname** и **id** для каждой ВМ. Также можно руководствоваться столбцами **ID** и **Физический узел** в портале.

ID	Физический узел	Имя	Конфигурация	vCPU	RAM, ГБ	HDD, ГБ	IP	Статус
575c3ebf-d43d-47a7-b936-98fb72f...	aio59.node.test.com	cirros	tiny	1	0.5	1	192.168.0.254	Запущен
67fd79e0-762b-4950-b50c-87fcf87...	comp61.node.test.com	test_ubunt...	small	2	1	6	10.11.16.142	Запущен
d50cea27-5dbb-4b02-bbbb-c6aff99...	comp60.node.test.com	ubuntu	small	2	1	10	192.168.0.234	Запущен

Столбцы **ID** и **Физический узел** л могут быть скрыты в портале.

Далее необходимо подключиться к CLI на физический узел, на котором запущена ВМ, и подключить локальное блочное устройство к ВМ. Если блочное устройство находится на другом узле гипервизора, то ВМ предварительно нужно смигрировать.

Диски, подключенные способами ниже, автоматически отключаются при выключении ВМ. Также невозможна миграция ВМ с подключенными локальными дисками.

5.5.1.4. Пример подключения PCI устройства

Для примера воспользуемся CL. Определим, на каком физическом узле работает ВМ и какое имя на гипервизоре она имеет. Для этого выполним на любом из узлов платформы команду:

```
    ohost24 # openstack --os-cloud rustack_system server show test-server -c OS-EXT-
SRV-ATTR:hypervisor_hostname -c id
    +-------------------------------------+--------------------------------------+
    | Field                               | Value                                |
    +-------------------------------------+--------------------------------------+
    | OS-EXT-SRV-ATTR:hypervisor_hostname | ohost1.node.rustack.example.com      |
    | id                                  | 2bdf9fcb-7966-4e7c-b437-d3246ecc3f79 |
    +-------------------------------------+--------------------------------------+
```

Далее подключимся по ssh на узел с именем ohost1.node.rustack.example.com и посмотрим на доступные блочные устройства:

```
    ohost1 ~ # lsblk
    NAME                                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
    sda                                   8:0    0  1.8T  0 disk
    ├─sda1                                8:1    0    2M  0 part
    ├─sda2                                8:2    0  512M  0 part  /boot
    └─sda3                                8:3    0  1.8T  0 part  /
    sdb                                   8:16   0  1.8T  0 disk
```

Подключим диск /dev/sdb с узла гипервизора к ВМ с id 2bdf9fcb-7966-4e7c-b437-d3246ecc3f79 с именем vda:

```
    ohost1 ~ # virsh attach-disk 2bdf9fcb-7966-4e7c-b437-d3246ecc3f79 /dev/sdb vda
    Disk attached successfully
```

Для отключения диска от ВМ можно воспользоваться командой:

```
    ohost1 ~ # virsh detach-disk 2bdf9fcb-7966-4e7c-b437-d3246ecc3f79 /dev/sdb
    Disk detached successfully
```

Для прозрачного обнаружения диска операционной системой Linux внутри ВМ в параметрах образа, из которого запущена ВМ нужно выбрать Дисковый контроллер **virtio-scsi**

5.5.1.5. Пример подключения USB-устройства

Следующий пример демонстрирует, как подключать к ВМ USB-устройства, например, USB-Flash-накопители. Устройство должно быть предварительно подключено к физической машине узла виртуализации.

Найдем USB-устройство, которое необходимо подключить:

```
    ohost1 ~ # lsusb -v | egrep -i 'idvendor|idproduct'
      idVendor           0x04ca Lite-On Technology Corp.
      idProduct          0x706d
```

Создадим XML-файл my_usb.xml со следующей структурой:

```
    <hostdev mode='subsystem' type='usb' managed='yes'>
      <source>
        <vendor id='0x04ca'/>
        <product id='0x706d'/>
      </source>
    </hostdev>
```

где параметры vendor id и product id нужно заменить на значения из вывода команды lsusb.

Подключим устройство к ВМ, используя конфигурационный файл my_usb.xml:

```
    ohost1 ~ # virsh attach-device 2bdf9fcb-7966-4e7c-b437-d3246ecc3f79 --file
my_usb.xml
```

Если при подсоединении устройства выводится ошибка, наподобие:

```
    error: Failed to attach device from my_usb.xml
    error: internal error: unable to execute QEMU command 'device_add': failed to
open /dev/bus/usb/001/002: Permission denied
```

необходимо дать права на устройство пользователю nova и повторить команду подключения

```
    ohost1 ~ # chown nova /dev/bus/usb/001/002
    ohost1 ~ # virsh attach-device 2bdf9fcb-7966-4e7c-b437-d3246ecc3f79 --file
my_usb.xml
```

5.5.2. Отключение и подключение основного загрузочного диска ВМ

Следует иметь ввиду, что не рекомендуется вносить изменения в базу данных.

Возможно возникновение необходимости отключения основного диска от ВМ (`/dev/hda`, `/dev/vda/`, `/dev/sda` — в зависимости от типа контроллера). Например, если к типу диска привязана спецификация QoS, то конвертировать в этот тип диска или из этого типа диска можно только тогда, когда исходный диск имеет статус *Доступен*. Поэтому если он подключен к ВМ, сначала нужно отключить его.

Если при создании ВМ был установлен флажок в чекбоксе **Удалять вместе с ВМ**, и после этого был отключен и снова подключен основной загрузочный диск, этот чекбокс станет неактуален, то есть после удаления ВМ загрузочный диск не будет удалён.

Например, в качестве дискового контроллера используется VirtIO, поэтому дисковые устройства имеют имена в формате /dev/vdX.

Для успешного отключения диска от ВМ в базах данных Cinder и Nova нужно изменить имя устройства на отличное от /dev/vda, а также в базе данных Nova изменить boot_index на 1.

Сначала необходимо выключить ВМ в консоли командой `sudo poweroff`. Командой `volume list` определяется ID основного загрузочного диска, который нужно отключить (значение в столбце Bootable для такого диска — true).

```
$ openstack volume list --long
+--------------------------------------+----------------+--------+------+------+----------+--------------------------------+--------------------+
| ID                                   | Name           | Status | Size | Type | Bootable | Attached to                    | Properties         |
+--------------------------------------+----------------+--------+------+------+----------+--------------------------------+--------------------+
| f43428f0-ac0d-48c9-b276-2ef077c0ff51 | volume2        | in-use |   20 | nfs  | false    | Attached to test1 on /dev/vdb  | independent='true' |
| 2c4d44b4-4886-4426-96ba-45e203e3e218 | Disk for test1 | in-use |   10 | nfs  | true     | Attached to test1 on /dev/vda  | attached_mode='rw' |
+--------------------------------------+----------------+--------+------+------+----------+--------------------------------+--------------------+
```

Далее следует внести изменения в базе данных сервиса управления дисками (Cinder), подставляя полученный ID из предыдущего вывода команды:

```
$ psql -h postgresql.service.`hostname -d | cut -d'.' -f2,3` -U postgres -d cinder
cinder=# SELECT * FROM volume_attachment WHERE volume_id = '2c4d44b4-4886-4426-96ba-45e203e3e218';
cinder=# UPDATE volume_attachment SET mountpoint = '/dev/vdx' WHERE volume_id = '2c4d44b4-4886-4426-96ba-45e203e3e218';
UPDATE 1
```

и в базе данных сервиса управления ВМ (Nova):

```
psql -h postgresql.service.`hostname -d | cut -d'.' -f2,3` -U postgres -d nova
nova=# SELECT * FROM block_device_mapping WHERE volume_id = '2c4d44b4-4886-4426-96ba-45e203e3e218';
nova=# UPDATE block_device_mapping SET device_name = '/dev/vdx', boot_index = '1' WHERE volume_id = '2c4d44b4-4886-4426-96ba-45e203e3e218';
UPDATE 1
```

Затем отключается диск от ВМ командой `server remove volume` и проверяется успешность отключения диска от ВМ.

```
$ openstack server remove volume test1 "Disk for test1"
$ openstack volume show "Disk for test1" -c attachments
+-------------+-------+
| Field       | Value |
+-------------+-------+
| attachments | []    |
+-------------+-------+
```

После этого возможно осуществить требуемые операции с диском (например, конвертирование типа диска).

После этого диск подключается к ВМ командой `server add volume` и проверяется успешность подключения диска к ВМ.

```
$ openstack server add volume test1 "Disk for test1"
$ openstack volume show "Disk for test1" -c attachments
+-------------+-----------------------------------------------------------------------------------------------------------------------------------------------
------
| Field       | Value
+-------------+-----------------------------------------------------------------------------------------------------------------------------------------------
------
| attachments | [{'id': '2c4d44b4-4886-4426-96ba-45e203e3e218', 'attachment_id': 'fef68145-349a-45a6-ad93-6b254982d54a',
                'volume_id': '2c4d44b4-4886-4426-96ba-45e203e3e218', 'server_id': 'c9fdeae5-2e35-46e2-9913-e4ab01424041',
|             | 'host_name': 'node03', 'device': '/dev/vdc', 'attached_at': '2021-02-17T23:21:13.354792'}]
+-------------+-----------------------------------------------------------------------------------------------------------------------------------------------
------
```

Выполнение обратных изменений в базе данных Cinder:

```
$ psql -h postgresql.service.`hostname -d | cut -d'.' -f2,3` -U postgres -d cinder
cinder=# UPDATE volume_attachment SET mountpoint = '/dev/vda' WHERE volume_id = '2c4d44b4-4886-4426-96ba-45e203e3e218' and deleted = 'f';
UPDATE 1
```

Выполнение обратных изменений в базе данных Nova:

```
$ psql -h postgresql.service.`hostname -d | cut -d'.' -f2,3` -U postgres -d nova
nova=# UPDATE block_device_mapping SET device_name = '/dev/vda', boot_index = '0' WHERE volume_id = '2c4d44b4-4886-4426-96ba-45e203e3e218' AND deleted = '0';
UPDATE 1
```

Запуск ВМ выполняется следующей командой:

```
$ openstack server reboot --hard test1
```

Проверка успешности запуска ВМ:

```
$ openstack server show test1 -c status
+--------+--------+
| Field  | Value  |
+--------+--------+
| status | ACTIVE |
+--------+--------+
```

6. Сеть

6.1. Компоненты сетевой подсистемы

Компоненты сетевой подсистемы ПВ РУСТЭК (OpenStack Neutron) настраиваются РУСТЭК.Конфигуратором при развертывании платформы. Для установки основного компонента в параметрах узла должна быть включена роль «Управление сетями»: она включается автоматически при выборе профилей Основной, Дополнительный при добавлении узла. Дополнительные компоненты устанавливаются при включении роли «Физический узел» — эта роль включается при использовании профилей Арбитр и Вычислительный узел.

Компонент(служба)	Описание	Роль узла при конфигурации
ovsdb-server	ВМ базы данных, осуществляющий хранение конфигурации и настроек Open vSwitch.	Управление сетями Физический узел
ovs-vswitchd	Демон, осуществляющий настройку и контроль коммутаторов Open vSwitch в ОС узла.	Управление сетями Физический узел
neutron-openvswitch-agent	Агент, осуществляющий настройку сетей для виртуальных машин.	Управление сетями Физический узел
neutron-dhcp-agent	Агент, предоставляющий службы DHCP для подсетей пользователей. Для каждой подсети, в которой назначен DHCP-сервер, агент создает пространство имён вида: qdhcp-<NET-UUID>, в котором отдельный процесс dnsmasq выделяет IP-адреса.	Управление сетями
neutron-l3-agent	Агент, реализующий маршрутизацию и NAT посредством iptables и сетевого стека операционной системы. Изоляция маршрутизируемых сетей осуществляется с помощью механизма ip namespaces.	Управление сетями Физический узел
neutron-metadata-agent	Агент, предоставляющий метаданные (hostname, ssh-ключ и т.д.) виртуальным машинам с адреса http://169.254.169.254 при помощи пространства имён роутера или DHCP-сервера.	Управление сетями Физический узел
neutron-bgp-dragent	Агент динамической маршрутизации BGP, позволяющий объявлять префиксы виртуальных сетей пользователей физическим сетевым устройствам, которые поддерживают протоколы динамической маршрутизации.	Управление сетями
neutron-server	Демон, управляющий пользовательскими запросами и предоставляющий API-интерфейс.	Управление сетями
neutron-ovs-cleanup	Служба, удаляющая порты ВМ и других сущностей из базы Open vSwitch. Запускается при загрузке операционной системы узла.	Управление сетями Физический узел

Как и для большинства сервисов ПВ РУСТЭК, для функционирования сетевых служб Neutron требуются:

- доступ к экземпляру БД, который работает или на одном, или на нескольких (в режиме HA) узлах;
- доступ к работающей службе сообщений — BM Rabbitmq;
- доступ к BM кэширования Redis;
- доступ к службе идентификации — Keystone, а также соответствующие записи: user, role, и endpoint.

6.2. Потоки трафика

В разделе описывается поток сетевого трафика для самого простого типа сегментации — VLAN, в трёх распространенных сценариях:

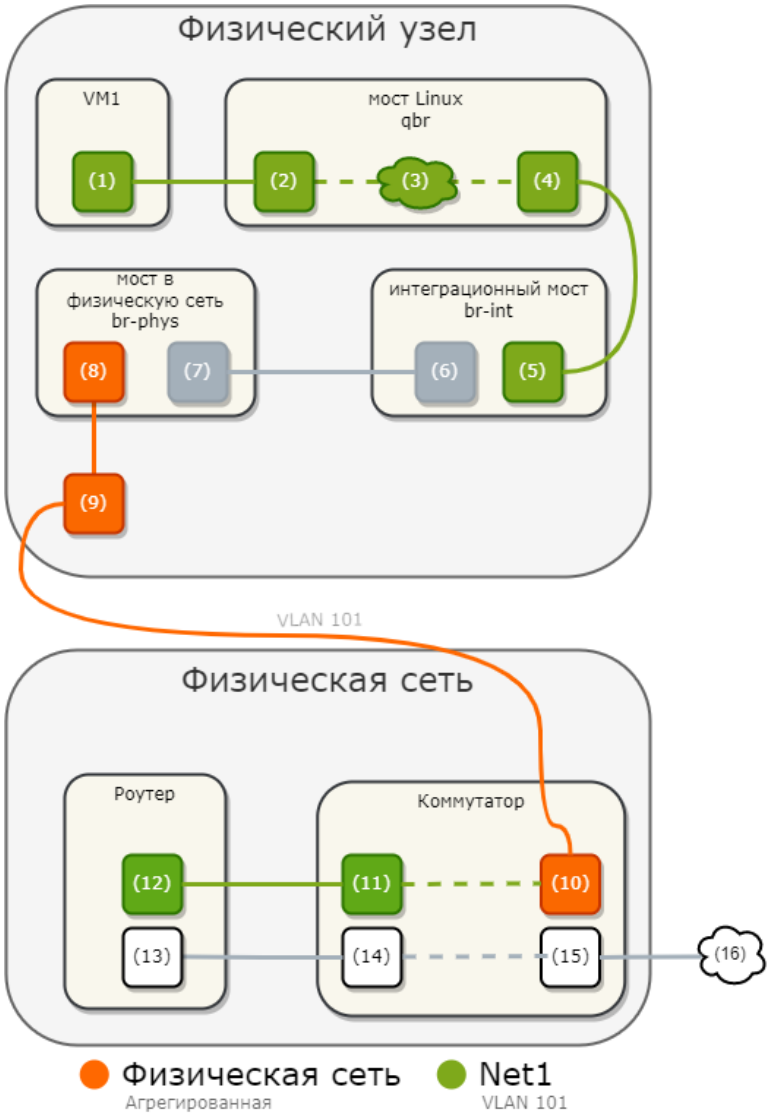
- «Север-Юг» — между виртуальными машинами и сетями в ПВ РУСТЭК и сетями внешнего мира;
- «Восток-Запад»-1 — между виртуальными машинами на разных узлах в одной сети;
- «Восток-Запад»-2 — между виртуальными машинами на одном узле в разных сетях.

Примеры используют следующую конфигурацию:

- виртуальная машина VM1;
- виртуальная машина VM2;
- внутренняя сеть Net1 (VLANID 101);
- внутренняя сеть Net2 (VLANID 102);
- физическая сеть (trunk VLANIDs 101-102).

6.2.1. Сценарий «Север-Юг»

Виртуальная машина VM1 отправляет пакет в Интернет.



Путь пакета на физическом узле:

- интерфейс VM1 (1) пересылает пакет на порт моста Linux (2) через виртуальные сетевые интерфейсы `veth`;
- пакет проходит фильтрацию правил группы безопасности (3) в мосте Linux;
- порт OVS моста Linux (4) перенаправляет пакет на порт интеграционного моста OVS (5) через `veth`;
- мост интеграции OVS добавляет к пакету внутренний тег VLAN;
- порт `int-br-phys` моста интеграции OVS (6) перенаправляет пакет на порт `phy-br-phys` моста OVS в физическую сеть (7);
- мост OVS в физическую сеть меняет внутренний тег VLAN на фактический тег VLAN 101;
- сетевой порт моста в физическую сеть OVS (8) перенаправляет пакет на физический сетевой интерфейс (9).

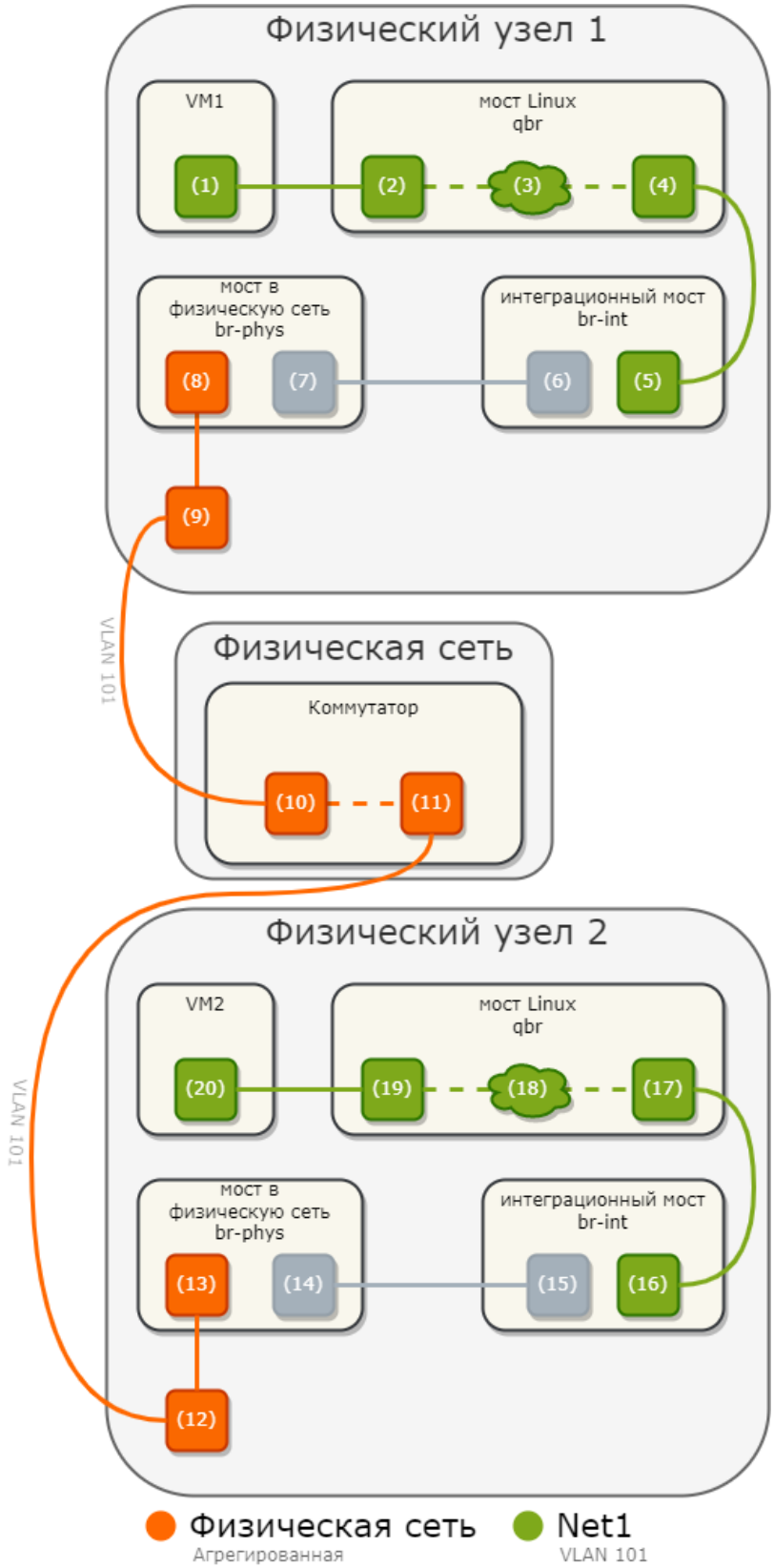
Путь пакета в физической сетевой инфраструктуре:

- физический сетевой интерфейс пересылает пакет на коммутатор (10);
- коммутатор удаляет тег VLAN 101 из пакета и пересылает его на маршрутизатор (11);
- маршрутизатор перенаправляет пакет из физической сети (12) во внешнюю сеть (13) и отправляет пакет на коммутатор (14);
- коммутатор пересылает пакет в вышестоящую сеть (15);
- пакет уходит в Интернет (16) .

Ответный трафик проходит в обратной последовательности.

6.2.2. Сценарий «Восток-Запад»-1: ВМ в одной сети

Виртуальная машина VM1 отправляет пакет машине VM2.



Путь пакета на физическом узле 1:

- интерфейс VM1 (1) пересылает пакет на порт моста Linux (2) через виртуальные сетевые интерфейсы `veth`;
- пакет проходит фильтрацию правил группы безопасности (3) в мосте Linux;
- порт OVS моста Linux (4) перенаправляет пакет на порт интеграционного моста OVS (5) через `veth`;
- мост интеграции OVS добавляет к пакету внутренний тег VLAN;

- порт `int-br-phys` моста интеграции OVS (6) перенаправляет пакет на порт `phy-br-phys` моста OVS в физическую сеть (7);
- мост OVS в физическую сеть меняет внутренний тег VLAN на фактический тег VLAN 101;
- сетевой порт моста в физическую сеть OVS (8) перенаправляет пакет на физический сетевой интерфейс (9).

Путь пакета в физической сетевой инфраструктуре:

- физический сетевой интерфейс пересылает пакет на коммутатор (10);
- коммутатор пересылает пакет с физического узла 1 на физический узел 2 (11).

Путь пакета на физическом узле 2:

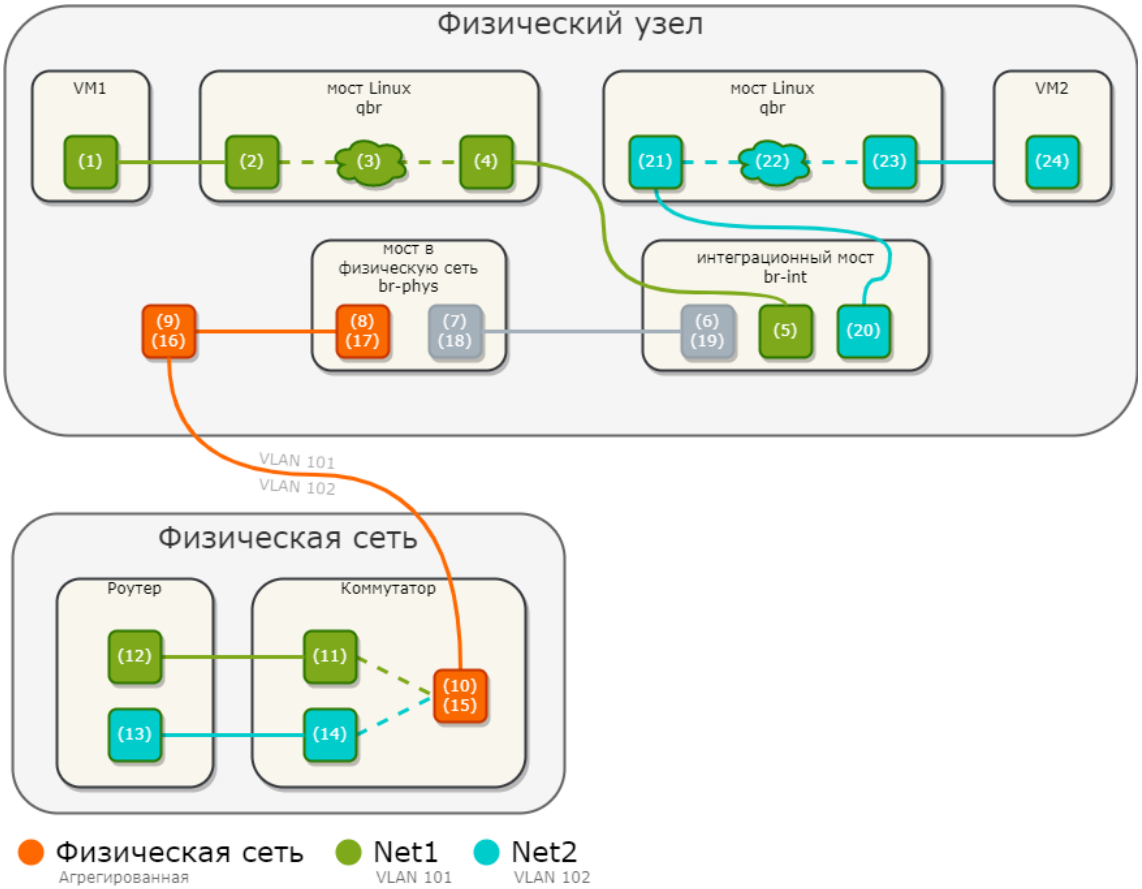
- физический сетевой интерфейс (12) пересылает пакет на сетевой порт моста в физическую сеть OVS (13);
- порт `phy-br-phys` моста провайдера OVS (14) перенаправляет пакет на порт `int-br-phys` моста интеграции OVS (15);
- мост интеграции OVS меняет фактический тег VLAN 101 на внутренний тег VLAN;
- порт группы безопасности моста интеграции OVS (16) перенаправляет пакет на порт OVS моста интеграции группы безопасности (17);
- пакет проходит фильтрацию правил группы безопасности в мосте Linux (18);
- порт ВМ моста Linux (19) пересылает пакет на интерфейс VM2 (20) через `veth`.

Ответный трафик проходит в обратной последовательности.

6.2.3. Сценарий «Восток-Запад»-2: ВМ в разных сетях

Виртуальная машина VM1 отправляет пакет машине VM2 через маршрутизатор в физической сети.

В примере обе ВМ находятся на одном и том же физическом узле. При этом показано, как тегирование VLAN позволяет нескольким логическим сетям использовать одну и ту же физическую сеть уровня 2.



Путь пакета на физическом узле:

1. интерфейс VM1 (1) пересылает пакет на порт моста Linux (2) через виртуальные сетевые интерфейсы `veth`;
2. пакет проходит фильтрацию правил группы безопасности (3) в мосте Linux;
3. порт OVS моста Linux (4) перенаправляет пакет на порт интеграционного моста OVS (5) через `veth`;
4. мост интеграции OVS добавляет к пакету внутренний тег VLAN;
5. порт `int-br-phys` моста интеграции OVS (6) перенаправляет пакет на порт `phy-br-phys` моста OVS в физическую сеть (7);
6. мост OVS в физическую сеть меняет внутренний тег VLAN на фактический тег VLAN 101;
7. сетевой порт моста в физическую сеть OVS (8) перенаправляет пакет на физический сетевой интерфейс (9).

Путь пакета в физической сетевой инфраструктуре:

- физический сетевой интерфейс пересылает пакет на коммутатор (10);
- коммутатор удаляет тег VLAN 101 из пакета и пересылает его маршрутизатору (11);
- маршрутизатор направляет пакет из сети Net1 (12) в сеть Net2 (13);
- маршрутизатор пересылает пакет коммутатору (14);
- коммутатор добавляет к пакету тег VLAN 102 и пересылает его на физический узел (15).

Путь пакета на физическом узле:

- физический сетевой интерфейс (16) пересылает пакет на сетевой порт моста OVS в физическую сеть (17);
- порт `phy-br-phys` моста провайдера OVS (18) перенаправляет пакет на порт исправления `phy-br-phys` моста интеграции OVS (19);
- мост интеграции OVS меняет фактический тег VLAN 102 на внутренний тег VLAN;
- порт моста интеграции OVS (20) удаляет внутренний тег VLAN и перенаправляет пакет на порт OVS моста Linux (21);
- пакет проходит фильтрацию правил группы безопасности (22) в мосте Linux;
- порт ВМ моста Linux (23) пересылает пакет на интерфейс VM2 (24) через veth.

6.3. Сегментация

Сегментация — это разделение одной физической сети на множество сегментов. В ПВ РУСТЭК под этим термином понимается в первую очередь создание виртуальных сетей на уровне платформы, в которые подключаются сетевые устройства, например, порты ВМ и роутеров. ПВ РУСТЭК поддерживает два типа сегментации: с использованием `VLAN` и через оверлейные сети.

6.3.1. VLAN (Virtual Local Area Network)

VLAN — это простейшая технология сегментации, которая позволяет разделять физическую L2-сеть на несколько виртуальных сетей, каждая из которых работает в отдельном broadcast-домене. Сетевые интерфейсы, настроенные на использование VLAN, добавляют в заголовок Ethernet-фрейма дополнительную информацию, содержащую тег VLAN — номер в диапазоне 1-4096, который используется для определения границ широковещательного домена и разрешает или запрещает проход фрейма через интерфейсы.

Диапазон по умолчанию: 100:200.

Максимальный диапазон: 1:4094.

> Для использования этого типа сегментации весь указанный диапазон `VLAN` должен присутствовать в физической сети. Проконсультируйтесь о доступных для использования значениях с сетевым администратором.

6.3.2. Оверлейные сети

Оверлейные сети, в отличие от `VLAN`, используют инкапсуляцию и технологии виртуальных сетевых интерфейсов (`VNI`) для создания логических сегментов поверх физической сети. Это более

современные типы сегментации, которые позволяют создавать гораздо большее количество виртуальных сетевых сегментов без необходимости поддержки и настройки со стороны физической сети. Из недостатков сетей такого типа можно отметить дополнительную нагрузку на CPU и уменьшение MTU при инкапсуляции.

6.3.2.1. VxLAN (Virtual eXtensible Local Area Network)

`Технология VxLAN` была создана как более гибкая альтернатива `VLAN`. Она позволяет создавать виртуальные сети с более высокими масштабируемостью и производительностью: `VxLAN` поддерживает до 16 миллионов виртуальных сетей, в отличие от 4096 у `VLAN`. Это технология туннелирования, которая работает поверх третьего уровня существующей физической сети и использует её в качестве UDP-транспорта для своих пакетов.

Диапазон по умолчанию: `1:65000`.

Максимальный диапазон: `1:224`.

6.3.2.2. GRE (Generic Routing Encapsulation)

GRE — это протокол создания туннелированных соединений. Обычно `GRE` используется для создания виртуальных частных сетей (VPN), но в ПВ РУСТЭК на его базе основан один из способов создания виртуальных сетей для использования ВМ.

Диапазон по умолчанию: `1:65000`.

Максимальный диапазон: `1:224`.

6.3.2.3. GENEVE (Generic Network Virtualization Encapsulation)

`Geneve` — это современный протокол сегментации, который был разработан для замены `VxLAN` и `GRE`. Отличается большей масштабируемостью, гибкостью и поддержкой большего количества сетевых функций.

Диапазон по умолчанию: `1:65000`.

Максимальный диапазон: `1:224`.

6.3.3. Сегментация в ПВ РУСТЭК

Тип сегментации по умолчанию — Geneve.

Диапазоны сегментов, доступных для создания сетей, настраиваются через РУСТЭК.Конфигуратор при установке, а тип сегментации выбирается для каждой новой сети при создании. Сети с разным типом сегментации можно связать между собой на сетевом уровне модели OSI через роутер.

При создании сети можно указать только её имя, а остальные параметры подставятся автоматически и будут использовать значения по умолчанию: например, в качестве типа сегментации будет выбран первый тип со свободными пулами из перечисленных в параметре `tenant_network_types` конфигурационного файла `/etc/neutron/plugin.ini`.

Конкретный сегмент также автоматически выбирается из пула, указанного в параметре `<segmentation_type>_ranges` в конфигурационном файле `/etc/neutron/plugin.ini` для выбранного типа сегментации. Например, для `VLAN`:

```
[ml2_type_vlan]
network_vlan_ranges = default:2352:2360
```

будет выбран первый свободный `VLANID` из диапазона `2352:2360`.

Администратор при создании сети может использовать `VLANID`, не относящийся к пулу по умолчанию. Например, в инсталляции из примера выше администратор может создать сеть с `VLANID` `1000`, который не входит в пул `2352:2360`.

6.4. Сети

Сети в ПВ РУСТЭК — это реализация соединений второго уровня модели OSI, но с дополнительной поддержкой `DHCP` и служб метаданных. В случае использования VLAN-сегментации, сети сопоставляются с существующими на физическом оборудовании сетями датацентра. При использовании оверлейных сетей сети второго уровня создаются в туннелях, построенных поверх третьего уровня физической сети. После создания сети в ней можно создать множество подсетей, реализующих подключения третьего уровня.

За создание и управление сетями в ПВ РУСТЭК отвечает Open vSwitch.

6.4.1. Open vSwitch

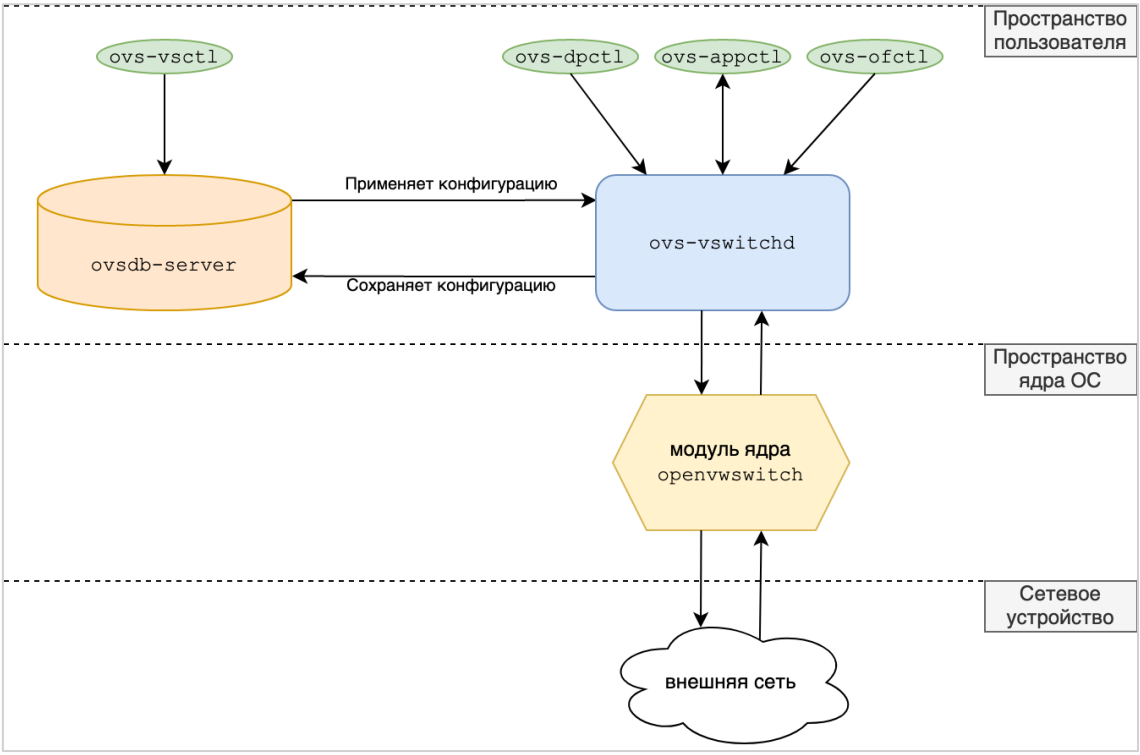
Open vSwitch (OVS) — это основа сетевой архитектуры ПВ РУСТЭК. Он представляет собой программно-определяемый коммутатор, аналогичный программному сетевому мосту Linux. OVS обеспечивает коммутацию виртуальных сетей и может интегрироваться с физическими коммутаторами с использованием функций уровня 2, таких как теги `STP`, `LACP` и `802.1Q VLAN`. Open vSwitch также поддерживает туннелирование `VxLAN`, `Geneve` и `GRE`.

6.4.1.1. Внутреннее устройство

Open vSwitch состоит из трёх основных компонентов:

- `openvswitch` — модуль ядра, эквивалентен `ASIC` на аппаратном коммутаторе: это плоскость данных (data plane) коммутатора, где происходит вся обработка пакетов;
- `ovs-vswitchd` — демон , который выполняется в пользовательском пространстве на каждом физическом хосте с ролями «Управление сетями» и «Физический узел» и определяет, как будет запрограммирован модуль ядра;
- `Open vSwitch Database Server (OVSDB)` — локальная база данных на каждом физическом сервере, которая хранит конфигурацию виртуальных коммутаторов на хосте.

Наглядно:



Сетевые устройства в концепции Open vSwitch, делятся на пять типов:

- tap-устройства;
- мосты Linux;
- виртуальные кабели (veth);
- мосты OVS;
- виртуальные патчкорды к OVS-портам.

Tap-интерфейсы создаются и используются гипервизором для подключения гостевых операционных систем к физическому узлу. Эти виртуальные интерфейсы соответствуют сетевым интерфейсам внутри ВМ. Кадр Ethernet, отправленный на tap-устройство на физическом узле, принимается гостевой операционной системой ВМ, а кадры, принятые из гостевой операционной системы ВМ, вводятся в сетевой стек физического узла.

Мост Linux — это виртуальный интерфейс (свитч), который соединяет несколько сетевых интерфейсов. В ПВ РУСТЭК мост обычно включает в себя физический интерфейс и один или несколько виртуальных или tap-интерфейсов.

Виртуальные кабели Virtual Ethernet (veth) представляют собой виртуальные интерфейсы, имитирующие сетевые соединительные кабели. ПВ РУСТЭК использует `veth-кабели` при подключении между сетевыми пространствами имен и мостами Linux, а также при подключении мостов Linux к коммутаторам Open vSwitch.

Мосты OVS — это виртуальные интерфейсы, аналог мостов Linux, но предоставляющие более широкий функционал, например, возможность подключения туннелированных соединений.

Виртуальные патчкорды OVS — это встроенный тип портов, который имитирует поведение виртуального кабеля `veth`, но оптимизирован для использования с мостами OVS. При подключении двух мостов OVS, порт на каждом коммутаторе зарезервирован как `patch-порт`. `Patch-порты` создаются с именем парного коммутатора, которое соответствует `patch-порту` на другом коммутаторе.

6.4.1.2. Open vSwitch на физических узлах

Типичная конфигурация в упрощённом виде выглядит так:

```
# ovs-vsctl show
cf28506f-8a1e-48c4-87d4-469bb0c7e8ff
```

```
        Manager "ptcp:6640:127.0.0.1"
        Bridge br-int
            fail_mode: secure
            Port br-int
                Interface br-int
                    type: internal
            Port "qvo9c6efcae-6e"
                tag: 9
                Interface "qvo9c6efcae-6e"
            Port int-br-phys
                Interface int-br-phys
                    type: patch
                    options: {peer=phy-br-phys}
        Bridge br-phys
            Port phy-br-phys
                Interface phy-br-phys
                    type: patch
                    options: {peer=int-br-phys}
            Port br-phys
                Interface br-phys
                    type: internal
            Port "bond0"
                Interface "bond0"
        ovs_version: "2.3.1"
```

В `OVS` создано два моста: `br-int` и `br-phys`. В `br-phys` подключен сетевой агрегат операционной системы `bond0` и виртуальный патчкорд `phy-br-phys` в мост `br-int`.

`br-int` — интеграционный мост, предназначенный для подключения виртуальных машин. В приведённой конфигурации к нему подключены три порта:

- `br-int` соответствует самому мосту,
- `int-br-phys` - соединение с мостом `br-phys`, приходящее в него интерфейсом `phy-br-phys`,
- `qvo9c6efcae-6e` — интерфейс подключения виртуальной машины.

Рассмотрим подключение сервера подробнее.

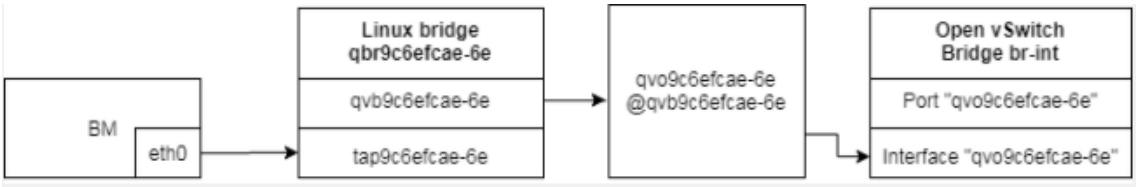
В порт `qvo9c6efcae-6e` подключен интерфейс `qvo9c6efcae-6e`, который вторым концом `qvb9c6efcae-6e` подключен в мост Linux `qbr9c6efcae-6e`. В этот же мост подключен интерфейс `tap9c6efcae-6e`, принадлежащий запущенной ВМ с ID `0cbd7a1d-1e75-4660-8152-90f09697b904`:

```
    # ip l | grep qvo9c6efcae
    40:  qvo9c6efcae-6e@qvb9c6efcae-6e:  <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP>
mtu 1500 qdisc noqueue master ovs-system state UP mode DEFAULT qlen 1000
    41:  qvb9c6efcae-6e@qvo9c6efcae-6e:  <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP>
mtu 1500 qdisc noqueue master qbr9c6efcae-6e state UP mode DEFAULT qlen 1000
    [root@compute ~]# brctl show | grep -A1 qvb9c6efcae
    qbr9c6efcae-6e          8000.4e25fe7b33a7       no              qvb9c6efcae-6e
                                                                    tap9c6efcae-6e
    # ip a | grep -A1 tap9c6efcae-6e
    42: tap9c6efcae-6e: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbr9c6efcae-6e state UNKNOWN qlen 1000
        link/ether fe:16:3e:78:3a:c6 brd ff:ff:ff:ff:ff:ff
```

Соответствие интерфейса конкретной ВМ определяется по идентификатору порта, подключенного к ВМ:

```
    # openstack port list --server 0cbd7a1d-1e75-4660-8152-90f09697b904 | egrep -i
"id|9c6efcae-6e"
    | ID                                   | Name | MAC Address       | Fixed IP Addresses
| Status |
    |  9c6efcae-6e1d-4fba-9e71-3076802f9758  |                   |  fa:16:3e:78:3a:c6  |
ip_address='192.168.88.124', subnet_id='a443bc05-b8b9-4771-a12b-0638bef401f5' | ACTIVE |
```

Схематично это выглядит так:



6.4.1.3. Open vSwitch на узлах с ролью «Управление сетями»

Open vSwitch на таких узлах имеет более сложную конфигурацию, чем на физических узлах:

```
    # ovs-vsctl show
    723e18de-a9d0-4ae5-9d93-ded702a93158
        Manager "ptcp:6640:127.0.0.1"
        Bridge br-phys
            Port phy-br-phys
                Interface phy-br-phys
                    type: patch
                    options: {peer=int-br-phys}
            Port br-phys
                Interface br-phys
                    type: internal
            Port "bond0"
                Interface "bond0"
        Bridge br-int
            fail_mode: secure
            Port int-br-phys
                Interface int-br-phys
                    type: patch
                    options: {peer=phy-br-phys}
            Port "qg-2ab5c9cf-05"
                tag: 3
                Interface "qg-2ab5c9cf-05"
                    type: internal
            Port "qr-ab5b4c29-7b"
                tag: 2
                Interface "qr-ab5b4c29-7b"
                    type: internal
            Port "ha-493f39a8-a6"
                tag: 4
                Interface "ha-493f39a8-a6"
                    type: internal
            Port br-int
                Interface br-int
                    type: internal
            Port "tap83171bc3-22"
                tag: 2
                Interface "tap83171bc3-22"
                    type: internal
        ovs_version: "2.3.1"
```

Здесь мы видим те же мосты `br-int` и `br-phys`, как и на физическом узле, причём конфигурация `br-phys` полностью идентична. Рассмотрим подробнее конфигурацию `br-int`.

Интерфейс `tap83171bc3-22` принадлежит `dnsmasq-серверу`, работающему в соответствующем пространстве имен. С этого интерфейса обслуживаются ВМ, принадлежащие сети из пространства имен. Для каждой подсети, в которой отмечена служба DHCP, запускается отдельный `dnsmasq-сервер`.

По имени этого интерфейса можно найти экземпляр процесса `dnsmasq` в системе и просмотреть его конфигурацию, выданные адреса аренды и другие параметры:

```
# ps aux | grep tap83171bc3-22
nobody   20916  0.0  0.0  15676  2004 ?        S    Jan11   0:02 dnsmasq --no-
hosts --no-resolv --strict-order --bind-interfaces --interface=tap83171bc3-22 --
except-interface=lo      --pid-file=/var/lib/neutron/dhcp/a443bc05-b8b9-4771-a12b-
0638bef401f5/pid   --dhcp-hostsfile=/var/lib/neutron/dhcp/a443bc05-b8b9-4771-a12b-
0638bef401f5/host      --addn-hosts=/var/lib/neutron/dhcp/a443bc05-b8b9-4771-a12b-
0638bef401f5/addn_hosts  --dhcp-optsfile=/var/lib/neutron/dhcp/a443bc05-b8b9-4771-
a12b-0638bef401f5/opts  --dhcp-leasefile=/var/lib/neutron/dhcp/a443bc05-b8b9-4771-
a12b-0638bef401f5/leases  --dhcp-range=set:tag0,192.168.88.0,static,86400s  --dhcp-
lease-max=256 --conf-file=/etc/neutron/dnsmasq.conf --domain=openstacklocal
```

Для ВМ через этот интерфейс предоставляются метаданные и службы `DHCP`, `DNS`.

```
# ip netns exec qdhcp-a443bc05-b8b9-4771-a12b-0638bef401f5 ip a | grep -A1 tap83171bc3-22
9: tap83171bc3-22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN qlen 1000
    link/ether fa:16:3e:f0:b1:7a brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.3/24 brd 192.168.88.255 scope global tap83171bc3-22
       valid_lft forever preferred_lft forever
    inet 169.254.169.254/16 brd 169.254.255.255 scope global tap83171bc3-22
       valid_lft forever preferred_lft forever
[root@controller ~]# ip netns exec qdhcp-a443bc05-b8b9-4771-a12b-0638bef401f5 ss -4lnp | egrep "67|53|80"
tcp   UNCONN    0    0         192.168.88.3:53              *:*      users:(("dnsmasq",20916,8))
tcp   UNCONN    0    0      169.254.169.254:53              *:*      users:(("dnsmasq",20916,6))
tcp   UNCONN    0    0              *:67              *:*      users:(("dnsmasq",20916,4))
tcp   LISTEN    0    128            *:80              *:*      users:(("neutron-ns-meta",2970,5))
tcp   LISTEN    0    5         192.168.88.3:53              *:*      users:(("dnsmasq",20916,9))
tcp   LISTEN    0    5      169.254.169.254:53              *:*      users:(("dnsmasq",20916,7))
```

Порт и интерфейс `ha-493f39a8-a6` принадлежат HA-роутеру и участвуют в процессах организации высокой доступности.

По имени этого интерфейса можно найти хелпер, управляющий процессом `keepalived` в системе, и просмотреть его `pid`, выданные адреса аренды и другие параметры:

```
# ps aux | grep ha-493f39a8-a6
neutron   3074  0.0  0.3 302208 49092 ?       S    2018   0:04 /usr/bin/python3.7 /usr/lib/python-exec/python3.7/neutron-keepalived-state-change --router_id=fbbc09f0-0673-49e8-8c5f-770798202239 --namespace=qrouter-fbbc09f0-0673-49e8-8c5f-770798202239 --conf_dir=/var/lib/neutron/ha_confs/fbbc09f0-0673-49e8-8c5f-770798202239 --monitor_interface=ha-493f39a8-a6 --monitor_cidr=169.254.0.1/24 --pid_file=/var/lib/neutron/external/pids/fbbc09f0-0673-49e8-8c5f-770798202239.monitor.pid --state_path=/var/lib/neutron --user=9003 --group=9003
```

А также статус текущего инстанса keepalived по отношению к другим контроллерам, лог файлы и другую информацию.

```
# cat /var/lib/neutron/ha_confs/fbbc09f0-0673-49e8-8c5f-770798202239/state
master
```

Порт и интерфейс `qr-ab5b4c29-7b` принадлежат маршрутизатору, смотрящему в виртуальную подсеть и держащему IP-адрес, являющийся шлюзом из этой виртуальной сети.

Порт и интерфейс `qg-2ab5c9cf-05` принадлежат маршрутизатору, смотрящему в реальную подсеть и держащему IP-адрес, являющийся шлюзом для маршрутизатора.

6.5. Подсети

Подсети — это представление в рамках платформы сетевого уровня модели OSI (L3).Фактически это блоки IP-адресов и сопутствующая конфигурация, которая будет передана виртуальным машинам при подключении к этой подсети: DHCP, DNS, статические маршруты.

При создании подсети, помимо стандартных сетевых настроек: имени, адресации и тому подобного, можно выбрать версию протокола IP. При выборе IPv6 становятся доступны опции «Режим адресации» и «Режим RA», которые отвечают за то, откуда и каким образом подключенная к подсети ВМ получает IP-адрес и дополнительную информацию.

Возможные комбинации этих опций:

Режим RA	Режим адресации	Результат
не задано	slaac	ВМ получает IPv6-адрес от внешнего маршрутизатора (не управляемого ПВ РУСТЭК) с использованием `SLAAC`.
не задано	dhcpv6-stateful	ВМ получает IPv6-адрес и дополнительную информацию от ПВ РУСТЭК (`dnsmasq`) с использованием `DHCPv6` с **отслеживанием состояния**.
не задано	dhcpv6-stateless	ВМ получает IPv6-адрес от внешнего маршрутизатора с использованием `SLAAC` и дополнительную информацию от ПВ РУСТЭК (`dnsmasq`) с использованием `DHCPv6` **без сохранения состояния**.
slaac	не задано	ВМ использует `SLAAC` для получения IPv6-адреса от ПВ РУСТЭК (`radvd`).
dhcpv6-stateful	не задано	ВМ получает адрес IPv6 и дополнительную информацию от внешнего сервера `DHCPv6` с **отслеживанием состояния**.

Режим RA	Режим адресации	Результат
dhcpv6-stateless	не задано	ВМ получает адрес IPv6 от ПВ РУСТЭК (`radvd`) с помощью `SLAAC` и дополнительную информацию от внешнего сервера `DHCPv6` **без сохранения состояния**.
slaac	slaac	ВМ получает IPv6-адрес от ПВ РУСТЭК (`radvd`) с использованием `SLAAC`.
dhcpv6-stateful	dhcpv6-stateful	ВМ получает IPv6-адрес и дополнительную информацию от ПВ РУСТЭК (`dnsmasq`) с использованием `DHCPv6` с **отслеживанием состояния**.
dhcpv6-stateless	dhcpv6-stateless	ВМ получает IPv6-адрес от ПВ РУСТЭК (`radvd`) с использованием `SLAAC` и дополнительную информацию от ПВ РУСТЭК (`dnsmasq`) с использованием `DHCPv6` **без сохранения состояния**.

6.6. Порты

6.6.1. Сетевые порты

Порт в ПВ РУСТЭК — это логическое подключение виртуального сетевого интерфейса к подсети. Порты могут быть связаны с ВМ, DHCP-серверами, маршрутизаторами, межсетевыми экранами, балансировщиками нагрузки и т.д. Можно даже создать порты, чтобы просто зарезервировать IP-адреса из подсети. `Neutron` хранит взаимосвязи портов в своей базе данных, и использует эту информацию для создания коммутирующих соединений на уровне виртуального коммутатора через сетевой плагин и агент. Всякий раз, когда что-то подключается к подсети, это соединение называется портом. Порт также описывает соответствующую сетевую конфигурацию, такую как MAC- и IP-адрес, которые будут использоваться на этом порту.

У портов есть атрибут `Device Owner`, который определяет, к какому типу относится конкретный порт:

```
node01 ~ $ openstack port list --long
+--------------------------------------+------------------------------------------------------+-------------------+--------
-------------------------------------------------------------------------+--------+----------------+-----------------------
--------------+------+
| ID                                   | Name                                                 | MAC Address       | Fixed IP
Addresses                                                                | Status | Security Groups | Device Owner
| Tags |
+--------------------------------------+------------------------------------------------------+-------------------+--------
-------------------------------------------------------------------------+--------+----------------+-----------------------
--------------+------+
| 07c0bade-134d-4fe4-9bf6-c6c16906e66c |                                                      | fa:16:3e:ad:03:2d |
ip_address='10.11.15.108', subnet_id='027f967b-4c0a-4d07-a688-b8bb8cccbd22'       | ACTIVE  | None                      |
network:floatingip_agent_gateway     |      |
| 0aa48fd9-3be2-4d10-95f2-fb357887f816 |                                                      | fa:16:3e:48:d7:48 |
ip_address='10.10.10.96', subnet_id='559bd6a5-70e4-4bfc-b262-9b7151cf9247'        | ACTIVE  | None                      |
network:router_centralized_snat      |      |
| 0baeac50-fae5-4202-8a92-b0866667cf51 |                                                      | fa:16:3e:bb:49:40 |
ip_address='10.11.15.150', subnet_id='027f967b-4c0a-4d07-a688-b8bb8cccbd22'    | N/A     | None            | network:floatingip
|      |
| 0f19a2b3-ac69-4c11-b03c-d59e3c7386e5 |                                                      | fa:16:3e:a7:f7:bc |
ip_address='10.11.15.107', subnet_id='027f967b-4c0a-4d07-a688-b8bb8cccbd22'       | ACTIVE  | None                      |
network:floatingip_agent_gateway     |      |
| 2a9823c9-a05f-423c-8096-8a0e2a834e3c |                                                      | fa:16:3e:90:5a:4e |
ip_address='10.10.10.51', subnet_id='559bd6a5-70e4-4bfc-b262-9b7151cf9247'      | ACTIVE | None              | compute:nova
|      |
| 2d177e68-9b06-494c-b5ab-d6362f78ca29 |                                                      | fa:16:3e:3a:c8:b8 |
ip_address='10.10.10.11', subnet_id='559bd6a5-70e4-4bfc-b262-9b7151cf9247'      | ACTIVE | None              | network:dhcp
|      |
| 2e862f22-6dc1-448e-b263-e85390e1daff | HA port tenant 3519e4fa44de4f1986e273a2a84b0207      | fa:16:3e:10:0a:25 |
ip_address='169.254.195.102', subnet_id='28d3dab0-4b76-41cc-94c6-857e0e1abbfd'  | ACTIVE  | None                      |
network:router_ha_interface          |      |
| 80107a8b-e34a-4217-8308-5cdc39ab9c68 | HA port tenant 3519e4fa44de4f1986e273a2a84b0207      | fa:16:3e:03:c2:c1 |
ip_address='169.254.193.88', subnet_id='28d3dab0-4b76-41cc-94c6-857e0e1abbfd'    | ACTIVE  | None                      |
network:router_ha_interface          |      |
```

```
      | 87b4c7a7-2275-41fc-8f52-4e07ec063297 |                                                                  | fa:16:3e:d3:25:80 |
ip_address='10.10.10.10', subnet_id='559bd6a5-70e4-4bfc-b262-9b7151cf9247'      | ACTIVE | None              | network:dhcp
|      |
      | af142409-d942-450e-8c75-4789a3f5c335 |                                                                  | fa:16:3e:4b:90:80 |
ip_address='10.11.15.127',  subnet_id='027f967b-4c0a-4d07-a688-b8bb8cccbd22'       | ACTIVE | None                      |
network:router_gateway              |      |
      | bc8f2868-0d66-4ccf-ad45-daeba8d63156 |                                                                  | fa:16:3e:be:ed:08 |
ip_address='10.11.15.115',  subnet_id='027f967b-4c0a-4d07-a688-b8bb8cccbd22'       | ACTIVE | None                      |
network:floatingip_agent_gateway    |      |
      | f307f935-1d4e-485a-bfd1-ff036cbeabb7 |                                                                  | fa:16:3e:13:a1:93 |
ip_address='10.10.10.1',  subnet_id='559bd6a5-70e4-4bfc-b262-9b7151cf9247'          | ACTIVE | None                      |
network:router_interface_distributed |      |
      +--------------------------------------+------------------------------------------------------------------+-------------------+--------
-------------------------------------------------------------------------+--------+----------------+-----------------------
-------------+------+
```

6.6.2. Основные типы портов

compute:nova — порт подключения ВМ. Эти порты обычно создаются автоматически в процессе создания виртуальной машины. Часть «compute» указывает, что порт создан на вычислительном узле.

network:dhcp — порт, связанный с сервером DHCP. Часть «network» подразумевает, что этот порт создан на узле с ролью «управление сетями». В случае, когда эта роль настроена на нескольких узлах, для каждой сети создается несколько таких портов, по одному на каждый узел.

network:router_interface — шлюз во внешнюю сеть для подсети и ее виртуальных машин. Такие порты создаются при подключении роутера к подсети, и, как и в предыдущем случае, создаётся по одному порту на каждом узле с ролью «управление сетями».

router_gateway — шлюз подключения роутера во внешнюю сеть.

network:floatingip — порт плавающего IP. Фактически порт не создается — обратите внимание на поле `Status` в листинге портов выше. Но в неймспейсе роутера на узле с ВМ, которой принадлежит этот IP, создается правило `iptables`, обеспечивающее трансляцию плавающего IP во внутренний IP-адрес ВМ:

```
    node03 ~ $ ip netns exec qrouter-86f5ce20-1926-4d58-b7a7-5f3fd49e897e iptables-
save|grep "10.11.15.150"
    -A neutron-l3-agent-PREROUTING -d 10.11.15.150/32 -i rfp-86f5ce20-1 -j DNAT --
to-destination 10.10.10.80
    -A  neutron-l3-agent-float-snat  -s  10.10.10.80/32  -j  SNAT  --to-source
10.11.15.150 --random-fully
```

network:floatingip_agent_gateway — порты для выхода во внешнюю сеть ВМ с плавающими IP. Создаются автоматически, на всех узлах с ролями «управление сетями» и «физический узел».

network:router_ha_interface — порты для обеспечения отказоустойчивости роутера. Создаются на всех узлах с ролью «управление сетями». IP-адреса, которые назначаются на такие порты, относятся к категории link-local и используются только для обмена служебным трафиком механизма HA.

network:router_centralized_snat — порт DVR-роутера для подключения к внешней сети.

6.7. Роутеры

Виртуальные маршрутизаторы (роутеры) в ПВ РУСТЭК существуют как сетевые пространства имен в операционной системе (network namespaces) на узлах с ролями «Физический узел» и «Управление сетями». Для удобства узлы с ролью «Управление сетями» мы будем называть «сетевой узел».

Итак, роутер обычно подключается к одной или нескольким внутренним сетям и одной внешней сети. Его интерфейсы можно идентифицировать по префиксам:

- **qg** — интерфейс шлюза, смотрящий в реальную подсеть;
- **qr** — интерфейс виртуального маршрутизатора, смотрящий в виртуальную подсеть.

Роутеры ПВ РУСТЭК ответственны за организацию входящих и исходящих подключений к виртуальным сетям и из них посредством маршрутизации или трансляции сетевых адресов.

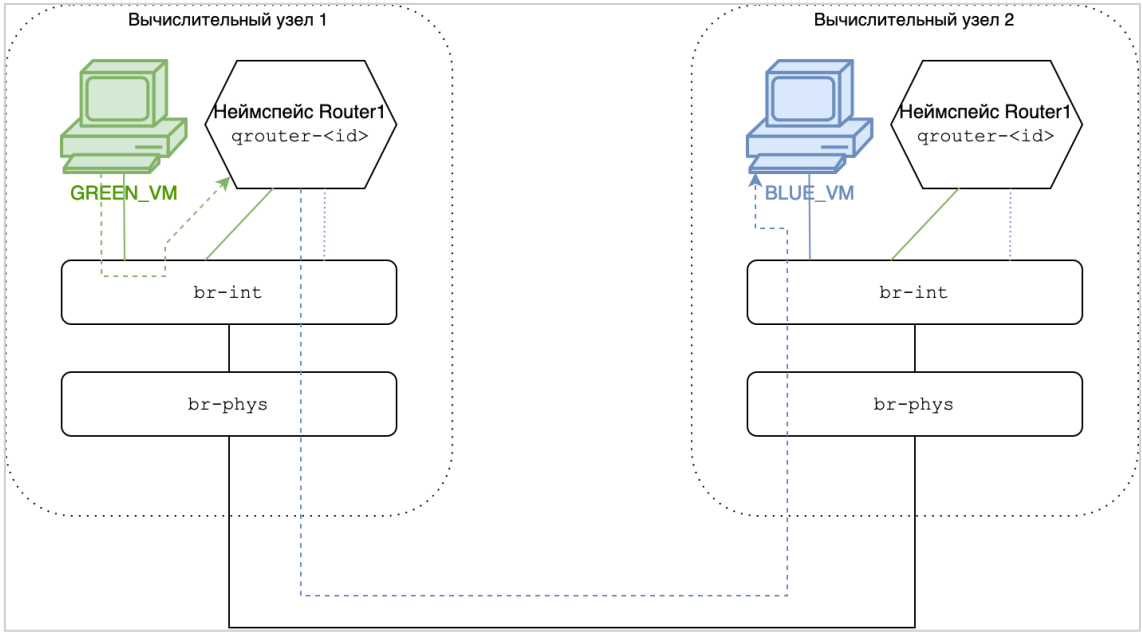
Трафик из виртуальных сетей отправляется через `qr`-интерфейсы и выходит во внешнюю сеть через `qg`-интерфейс. Таблицы маршрутизации внутри пространства имен определяют, как маршрутизируется трафик, а правила `iptables` определяют как он фильтруется или транслируется, если это необходимо.

6.7.1. Распределенный виртуальный роутер (DVR-router)

В ПВ РУСТЭК используется связка DVR-SNAT-роутеров на сетевых узлах и DVR-роутеров на физических узлах: это минимизирует нагрузку на сетевые узлы за счёт того, что трафик между ВМ или от ВМ с плавающими IP-адресами маршрутизируется непосредственно на физических узлах.

Рассмотрим ситуацию, когда DVR-маршрутизатор с именем `Router1` подключен к двум подсетям пользователей: `GREEN_NET` и `BLUE_NET`. Виртуальные машины в каждой подсети используют соответствующие шлюзы по умолчанию для маршрутизации трафика в другую подсеть через маршрутизатор Router1.

Рассмотрим детально прохождение трафика от `GREEN_VM` к `BLUE_VM`:



Итак, две подсети соединены одним маршрутизатором `Router1`, имеющим по одному интерфейсу `Router1 interface 0` и `Router1 interface 1` в каждой из этих подсетей.

```
# openstack router list
+--------------------------------------+---------+--------+-------+----------------------------------+-------------+------+
| ID                                   | Name    | Status | State | Project                          | Distributed | HA   |
+--------------------------------------+---------+--------+-------+----------------------------------+-------------+------+
| ca269272-7fe0-4b84-ae61-fb6b2a1806fb | Router1 | ACTIVE | UP    | 6aba365ce4cf4578810752488d2d60a8 | True        | True |
+--------------------------------------+---------+--------+-------+----------------------------------+-------------+------+
host201 ~ # openstack port list --router Router1
+--------------------------------------+-------------------------------------------------------+-------------------+-------------
----------------------------------------+--------+
| ID                                   | Name                                                  | MAC Address       | Fixed IP
Addresses                              | Status |
+--------------------------------------+-------------------------------------------------------+-------------------+-------------
----------------------------------------+--------+
| 020d686f-fc09-4822-ad97-0ec3483fd5ae | Router1 interface 0                                   | fa:16:3e:92:c9:9a |
ip_address='192.168.0.1',                          | ACTIVE |
|                                      |                                                       |                   |
subnet_id='BLUE_NET'     |        |
|  91c4e113-739a-4c2b-992e-0689750fa4de  |  HA  port  tenant  6aba365ce4cf4578810752488d2d60a8  |  fa:16:3e:00:aa:3e  |
ip_address='169.254.194.7',                        | ACTIVE |
|                                      |                                                       |                   |
subnet_id='3d42e445-e3f3-463d-83b1-1dd3e78f3bad'     |        |
| c8dad3c3-37dd-439b-8a06-ab72b28b09ef | Router1 interface 1                                   | fa:16:3e:0b:0e:8b |
ip_address='172.16.0.1',                           | ACTIVE |
|                                      |                                                       |                   |
subnet_id='GREEN_NET'     |        |
+--------------------------------------+-------------------------------------------------------+-------------------+-------------
----------------------------------------+--------+
```

Используя команду `ip netns exec`, мы видим, что `qr-`интерфейсы внутри пространства имен роутера на каждом физическом и сетевом узле используют один и тот же интерфейс, IP- и MAC-адреса:

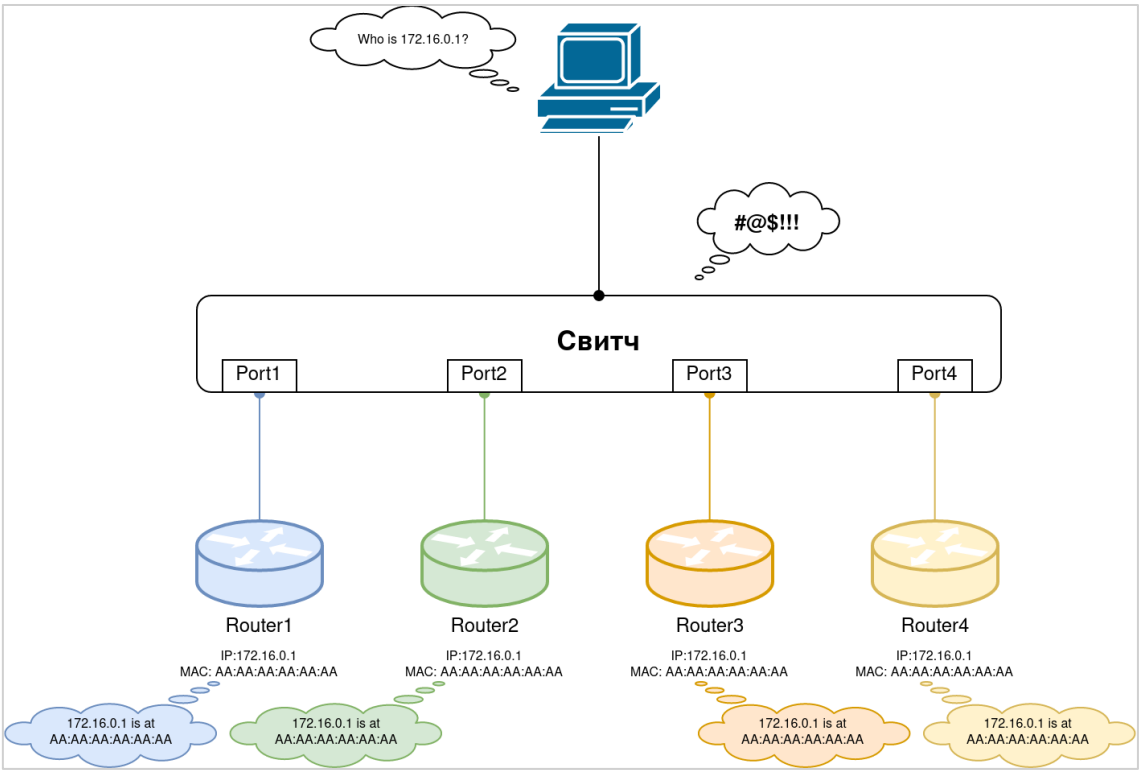
```
    host201 ~ # ip net exec qrouter-ca269272-7fe0-4b84-ae61-fb6b2a1806fb ip a
    1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
           valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
           valid_lft forever preferred_lft forever
    23: qr-020d686f-fc: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
        link/ether fa:16:3e:92:c9:9a brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.1/24 brd 192.168.0.255 scope global qr-020d686f-fc
           valid_lft forever preferred_lft forever
        inet6 fe80::f816:3eff:fe92:c99a/64 scope link
           valid_lft forever preferred_lft forever
    24: qr-c8dad3c3-37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
        link/ether fa:16:3e:0b:0e:8b brd ff:ff:ff:ff:ff:ff
        inet 172.16.0.1/24 brd 172.16.0.255 scope global qr-c8dad3c3-37
           valid_lft forever preferred_lft forever
        inet6 fe80::f816:3eff:fe0b:e8b/64 scope link
           valid_lft forever preferred_lft forever

    host199 ~ # ip net exec qrouter-ca269272-7fe0-4b84-ae61-fb6b2a1806fb ip a
    1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
           valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
           valid_lft forever preferred_lft forever
    25: qr-020d686f-fc: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
        link/ether fa:16:3e:92:c9:9a brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.1/24 brd 192.168.0.255 scope global qr-020d686f-fc
           valid_lft forever preferred_lft forever
        inet6 fe80::f816:3eff:fe92:c99a/64 scope link
           valid_lft forever preferred_lft forever
    28: qr-c8dad3c3-37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
        link/ether fa:16:3e:0b:0e:8b brd ff:ff:ff:ff:ff:ff
        inet 172.16.0.1/24 brd 172.16.0.255 scope global qr-c8dad3c3-37
           valid_lft forever preferred_lft forever
        inet6 fe80::f816:3eff:fe0b:e8b/64 scope link
           valid_lft forever preferred_lft forever
```

В выводе видно, что неймспейсы `qrouter-`, которые относятся к маршрутизатору `Router1`, содержат интерфейсы с IP-адресами из `GREEN_NET` и `BLUE_NET`. Использование таблиц маршрутизации и правил потоков Open vSwitch позволяет трафику между ВМ за одним и тем же DVR-роутером маршрутизироваться между физическими узлами напрямую.

Когда DVR-роутер подключается к подсети, он создаётся на всех физических и сетевых узлах. Агенты L3 (`neutron-l3-agent`) настраивают соответствующий `qrouter-`неймспейс, а агенты Open vSwitch (`neutron-openvswitch-agent`) подключают интерфейсы маршрутизатора к сетевым мостам и настраивают соответствующие правила в Open vSwitch.

6.7.1.1. Распределенные порты маршрутизаторов

Без должных мер предосторожности создание портов с одинаковыми IP- и MAC-адресами на нескольких физических узлах будет создавать серьёзные проблемы в сети. Представьте себе физическую топологию, похожую на следующую схему:



В большинстве сетевых сред окружение, состоящее из нескольких маршрутизаторов с одинаковыми IP- и MAC-адресами, подключенных к коммутатору, приведет к тому, что коммутатор будет постоянно перезаписывать ARP-таблицу из-за повторного изучения местоположения MAC-адресов на разных портах. Такое поведение называют MAC-flapping, и оно приводит к нестабильности в работе сети. Виртуальные коммутаторы могут демонстрировать такое же поведение независимо от типа сегментации: когда виртуальный коммутатор обнаружит, что MAC-адрес существует как локально на вычислительном узле, так и удаленно, это приведет к MAC-flapping точно так же, как и на обычном физическом коммутаторе.

Чтобы избежать MAC-flapping в сети, Neutron выделяет уникальный MAC-адрес для каждого узла, который используется всякий раз, когда трафик от DVR-маршрутизатора покидает узел. Для изменения исходного MAC-адреса пакета на выделенный уникальный MAC-адрес, закрепленный за соответствующим узлом, используются правила потоков Open vSwitch.

В следующем выводе показаны правила потоков на мосту `br-phys` узла `host201`, которые демонстрируют переписывание неуникального MAC-адреса `qr-`интерфейса на уникальный MAC-адрес, закрепленный за узлом `host201`:

```
    host201 ~ # ovs-ofctl --no-stats dump-flows br-phys
    ...
     table=1,                          priority=1,dl_vlan=2,dl_src=fa:16:3e:92:c9:9a
actions=mod_dl_src:fa:16:3f:cc:36:33,resubmit(,2)
     table=1,                          priority=1,dl_vlan=3,dl_src=fa:16:3e:0b:0e:8b
actions=mod_dl_src:fa:16:3f:cc:36:33,resubmit(,2)
    ...
```

Аналогичным образом, если трафик, поступающий на физический узел, соответствует MAC-адресу и ID сегментации ВМ, MAC-адрес источника меняется с уникального MAC-адреса исходного узла на MAC-адрес локального шлюза при прохождении пакетом моста `br-int`:

```
    host201 ~ # ovs-ofctl --no-stats  dump-flows br-int
    ....
    priority=5,in_port="int-br-phys",dl_dst=fa:16:3f:cc:36:33 actions=resubmit(,4)
    ....
    priority=4,in_port="int-br-phys",dl_src=fa:16:3f:8f:7f:9d actions=resubmit(,2)
    ....
    priority=4,in_port="int-br-phys",dl_src=fa:16:3f:61:a7:9d actions=resubmit(,2)
    ....
    table=2,                     priority=20,dl_vlan=2305,dl_dst=fa:16:3e:bf:e0:f1
actions=mod_dl_src:fa:16:3e:92:c9:9a,resubmit(,60)
    table=2,                     priority=20,dl_vlan=2309,dl_dst=fa:16:3e:7b:b2:1b
actions=mod_dl_src:fa:16:3e:0b:0e:8b,resubmit(,60)
    table=2, priority=1 actions=drop
    ....
    table=4,                         priority=5,dl_vlan=2,dl_dst=fa:16:3f:cc:36:33
actions=mod_dl_dst:fa:16:3e:92:c9:9a,strip_vlan,output:"qr-020d686f-fc"
    table=4,                         priority=5,dl_vlan=3,dl_dst=fa:16:3f:cc:36:33
actions=mod_dl_dst:fa:16:3e:0b:0e:8b,strip_vlan,output:"qr-c8dad3c3-37"
    ....
```

Строка 3: когда пакет приходит из моста `br-phys` через интерфейс `int-br-phys` с назначением уникального MAC-адреса узла `fa:16:3f:cc:36:33` — переслать пакет в таблицу `table 4`.

Строки 13-14: заменить уникальный MAC-адрес узла на внутренний MAC-адрес интерфейса роутера, принадлежащего соответствующей подсети, затем убрать локальный тег `VLAN` и отправить пакет через интерфейс `qr-`.

Поскольку перезапись заголовка фреймов происходит до или после того, как трафик покидает или поступает в ВМ, виртуальная машина ничего "не знает" о внесении изменений в её в фреймы и работает в обычном режиме без сбоев.

Рассмотрим путь трафика более детально.

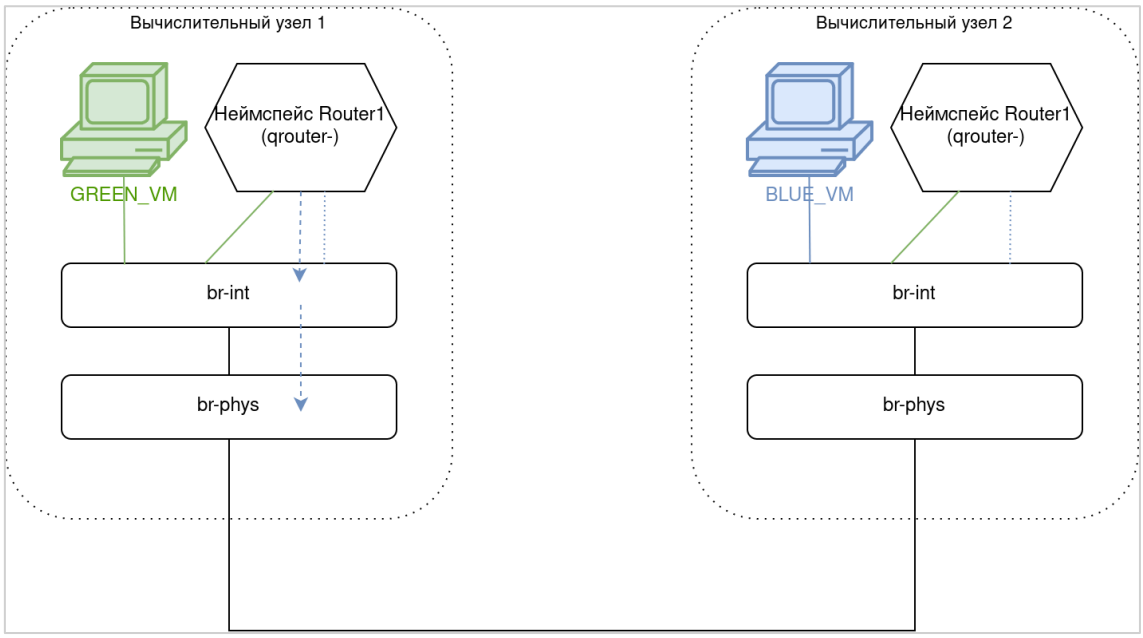
Трафик от `GREEN_VM`, находящийся на Физическом узле 1 до `BLUE_VM`, находящийся на Физическом узле 2, сначала будет перенаправлен к своему локальному шлюзу подсети через мост `br-int` и далее в пространство имен маршрутизатора `qrouter-`:

source MAC	destination MAC	source IP	destination IP
`GREEN_VM`	`green router interface (gateway)`	`GREEN_VM`	`BLUE_VM`

Маршрутизатор на Физическом узле 1 перенаправит трафик от `GREEN_VM` к `BLUE_VM`, заменив исходный MAC-адрес ВМ адресом шлюза из `BLUE_NET`, а MAC-адрес назначения — на адрес `BLUE_VM`:

source MAC	destination MAC	source IP	destination IP
`blue router interface (gateway)`	`BLUE_VM`	`GREEN_VM`	`BLUE_VM`

Затем маршрутизатор отправит пакет обратно в мост `br-int`, который перенаправит пакет дальше, на мост `br-phys`:



Когда пакет поступит на мост `br-phys` на Физическом узле 1, пакет пройдёт по цепочкам правил потоков Open vSwitch, в результате чего исходный MAC-адрес изменится с `blue router interface` на уникальный MAC-адрес узла:

source MAC	destination MAC	source IP	destination IP
`Source host (FA-16-3F-CC-36-33)`	`BLUE_VM`	`GREEN_VM`	`BLUE_VM`

Далее пакет пересылается в физическую сеть и поступает на Физический узел 2 и пересылается через мост `br-phys`. Затем правило потока Open vSwitch добавляет в заголовок тег локального `VLAN`, который позволит идентифицировать пакет при пересылке на мост `br-int`.

В мосту `br-int` правило потока Open vSwitch удаляет локальный тег `VLAN` и изменяет MAC-адрес источника обратно на MAC-адрес интерфейса из роутера сети `BLUE_NET`, после чего пакет перенаправляется на `BLUE_VM`:

source MAC	destination MAC	source IP	destination IP
`blue router interface`	`BLUE_VM`	`GREEN_VM`	`BLUE_VM`

Обратный трафик от `BLUE_VM` к `GREEN_VM` проходит аналогичную маршрутизацию через соответствующие маршрутизаторы и мосты на каждом физическом узле, но в обратной последовательности.

6.8. Безопасность

6.8.1. Профили безопасности

Профили безопасности представляют собой контейнер для правил брандмауэра в сетевом неймспейсе, которые контролируют входящий к ВМ и исходящий от неё сетевой трафик на уровне порта.

Основные аспекты профилей безопасности:

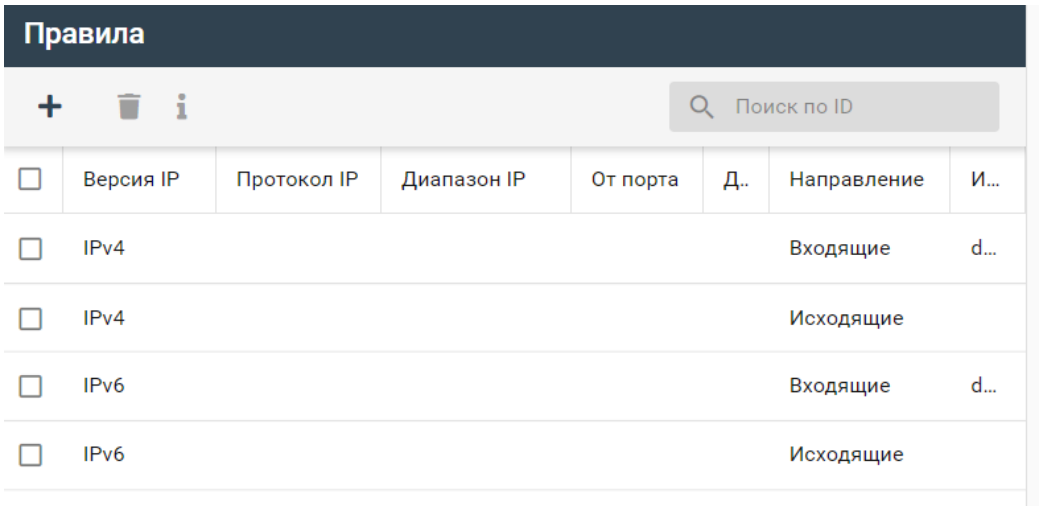
- профили безопасности используют политику «что не разрешено, то запрещено», и содержат только правила, разрешающие определенный трафик;
- каждый порт может ссылаться на одну или несколько групп безопасности;
- все правила профилей безопасности представляют собой конфигурацию `iptables` в соответствующем сетевом неймспейсе.

Каждый проект содержит профиль безопасности по умолчанию, который разрешает весь исходящий трафик и трафик внутри сети. Правила в дефолтном профиле можно изменить, но важно помнить, что изменения затронут любой ВМ и порт, для которых явно не указан профиль безопасности, поскольку для них применяется профиль безопасности по умолчанию.

Будьте внимательны, при использовании в сети `службы метаданных` удаление правила «разрешить весь исходящий трафик» запретит доступ также и к адресу `169.254.169.254`, с которого ВМ получают метаданные.

6.8.2. Группа безопасности «по умолчанию»

Вновь созданный проект будет иметь группу безопасности с именем «по умолчанию». Эта группа безопасности уникальна для этого проекта, и все проекты будут иметь группу безопасности с именем «по умолчанию». Эту группу безопасности нельзя удалить. Эта группа безопасности содержит четыре правила:



Как мы видим, группа безопасности «по умолчанию» содержит два правила **выхода** (исходящий трафик) и два правила **входа** (входящий трафик).

6.8.2.1. Правила исходящего трафика

Два правила выхода работают одинаково, но одно из них нацелено на IPv4, а другое — на IPv6. Они полностью открывают брандмауэр из ВМ, используя CIDR **0.0.0.0/0** и **::/ 0** для диапазона IP.

6.8.2.2. Правила входа

Правила входа по умолчанию закрывают весь трафик извне, за исключением ответов. На это указывает "Исходящий профиль" — default

6.8.2.3. Резюме

Группа безопасности позволяет ВМ взаимодействовать с хостами во внешних сетях, если трафик инициируется изнутри, и свободно взаимодействовать ВМ находящимся в одной сети друг с другом.

6.8.3. Операции с профилями безопасности при их использовании в ВМ

- Добавление/удаление правил влияет на трафик в/из ВМ.
- Для удаления профиля безопасности, который используется в ВМ, необходимо сначала отключить его от использующих этот профиль ВМ.

6.8.4. Просмотр правил брандмауэра для ВМ

Сначала необходимо узнать ID порта, выполнив на физическом узле, на котором запущена виртуальная машина, команду:

```
$ openstack port list --server 37df4451-28d9-4bc1-9adc-960d532e7b68
+--------------------------------------+------+-------------------+-------------------------------------------------------
-------------------------+--------+
| ID                                   | Name | MAC Address       | Fixed IP Addresses                                                                  |
Status |
+--------------------------------------+------+-------------------+-------------------------------------------------------
-------------------------+--------+
| 0aee4ca5-d719-4339-bb27-724a5790f637 |      | fa:16:3e:40:88:2b | ip_address='10.10.10.62', subnet_id='f8c51404-2cda-
47b7-8e25-cc53ac2f7f4e' | ACTIVE |
+--------------------------------------+------+-------------------+-------------------------------------------------------
-------------------------+--------+
```

Узнав ID порта, можно просмотреть правила iptables, которые для него действуют. Для этого нужно выполнить команду `iptables-save` и отфильтровать её вывод по первой группе чисел в ID:

```
$ iptables-save |grep 0aee4ca5

-A neutron-openvswi-PREROUTING -m physdev --physdev-in qvb0aee4ca5-d7 -m comment --comment "Set zone
for 0aee4ca5-d719-4339-bb27-724a5790f637" -j CT --zone 4097
-A neutron-openvswi-PREROUTING -i qvb0aee4ca5-d7 -m comment --comment "Set zone for 0aee4ca5-d719-
4339-bb27-724a5790f637" -j CT --zone 4097
-A neutron-openvswi-PREROUTING -m physdev --physdev-in tap0aee4ca5-d7 -m comment --comment "Set zone
for 0aee4ca5-d719-4339-bb27-724a5790f637" -j CT --zone 4097
:neutron-openvswi-i0aee4ca5-d - [0:0]
:neutron-openvswi-o0aee4ca5-d - [0:0]
:neutron-openvswi-s0aee4ca5-d - [0:0]
-A neutron-openvswi-FORWARD -m physdev --physdev-out tap0aee4ca5-d7 --physdev-is-bridged -m comment -
-comment "Direct traffic from the VM interface to the security group chain." -j neutron-openvswi-sg-chai
n
-A neutron-openvswi-FORWARD -m physdev --physdev-in tap0aee4ca5-d7 --physdev-is-bridged -m comment --
comment "Direct traffic from the VM interface to the security group chain." -j neutron-openvswi-sg-chain
-A neutron-openvswi-INPUT -m physdev --physdev-in tap0aee4ca5-d7 --physdev-is-bridged -m comment --
comment "Direct incoming traffic from VM to the security group chain." -j neutron-openvswi-o0aee4ca5-d
-A neutron-openvswi-i0aee4ca5-d -m state --state RELATED,ESTABLISHED -m comment --comment "Direct
packets associated with a known session to the RETURN chain." -j RETURN
-A neutron-openvswi-i0aee4ca5-d -d 10.10.10.62/32 -p udp -m udp --sport 67 --dport 68 -j RETURN
-A neutron-openvswi-i0aee4ca5-d -d 255.255.255.255/32 -p udp -m udp --sport 67 --dport 68 -j RETURN
-A neutron-openvswi-i0aee4ca5-d -p tcp -m tcp --dport 22 -j RETURN
-A neutron-openvswi-i0aee4ca5-d -m set --match-set NIPv4cb19b00e-bbb8-4d82-b31c- src -j RETURN
-A neutron-openvswi-i0aee4ca5-d -m state --state INVALID -m comment --comment "Drop packets that appear
related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack." -j DROP
-A neutron-openvswi-i0aee4ca5-d -m comment --comment "Send unmatched traffic to the fallback chain." -
j neutron-openvswi-sg-fallback
-A neutron-openvswi-o0aee4ca5-d -s 0.0.0.0/32 -d 255.255.255.255/32 -p udp -m udp --sport 68 --dport
67 -m comment --comment "Allow DHCP client traffic." -j RETURN
-A neutron-openvswi-o0aee4ca5-d -j neutron-openvswi-s0aee4ca5-d
-A neutron-openvswi-o0aee4ca5-d -p udp -m udp --sport 68 --dport 67 -m comment --comment "Allow DHCP
client traffic." -j RETURN
-A neutron-openvswi-o0aee4ca5-d -p udp -m udp --sport 67 --dport 68 -m comment --comment "Prevent DHCP
Spoofing by VM." -j DROP
-A neutron-openvswi-o0aee4ca5-d -m state --state RELATED,ESTABLISHED -m comment --comment "Direct
packets associated with a known session to the RETURN chain." -j RETURN
-A neutron-openvswi-o0aee4ca5-d -j RETURN
-A neutron-openvswi-o0aee4ca5-d -m state --state INVALID -m comment --comment "Drop packets that appear
related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack." -j DROP
-A neutron-openvswi-o0aee4ca5-d -m comment --comment "Send unmatched traffic to the fallback chain." -
j neutron-openvswi-sg-fallback
-A neutron-openvswi-s0aee4ca5-d -s 10.10.10.62/32 -m mac --mac-source fa:16:3e:40:88:2b -m comment --
comment "Allow traffic from defined IP/MAC pairs." -j RETURN
-A neutron-openvswi-s0aee4ca5-d -m comment --comment "Drop traffic without an IP/MAC allow rule." -j
DROP
-A neutron-openvswi-sg-chain -m physdev --physdev-out tap0aee4ca5-d7 --physdev-is-bridged -m comment -
-comment "Jump to the VM specific chain." -j neutron-openvswi-i0aee4ca5-d
-A neutron-openvswi-sg-chain -m physdev --physdev-in tap0aee4ca5-d7 --physdev-is-bridged -m comment -
-comment "Jump to the VM specific chain." -j neutron-openvswi-o0aee4ca5-d
```

Обратите внимание на выделенную строку: по умолчанию всем ВМ запрещено работать в качестве DHCP-серверов во избежание конфликтов, поэтому такой трафик блокируется.

6.8.5. Port-security

Расширение `port-security` автоматически добавляет правила защиты от спуфинга (anti-spoofing) виртуальных машин, предотвращающие прохождение стороннего трафика через виртуальную машину. Этот механизм можно отключить, если требуется использовать ВМ, например, в качестве роутера.

За включение или отключение `port-security` для сети отвечает чекбокс **Безопасность портов** в диалоге создания сети. По умолчанию этот параметр включен и наследуется всеми ВМ, подключенными в такую сеть. Отключение этой опции для конкретного ВМ или для всей сети целиком возможно только через интерфейс командной строки.

6.8.5.1. Отключение port-security

6.8.5.1.1 Для ВМ

С помощью CLI находим порт нужного ВМ. В нашем примере ID — `a149b3ed-32f1-422b-a7ba-4bf6fc834705`:

```
[root@compute1 ~]$ openstack port list -f value -c ID --server a149b3ed-32f1-422b-a7ba-4bf6fc834705
1d818f7c-fea3-40cf-9543-0762eda8e3ec
```

Отключаем для этого порта механизм `port-security`:

```
[root@compute1 ~]$ openstack port set --no-allowed-address --disable-port-security --no-security-group 1d818f7c-fea3-40cf-9543-0762eda8e3ec
```

Отключение port-security на порту ВМ влечет за собой отключение всех профилей безопасности и механизма `allowed address pairs`.

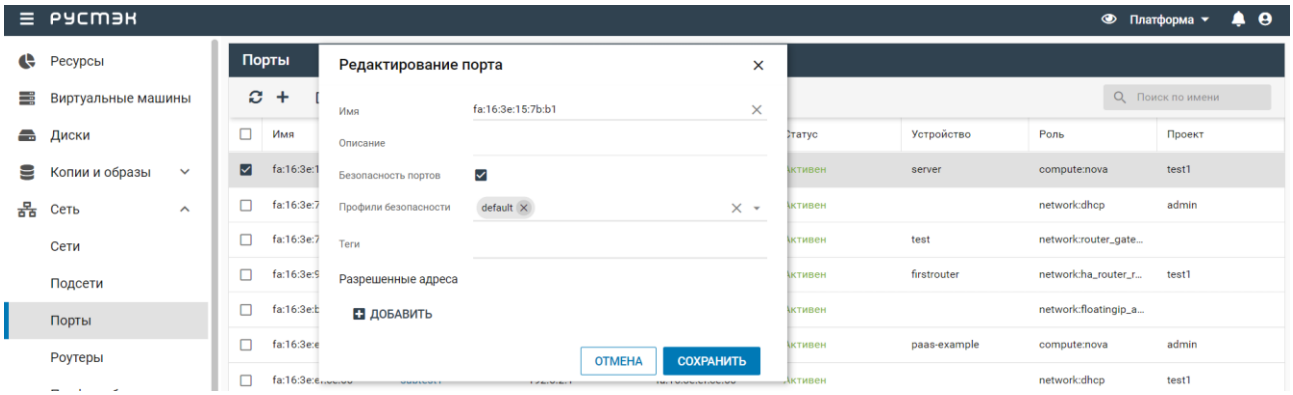
6.8.5.1.2 Для сети

Отключение механизма в параметрах сети для выключения наследования на порты ВМ также возможно с помощью CLI:

```
[root@compute1 ~]$ openstack net set --disable-port-security <network>
```

6.8.6. Allowed address pairs

Расширение `allowed-address-pairs` позволяет на интерфейсах ВМ назначать IP- и MAC-адреса, отличные от назначенных автоматически. По умолчанию весь трафик с интерфейса ВМ, о котором «не знает» ПВ РУСТЭК, блокируется: например, если пользователь решит сменить IP- или MAC-адрес в гостевой операционной системе. Однако, иногда такие изменения необходимы, например, при использовании виртуальной машины в качестве роутера или VPN-сервера или при использовании технологий балансировки и высокой доступности, таких как `haproxy` или `keepalived/pacemaker`. Для этого в портале следует добавить новые пары IP- и MAC-адресов через диалог редактирования порта:



Максимальное количество доступных пар адресов регулируется опцией `max_allowed_address_pair` в конфигурационном файле `/etc/neutron/neutron.conf` узлов с ролью «Управление сетями». Если оно не определено, то в «Разрешённые адреса» можно указать до 10 пар IP/MAC-адресов для одного порта.

6.8.6.1. port-security и allowed address pairs наглядно

Вернемся к нашему примеру и рассмотрим функционал детально. Вывод с включенными механизмами `port-security` и `allowed address pairs` выглядит так:

```
[root@compute1 ~]$ openstack port list --server 0cbd7a1d-1e75-4660-8152-90f09697b904
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------+--------+
| ID                                   | Name | MAC Address       | Fixed IP Addresses                                                            | Status |
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------+--------+
| 7799d60d-fea3-40cf-9543-0762eda8e3ec |      | fa:16:3e:5b:8e:3d | ip_address='192.168.88.127', subnet_id='a443bc05-b8b9-4771-a12b-0638bef401f5' | ACTIVE |
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------+--------+
[root@compute ~]$ iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N neutron-filter-top
-N neutron-openvswi-FORWARD
-N neutron-openvswi-INPUT
-N neutron-openvswi-OUTPUT
-N neutron-openvswi-i7799d60d-3
-N neutron-openvswi-local
-N neutron-openvswi-o7799d60d-3
-N neutron-openvswi-s7799d60d-3
-N neutron-openvswi-sg-chain
-N neutron-openvswi-sg-fallback
-A INPUT -j neutron-openvswi-INPUT
-A FORWARD -j neutron-filter-top
-A FORWARD -j neutron-openvswi-FORWARD
-A OUTPUT -j neutron-filter-top
-A OUTPUT -j neutron-openvswi-OUTPUT
-A neutron-filter-top -j neutron-openvswi-local
-A neutron-openvswi-FORWARD -m physdev --physdev-out tap7799d60d-3d --physdev-is-bridged -m comment --comment "Direct traffic from the VM interface to the security group chain." -j neutron-openvswi-sg-
chain
-A neutron-openvswi-FORWARD -m physdev --physdev-in tap7799d60d-3d --physdev-is-bridged -m comment --comment "Direct traffic from the VM interface to the security group chain." -j neutron-openvswi-sg-chain
-A neutron-openvswi-INPUT -m physdev --physdev-in tap7799d60d-3d --physdev-is-bridged -m comment --comment "Direct incoming traffic from VM to the security group chain." -j neutron-openvswi-o7799d60d-3
-A neutron-openvswi-i7799d60d-3 -m state --state RELATED,ESTABLISHED -m comment --comment "Direct packets associated with a known session to the RETURN chain." -j RETURN
-A neutron-openvswi-i7799d60d-3 -s 192.168.88.2/32 -p udp -m udp --sport 67 --dport 68 -j RETURN
-A neutron-openvswi-i7799d60d-3 -s 192.168.88.3/32 -p udp -m udp --sport 67 --dport 68 -j RETURN
-A neutron-openvswi-i7799d60d-3 -m set --match-set NETIPv4739375d7-7bac-4530-8 src -j RETURN
-A neutron-openvswi-i7799d60d-3 -p icmp -j RETURN
-A neutron-openvswi-i7799d60d-3 -j RETURN
-A neutron-openvswi-i7799d60d-3 -m state --state INVALID -m comment --comment "Drop packets that appear related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack." -j NFLOG --
nflog-prefix "chain:i7799d60d-3:8,m=state,state=INVALID"
-A neutron-openvswi-i7799d60d-3 -m state --state INVALID -m comment --comment "Drop packets that appear related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack." -j DROP
-A neutron-openvswi-i7799d60d-3 -m comment --comment "Send unmatched traffic to the fallback chain." -j neutron-openvswi-sg-fallback
-A neutron-openvswi-o7799d60d-3 -s 0.0.0.0/32 -d 255.255.255.255/32 -p udp -m udp --sport 68 --dport 67 -m comment --comment "Allow DHCP client traffic." -j RETURN
-A neutron-openvswi-o7799d60d-3 -j neutron-openvswi-s7799d60d-3
-A neutron-openvswi-o7799d60d-3 -p udp -m udp --sport 68 --dport 67 -m comment --comment "Allow DHCP client traffic." -j RETURN
-A neutron-openvswi-o7799d60d-3 -p udp -m udp --sport 67 --dport 68 -m comment --comment "Prevent DHCP Spoofing by VM." -j NFLOG --nflog-prefix "chain:o7799d60d-3:5,p=udp,m=udp,sport=67,dport=68"
-A neutron-openvswi-o7799d60d-3 -p udp -m udp --sport 67 --dport 68 -m comment --comment "Prevent DHCP Spoofing by VM." -j DROP
-A neutron-openvswi-o7799d60d-3 -m state --state RELATED,ESTABLISHED -m comment --comment "Direct packets associated with a known session to the RETURN chain." -j RETURN
-A neutron-openvswi-o7799d60d-3 -j RETURN
-A neutron-openvswi-o7799d60d-3 -m state --state INVALID -m comment --comment "Drop packets that appear related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack." -j NFLOG --
nflog-prefix "chain:o7799d60d-3:9,m=state,state=INVALID"
-A neutron-openvswi-o7799d60d-3 -m state --state INVALID -m comment --comment "Drop packets that appear related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack." -j DROP
-A neutron-openvswi-o7799d60d-3 -m comment --comment "Send unmatched traffic to the fallback chain." -j neutron-openvswi-sg-fallback
-A neutron-openvswi-s7799d60d-3 -s 8.8.8.8/32 -m mac --mac-source FA:16:3E:5B:8E:3D -m comment --comment "Allow traffic from defined IP/MAC pairs." -j RETURN
-A neutron-openvswi-s7799d60d-3 -s 192.168.88.127/32 -m mac --mac-source FA:16:3E:5B:8E:3D -m comment --comment "Allow traffic from defined IP/MAC pairs." -j RETURN
-A neutron-openvswi-s7799d60d-3 -j NFLOG --nflog-prefix "chain:s7799d60d-3:3,any"
-A neutron-openvswi-s7799d60d-3 -m comment --comment "Drop traffic without an IP/MAC allow rule." -j DROP
-A neutron-openvswi-sg-chain -m physdev --physdev-out tap7799d60d-3d --physdev-is-bridged -m comment --comment "Jump to the VM specific chain." -j neutron-openvswi-i7799d60d-3
-A neutron-openvswi-sg-chain -m physdev --physdev-in tap7799d60d-3d --physdev-is-bridged -m comment --comment "Jump to the VM specific chain." -j neutron-openvswi-o7799d60d-3
-A neutron-openvswi-sg-chain -j ACCEPT
-A neutron-openvswi-sg-fallback -j NFLOG --nflog-prefix "chain:sg-fallback:2,any"
-A neutron-openvswi-sg-fallback -m comment --comment "Default drop rule for unmatched traffic." -j DROP
```

Функционал `allowed address pairs` регулируется правилом выделенных в листинге строк, которые разрешают трафик из ВМ порта `7799d60d` с IP адресов `192.168.88.127`,`8.8.8.8/32` и MAC `FA:16:3E:5B:8E:3D`.

Выключив механизм на порту ВМ, видим, что вывод изменился:

```
[root@compute1 ~]$ openstack port list --server 0cbd7a1d-1e75-4660-8152-90f09697b904
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------+--------+
| ID                                   | Name | MAC Address       | Fixed IP Addresses                                                            | Status |
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------+--------+
| 7799d60d-fea3-40cf-9543-0762eda8e3ec |      | fa:16:3e:5b:8e:3d | ip_address='192.168.88.127', subnet_id='a443bc05-b8b9-4771-a12b-0638bef401f5' | ACTIVE |
+--------------------------------------+------+-------------------+-------------------------------------------------------------------------------+--------+
[root@compute ~]$ iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N neutron-filter-top
-N neutron-openvswi-FORWARD
-N neutron-openvswi-INPUT
-N neutron-openvswi-OUTPUT
-N neutron-openvswi-i7799d60d-3
-N neutron-openvswi-local
-N neutron-openvswi-o7799d60d-3
-N neutron-openvswi-s7799d60d-3
-N neutron-openvswi-sg-chain
-N neutron-openvswi-sg-fallback
-A INPUT -j neutron-openvswi-INPUT
-A FORWARD -j neutron-filter-top
-A FORWARD -j neutron-openvswi-FORWARD
-A OUTPUT -j neutron-filter-top
-A OUTPUT -j neutron-openvswi-OUTPUT
-A neutron-filter-top -j neutron-openvswi-local
-A neutron-openvswi-FORWARD -m physdev --physdev-out tap7799d60d-3d --physdev-is-bridged -m comment --comment "Accept all packets when port security is disabled." -j ACCEPT
-A neutron-openvswi-FORWARD -m physdev --physdev-in tap7799d60d-3d --physdev-is-bridged -m comment --comment "Accept all packets when port security is disabled." -j ACCEPT
-A neutron-openvswi-INPUT -m physdev --physdev-in tap7799d60d-3d --physdev-is-bridged -m comment --comment "Accept all packets when port security is disabled." -j ACCEPT
-A neutron-openvswi-sg-chain -j ACCEPT
-A neutron-openvswi-sg-fallback -m comment --comment "Default drop rule for unmatched traffic." -j DROP
```

6.9. PAT: floating IP

6.9.1. Переадресация портов с плавающего IP-адреса

Переадресация портов (Port address translation, PAT) — распространенная функция, позволяющая использовать один и тот же IP-адрес для предоставления доступа извне к разным виртуальным машинам по разным портам. Это особенно актуально для инсталляций, в которых не так много IP-адресов, доступных конечным клиентам. Например, PAT может использоваться в таком сценарии:

```
client1 -> 1.1.1.1:22   TCP => пересылается на 192.168.0.100 порт 22 TCP (VM1)
client2 -> 1.1.1.1:2022 TCP => пересылается на 192.168.0.101 порт 22 TCP (VM2)
client3 -> 1.1.1.2:2022 TCP => пересылается на 192.168.0.102 порт 22 TCP (VM3)
```

Здесь два плавающих IP-адреса используются для организации доступа по ssh к трём разным виртуальным машинам.

Механизм переадресации возможно использовать только на тех IP-адресах, которые не назначены ВМ или роутерам.
Если порт ВМ или floating IP удален, записи переадресации, соответствующие этому порту, также удаляются.

При использовании переадресации порта с floating IP необходимо разрешить в группе безопасности ВМ прохождение пакетов на соответствующий порт назначения, например, при трансляции:
`client -> 1.1.1.2:2022 TCP => пересылается на 192.168.0.101 порт 22 TCP ВМ`
необходимо открыть 22 порт TCP на ВМ правилами безопасности.

6.9.2. Настройка переадресации через интерфейс командной строки

Реализуем такую схему: `client -> 10.11.7.4:2222 TCP => пересылается на 192.168.2.79 порт 22 TCP VM1.` Для демонстрации в качестве floating-адресов будем

использовать «серые» IP-адреса. Предполагается, что внутри `VM1` запущен SSH-сервер на стандартном порту 22.

Выделим floating IP, с которого будем пробрасывать внешние порты на внутренние порты `VM1`:

```
(openstack) floating ip create --floating-ip-address 10.11.7.4 --tag port_forwarding_VM1 ext-net
.........skip..........
(openstack) floating ip list --long --any-tags 'port_forwarding_VM1'
+--------------------+--------------------+------------------+------+--------------------+----------
----------+--------+--------+-------------+----------------------+----------+------------+
| ID                 | Floating IP Address | Fixed IP Address | Port | Floating Network   | Project
| Router | Status | Description | Tags                 | DNS Name | DNS Domain |
+--------------------+--------------------+------------------+------+--------------------+----------
----------+--------+--------+-------------+----------------------+----------+------------+
| 4f2d57c4-bb79-42e3 | 10.11.7.4          | None             | None | b3a3eba3-eca0-42bf |
21e81f6de6b8488fac | None   | DOWN   |             | ['port_forwarding_VM1 | None     | None       |
| -b992-3c96f1653531 |                    |                  |      | -9872-c355ffbefb74 | 9a95c35e16cf82
|        |        |             | ']                   |          |            |
+--------------------+--------------------+------------------+------+--------------------+----------
----------+--------+--------+-------------+----------------------+----------+------------+
```

Выясним, какие порты подключены к `VM1` и какие группы безопасности они используют:

```
(openstack) port list --server VM1 --long
+--------------------------------------+------+-------------------+--------------------------------
---------+--------+--------------------------------------+--------------+------+
| ID                                   | Name | MAC Address       | Fixed IP Addresses                    |
Status | Security Groups                      | Device Owner | Tags |
+--------------------------------------+------+-------------------+--------------------------------
---------+--------+--------------------------------------+--------------+------+
| 9b68c54c-2d19-4ad4-a2a1-30574c83647c |      | fa:16:3e:c4:37:f0 | ip_address='192.168.2.79',
subnet_id='055 | ACTIVE | b354e63b-1c67-435e-8aee-81059b5f8fdd | compute:nova |      |
|                                      |      |                   | c922b-0a3b-44c8-b4fb-b6237accb1fd'
|        |                                      |              |      |
+--------------------------------------+------+-------------------+--------------------------------
---------+--------+--------------------------------------+--------------+------+
```

Создадим правило проброса внешнего порта `TCP 2222` с floating IP на внутренний порт `TCP 22` `VM1`:

```
(openstack) floating ip port forwarding create --internal-ip-address 192.168.2.79 --port 9b68c54c-
2d19-4ad4-a2a1-30574c83647c --protocol tcp --internal-protocol-port 22 --external-protocol-port 2222
10.11.7.4
.........skip..........
(openstack) floating ip port forwarding list 4f2d57c4-bb79-42e3-b992-3c96f1653531
+--------------------------------------+--------------------------------------+--------------------
+---------------+---------------+----------+
| ID                                   | Internal Port ID                     | Internal IP Address |
Internal Port | External Port | Protocol |
+--------------------------------------+--------------------------------------+--------------------
+---------------+---------------+----------+
| 0011665d-774f-4a16-82e2-2ff0c08a597d | 9b68c54c-2d19-4ad4-a2a1-30574c83647c | 192.168.2.79        |
22 |          2222 | tcp      |
+--------------------------------------+--------------------------------------+--------------------
+---------------+---------------+----------+
```

Создадим правило в группе безопасности порта ВМ:

```
(openstack) security group rule create --ingress --protocol tcp --dst-port 22 b354e63b-
1c67-435e-8aee-81059b5f8fdd
+-------------------+------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------
-----------+
|     Field                                                                                    |     Value
|
```

```
+-------------------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
-----------+
|      created_at                                                |      2021-03-09T12:39:08Z
|
|          description                                                                           |
|
|       direction                                                                 |       ingress
|
|        ether_type                                                                 |        IPv4
|
|   id                                                |   60118ee5-6876-4e24-92a0-3d43f69a25d3
|
|  location                  |  Munch({'project':  Munch({'domain_id':  'default',  'id':
'21e81f6de6b8488fac9a95c35e16cf82',   'name':   'admin',   'domain_name':   None}),   'cloud':
'rustack', 'zone': None,      |
|                                                          |   'region_name':   'RegionOne'})
|
|     name                                                                                |     None
|
|            port_range_max                                                       |            22
|
|            port_range_min                                                       |            22
|
|     project_id                                      |     21e81f6de6b8488fac9a95c35e16cf82
|
|       protocol                                                                  |       tcp
|
|              remote_group_id                                                |              None
|
|                remote_ip_prefix                                    |                0.0.0.0/0
|
|              revision_number                                                |              0
|
|           security_group_id           |           b354e63b-1c67-435e-8aee-81059b5f8fdd
|
|     tags                                                                                |     []
|
|      updated_at                                                |      2021-03-09T12:39:08Z
|
+-------------------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
-----------+
```

На этом настройка окончена. Проверим работу механизма floating IP port forwarding:

```
[test@host]$ ssh -p2222 10.11.7.4
The authenticity of host '[10.11.7.4]:2222 ([10.11.7.4]:2222)' can't be established.
ECDSA key fingerprint is SHA256:w5ZnVVzVqK+ujX7VqwLbi9fTKCmEdhDxnPsYChbeY7g.
Are you sure you want to continue connecting (yes/no/[fingerprint])? ^C   # успешное
соединение
[test@host]$ ssh -p22 10.11.7.4
ssh: connect to host 10.11.7.4 port 22: Connection refused  # соединение отклонено
```

6.9.3. Внутреннее устройство PAT

При создании правила форвардинга для floating IP на узлах, где запущен neutron-server, видно, что в сетевом неймспейсе `snat-<ID роутера>`, который обеспечивает связанность между внешней и виртуальной сетями, в `iptables` автоматически создаются соответствующие цепочки с правилами форвардинга портов:

```
(openstack) router list --name Router --long
+--------------------------------------+--------+--------+-------+----------------------------------+----------
---+------+--------+----------------------------------+------------------+-------+
| ID                                   | Name   | Status | State | Project                          | Distributed
| HA   | Routes | External gateway info                | Availability zones | Tags |
+--------------------------------------+--------+--------+-------+----------------------------------+----------
---+------+--------+----------------------------------+------------------+-------+
| e3266d21-9761-4fcb-8560-ea161 | Router | ACTIVE | UP    | 21e81f6de6b8488fac9a95c35e16c | True       |
True |        | {"enable_snat": true,          | nova               |      |
| 8057f48                       |        |        |       | f82                              |            |
|        | "external_fixed_ips":         |                    |      |
|                               |        |        |       |                                  |            |
|        | [{"subnet_id": "85fc9296-eac8 |                    |      |
|                               |        |        |       |                                  |            |
|        | -476b-b1f1-b4e2e9f3b788",     |                    |      |
|                               |        |        |       |                                  |            |
|        | "ip_address": "10.11.7.3"}],  |                    |      |
|                               |        |        |       |                                  |            |
|        | "network_id": "b3a3eba3-eca0- |                    |      |
|                               |        |        |       |                                  |            |
|        | 42bf-9872-c355ffbefb74"}      |                    |      |
+--------------------------------------+--------+--------+-------+----------------------------------+----------
---+------+--------+----------------------------------+------------------+-------+
# ip net exec snat-e3266d21-9761-4fcb-8560-ea1618057f48 iptables -S -t nat | egrep '0011665d|agent-
fip'
-N neutron-l3-agent-fip-pf
-N neutron-l3-agent-pf-0011665d
-A neutron-l3-agent-PREROUTING -j neutron-l3-agent-fip-pf
-A neutron-l3-agent-fip-pf -j neutron-l3-agent-pf-0011665d
-A neutron-l3-agent-pf-0011665d -d 10.11.7.4/32 -p tcp -m tcp --dport 2222 -j DNAT --to-destination
192.168.2.79:22
```

6.10. SR-IOV

6.10.1. Назначение SR-IOV

Механизм SR-IOV может виртуализировать один Ethernet-контроллер PCIe для отображения в виде нескольких виртуальных PCIe-устройств. Каждое виртуальное устройство может быть напрямую назначено виртуальной машине, минуя уровень гипервизора и виртуального коммутатора, что позволит достичь низкой задержки и максимальной скорости передачи.

Условные обозначения:

- PF (Physical Function) — физический Ethernet-контроллер с поддержкой SR-IOV;
- VF (Virtual Function) — виртуальное PCIe-устройство, созданное из физического Ethernet-контроллера.

ВМ могут использовать порты SR-IOV или обычные порты Open vSwitch в качестве сетевых интерфейсов:

- если используется сегментация `VLAN`, порты SR-IOV и обычные порты Open vSwitch могут взаимодействовать друг с другом по сети с разных узлов или из разных PF на одном физическом узле;
- если оба ВМ находятся на одном физическом узле и совместно используют PF на сетевом адаптере, они могут обмениваться данными только, когда оба используют один и тот же тип порта: оба используют SR-IOV или оба используют обычный Open vSwitch.

6.10.2. Настройка физического узла

6.10.2.1. Аппаратная поддержка IOMMU

Для использования SR-IOV необходима поддержка со стороны аппаратной части: IOMMU (Input/Output Memory Management Unit).

Проверьте, что в BIOS физического сервера, с которого вы планируете пробрасывать устройство включена поддержка следующих технологий:

- `Intel VT-d` или `AMD Vi` в зависимости от производителя CPU;
- `SR-IOV`, на некоторых моделях материнских плат её нужно активировать отдельно.

6.10.2.2. Программная поддержка IOMMU

В ПВ РУСТЭК по умолчанию включена поддержка `intel_iommu`, проверить это можно в файле `/proc/cmdline`: опция `intel_iommu` должна иметь значение `on`. Если по какой-либо причине поддержка отключена, добавьте параметр `intel_iommu=on` в переменную `GRUB_CMDLINE_LINUX_DEFAULT` файла `/etc/default/grub`, после чего пересоздайте конфиг `grub` командой:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Перезагрузите узел для принятия изменений.

6.10.2.3. Настройка VF

Первым делом нужно выяснить, какое количество VF поддерживает выбранный адаптер. В нашем примере мы будем использовать устройство Emulex OneConnect с обозначением в системе `ens15f1`.

Выяснить количество поддерживаемых VF можно тремя путями из документации производителя устройства, через lspci и через псевдоФС по имени устройства в системе.

6.10.2.3.1 Использование lspci

Чтобы узнать количество VF на контроллере этим способом, нам потребуется выяснить его BDF (bus/device/function) — это число формата `xx:yy:z` в начале строки вывода:

```
# lspci -vv | grep -i emulex
03:00.0 Ethernet controller: Emulex Corporation OneConnect NIC (Skyhawk) (rev
11)
```

Используя BDF устройства, выясним нужный параметр:

```
# lspci -vvs 03:00.0 |grep VFs
Initial VFVFss: 64, Total VFs: 64, Number of VFs: 63, Function Dependency Link:
01
```

Из вывода нас интересует параметр `Number of VFs: 63`.

6.10.2.3.2 Использование псевдоФС

Мы знаем, что имя устройства в системе — `ens15f1`. Нужный параметр хранится в файле `sriov_totalvfs:`

```
# cat /sys/class/net/ens15f1/device/sriov_totalvfs
63
```

6.10.2.3.3 Активация VF

Активировать поддержку VF можно, добавив количество используемых VF в файл `sriov_numvfs` и перезапустив интерфейс:

```
# echo 63 > /sys/class/net/ens15f1/device/sriov_numvfs
# ip link set up ens15f1
```

6.10.2.3.4 Настройка автозагрузки параметров

Для того, чтобы после каждой перезагрузки узла не приходилось активировать VF вручную, добавьте активацию функции в конфиг загрузки интерфейса:

```
# ln -fs /etc/init.d/net.lo /etc/init.d/net.ens15f1
# cat << EOF >> /etc/conf.d/net
config_ens15f1="null"
postup() {
    if [[ ${IFACE} == "ens15f1" ]];then
        echo 63 > /sys/class/net/${IFACE}/device/sriov_numvfs
    fi
    return 0
}
EOF
# rc-update add net.ens15f1
```

6.10.3. Настройка проброса

Для начала разрешим проброс устройств виртуальные машины. Для этого добавим новую секцию в `/etc/nova/nova.conf` — конфигурационный файл службы управления виртуальными машинами OpenStack Nova:

```
[pci]
passthrough_whitelist = { "devname": "ens15f1", "physical_network": "srnet"}
```

Чтобы изменения вступили в силу, перезапустим службу:

```
rc-config restart nova-compute
```

Создадим дополнительный конфиг `/etc/neutron/sriov-agent.ini` для службы управления сетями OpenStack Neutron:

```
[securitygroup]
firewall_driver = neutron.agent.firewall.NoopFirewallDriver

[sriov_nic]
physical_device_mappings = srnet:ens15f1
exclude_devices =

[agent]
root_helper = sudo neutron-rootwrap /etc/neutron/rootwrap.conf
```

Обратите внимание: с тегом физической сети `srnet` ассоциируется адаптер `ens15f1`.

Запустим агент, отвечающий за SR-IOV:

```
# rc-update add neutron-sriov-nic-agent
# rc-config start neutron-sriov-nic-agent:
```

Добавим в конфиг `/etc/neutron/plugin.ini` на всех узлах с ролями «Физический узел» и «Управление сетями» настройки проброса:

```
[ml2]
mechanism_drivers = <СУЩЕСТВУЮЩИЕ ПАРАМЕТРЫ>,sriovnicswitch
[ml2_type_vlan]
network_vlan_ranges = <СУЩЕСТВУЮЩИЕ ПАРАМЕТРЫ>,srnet:3180:3189
```

```
[ml2_sriov]
supported_pci_vendor_devs = 10df:0720
[ovs]
bridge_mappings = <СУЩЕСТВУЮЩИЕ ПАРАМЕТРЫ>,srnet:srnet
[agent]
extensions = <СУЩЕСТВУЮЩИЕ ПАРАМЕТРЫ>,fdb
[FDB]
shared_physical_device_mappings = srnet:ens15f1
```

Значения `supported_pci_vendor_devs` для конкретной инсталляции можно получить из вывода:

```
# cat /sys/class/net/ens15f1/device/{vendor,device}
0x10df
0x0720
```

6.10.3.1. Частные случаи

6.10.3.1.1 Неиспользованный Ethernet-контроллер

При использовании отдельного Ethernet-контроллера, не использованного при установке ПВ РУСТЭК, потребуется создать сетевой мост в Open vSwitch с дополнительным контроллером:

```
# ovs-vsctl add-br srnet
# ovs-vsctl add-port srnet ens15f1
```

Перезапустим службу `neutron-openvswitch-agent` для применения изменений:

```
# rc-config restart neutron-openvswitch-agent
```

6.10.3.1.2 Фильтр планировщика Nova

Чтобы планировщик, отвечающий за размещение виртуальных машин на узлах, учитывал машины с адаптерами SR-IOV, в конфиге `/etc/nova/nova.conf` должен учитываться фильтр `PciPassthroughFilter` в секции планировщика `filter-scheduler` на узлах с ролью «Управление ВМ»:

```
[filter_scheduler]
enabled_filters = <СУЩЕСТВУЮЩИЕ ПАРАМЕТРЫ>,PciPassthroughFilter
```

Применим изменения:

```
rc-config restart nova-api nova-scheduler
```

6.10.4. Подключение

6.10.4.1. Создание сети

Создадим отдельную сеть для устройств SR-IOV:

```
# openstack network create --provider-network-type=vlan --provider-physical-
network=srnet sriovnet1
```

Где `provider-physical-network` ссылается на тег `srnet` из конфига `/etc/neutron/sriov-agent.ini`.

Для подключения виртуальных машин потребуется подсеть в сети sriovnet1: создайте её с корректными индивидуальными настройками сети.

6.10.4.2. Подключение ВМ

Чтобы подключить ВМ с использованием SR-IOV в новую подсеть, в ней нужно создать порт с типом `VNIC` «Напрямую» — этот тип выбирается при создании порта в подсети. Если вы не указываете конкретный IP-адрес, запомните адрес, который будет присвоен порту автоматически — по нему вы сможете определить нужный порт при подключении существующей ВМ к новой подсети: из диалогового окна создания ВМ этого сделать невозможно, только из подключения.

Чтобы создать ВМ, который сразу будет использовать SR-IOV-порт:

```
    # openstack server create --volume mydisk --flavor 1 --nic port-id=<UUI SR-IOV
порта> mytestvm
```

6.10.5. Ограничения SR-IOV

Использование технологии SR-IOV накладывает определённые ограничения на ВМ с портами такого типа:

- при использовании Quality of Service (QoS), опция `max_burst_kbps` не поддерживается, а `max_kbps` округляется до Mbps;
- не поддерживаются группы безопасности;
- невозможна живая миграция.

Также существует важное ограничение платформы: при запуске переконфигурации ПВ РУСТЭК параметры в `/etc/nova/nova.conf` и `/etc/neutron/plugin.ini` придётся добавлять заново.

6.11. QoS

QoS (Quality of Service) — это набор политик и их правил, гарантирующих стабильное качество некоторых сетевых параметров, таких, как пропускная способность или величина задержки в сети. Маркировка трафика определённым в рамках стандарта QoS способом позволяет сетевым устройствам обрабатывать его с более высоким приоритетом, чем немаркированный трафик. В ситуациях, когда весь трафик не маркирован, сетевые устройства обрабатывают весь трафик одинаково, что приводит к коллизиям при высоких нагрузках.

6.11.1. Типы правил QoS

Типы сетевых устройств, поддерживаемые правила и направления трафика с точки зрения виртуальной машины:

Правило \ Устройство	Open vSwitch	SR-IOV	сетевой мост Linux	Описание
`bandwidth_limit`	egress | ingress	egress	egress | ingress	ограничение пропускной способности сетей, портов и плавающих IP
`minimum_bandwidth`	N/A	egress	N/A	минимальное ограничение пропускной способности для определённых типов трафика
`dscp_marking`	egress	N/A	egress	маркировка трафика тегами DSCP

6.11.2. Создание политики QoS

Первым делом, создадим политику и правило для ограничения пропускной способности:

```
# openstack network qos policy create bw-limiter
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| description |                                      |
| id          | d681fdd9-abee-40ea-bcf4-1d95a34a2187 |
| is_default  | False                                |
| name        | bw-limiter                           |
| project_id  | 6112b477a55f4139b1d53feb071c169c     |
| rules       | []                                   |
| shared      | False                                |
| tags        | []                                   |
+-------------+--------------------------------------+

# openstack network qos rule create --type bandwidth-limit --max-kbps 3000 \
    --max-burst-kbits 2400 --egress bw-limiter
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| direction      | egress                               |
| id             | 09d72cf4-36a1-40eb-ae01-e38e2b53d602 |
| max_burst_kbps | 2400                                 |
| max_kbps       | 3000                                 |
| name           | None                                 |
| project_id     |                                      |
+----------------+--------------------------------------+
```

При настройке правил, регулирующих полосу пропускания, как в примере выше, особое внимание нужно уделить опции `max_burst` — это важный параметр, отвечающий за корректную работу лимитеров в Open vSwitch и сетевом мосте Linux. Рекомендуется устанавливать значение этого параметра в 80% от ширины заданной полосы пропускания.

6.11.3. Назначение QoS на порт ВМ

В нашем примере мы будем использовать ВМ с IP-адресом `10.10.10.65`. Определим его порт и назначим политику:

```
# openstack port list
+--------------------------------------+-------------------------------------------------------------------------------+
| ID                                   | Fixed IP Addresses                                                            |
+--------------------------------------+-------------------------------------------------------------------------------+
| 26615f9c-bd7c-4b01-988b-9b055f13627d | ip_address='10.10.10.65', subnet_id='efc4bf56-b720-4a40-9fc6-d24726295a35'    |
| 3a574685-dd97-47a0-b308-97bc1fcfe1bf | ip_address='10.10.10.219', subnet_id='efc4bf56-b720-4a40-9fc6-d24726295a35'   |
| 474dc147-5bab-4289-bf75-a677c5343382 | ip_address='10.10.10.163', subnet_id='efc4bf56-b720-4a40-9fc6-d24726295a35'   |
+--------------------------------------+-------------------------------------------------------------------------------+

# openstack port set --qos-policy bw-limiter 26615f9c-bd7c-4b01-988b-9b055f13627d
```

6.11.4. Назначение QoS на виртуальную сеть

Назначим созданную политику на сеть `private`:

```
# openstack network set --qos-policy bw-limiter private
```

Когда вы назначаете QoS-политики на сеть, её будут использовать все созданные и подключенные порты ВМ в этой сети. При этом для каждого конкретного порта ВМ политику можно переопределить. Политика не распространяется на служебные порты в этой сети, такие как порты роутера или DHCP.

6.11.5. Политики по умолчанию

Для каждого проекта можно создать одну QoS-политику по умолчанию: она будет автоматически применяться к любой созданной сети в рамках проекта, за исключением случаев, когда другая политика назначена конкретно на сеть.

Создадим политику по умолчанию с именем `default-limiter`:

```
# openstack network qos policy create --default default-limiter
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| description |                                      |
| id          | 3e0d67b6-9b0f-473e-89bd-77fef048add1 |
| is_default  | True                                 |
| name        | default-limiter                      |
| project_id  | 6112b477a55f4139b1d53feb071c169c     |
| rules       | []                                   |
| shared      | False                                |
| tags        | []                                   |
+-------------+--------------------------------------+

# openstack network qos policy list
+--------------------------------------+------------------+--------+---------+----------------------------------+
| ID                                   | Name             | Shared | Default | Project                          |
+--------------------------------------+------------------+--------+---------+----------------------------------+
| 3e0d67b6-9b0f-473e-89bd-77fef048add1 | default-limiter  | False  | True    | 6112b477a55f4139b1d53feb071c169c |
| 87613a59-4fcb-47b5-aefc-37e62dd4e11a | bandwidth-control | False  | False   | 6112b477a55f4139b1d53feb071c169c |
| d681fdd9-abee-40ea-bcf4-1d95a34a2187 | bw-limiter       | False  | False   | 6112b477a55f4139b1d53feb071c169c |
+--------------------------------------+------------------+--------+---------+----------------------------------+
```

Отвяжем политику по умолчанию от проекта:

```
# openstack network qos policy set --no-default default-limiter

# openstack network qos policy list
+--------------------------------------+------------------+--------+---------+----------------------------------+
| ID                                   | Name             | Shared | Default | Project                          |
+--------------------------------------+------------------+--------+---------+----------------------------------+
| 3e0d67b6-9b0f-473e-89bd-77fef048add1 | default-limiter  | False  | False   | 6112b477a55f4139b1d53feb071c169c |
| 87613a59-4fcb-47b5-aefc-37e62dd4e11a | bandwidth-control | False  | False   | 6112b477a55f4139b1d53feb071c169c |
| d681fdd9-abee-40ea-bcf4-1d95a34a2187 | bw-limiter       | False  | False   | 6112b477a55f4139b1d53feb071c169c |
+--------------------------------------+------------------+--------+---------+----------------------------------+
```

Сделаем политику `bw-limiter политикой по умолчанию:`

```
# openstack network qos policy set --default bw-limiter

# openstack network qos policy list
+--------------------------------------+------------------+--------+---------+----------------------------------+
| ID                                   | Name             | Shared | Default | Project                          |
+--------------------------------------+------------------+--------+---------+----------------------------------+
| 3e0d67b6-9b0f-473e-89bd-77fef048add1 | default-limiter  | False  | False   | 6112b477a55f4139b1d53feb071c169c |
| 87613a59-4fcb-47b5-aefc-37e62dd4e11a | bandwidth-control | False  | False   | 6112b477a55f4139b1d53feb071c169c |
| d681fdd9-abee-40ea-bcf4-1d95a34a2187 | bw-limiter       | False  | True    | 6112b477a55f4139b1d53feb071c169c |
+--------------------------------------+------------------+--------+---------+----------------------------------+
```

6.11.6. Назначение QoS на Floating IP

Назначим политику на плавающий IP `172.16.100.18`. Определим ID:

```
$ openstack floating ip list
+--------------------------------------+---------------------+------------------+------
+-----+
| ID                                   | Floating IP Address | Fixed IP Address | Port |
... |
+--------------------------------------+---------------------+------------------+------
+-----+
| 1163d127-6df3-44bb-b69c-c0e916303eb3 | 172.16.100.9        | None             | None |
... |
| d0ed7491-3eb7-4c4f-a0f0-df04f10a067c | 172.16.100.18       | None             | None |
... |
```

```
    | f5a9ed48-2e9f-411c-8787-2b6ecd640090 | 172.16.100.2        | None             | None |
... |
    +--------------------------------------+---------------------+------------------+------
+-----+
```

Привяжем политику к плавающему IP:

```
    # openstack floating ip set --qos-policy bw-limiter d0ed7491-3eb7-4c4f-a0f0-
df04f10a067c
```

Правила ограничения пропускной способности начнут работать только после того, как плавающий IP будет привязан к внутреннему IP ВМ. Привязанная к такому IP политика не будет видна на соответствующем порту ВМ, только в атрибутах плавающего IP.

6.11.7. Назначение QoS роутеру

В случае с роутером политики QoS применяются к его интерфейсу external gateway. Политику можно назначить по имени роутера:

```
    (openstack) router set --qos-policy bw-limiter Router
    (openstack) router show Router -f yaml
    ...SKIP...
    external_gateway_info:   '{"external_fixed_ips":   [{"ip_address":   "10.11.2.15",
"subnet_id":
      "c43712d0-7e72-4e5d-8d7e-804bde3f783d"}],   "enable_snat":   true,   "network_id":
"ccf8260a-672f-4ca1-8893-dc0c92d6929c",
      "qos_policy_id": "d681fdd9-abee-40ea-bcf4-1d95a34a2187"}'
    ...SKIP...
```

6.11.8. DSCP-маркировка пакетов

DSCP (Differentiated Services CodePoint, DSCP) — это число в диапазоне от 0 до 63, которое помещается в заголовок IP-пакета и определяет, к какому классу трафика относится пакет.

Доступные для использования DSCP: `0, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 46, 48 и 56`.

Часть DSCP стандартизована и закреплена за некоторыми классами сервисов:

Service class	DSCP Name	DSCP Value
Network control	CS6	48
Telephony	EF	46
Signaling	CS5	40
Multimedia conferencing	AF41, AF42, AF43	34, 36, 38
Real-time interactive	CS4	32
Multimedia streaming	AF31, AF32, AF33	26, 28, 30
Broadcast video	CS3	24
Low-latency data	AF21, AF22, AF23	18, 20, 22
OAM	CS2	16

Service class	DSCP Name	DSCP Value
High-throughput data	AF11, AF12, AF13	10, 12, 14
Standard	DF	0
Low-priority data	CS1	8

DSCP не работает на плавающих IP-адресах.

При создании политики, использующей DSCP, нужно указать тип и конкретное значение DSCP:

```
(openstack) network qos rule create --type dscp-marking --dscp-mark 22 bw-limiter
```

6.11.8.1. DSCP-маркировка в оверлейных сетях

При использовании оверлейных сетей, таких, как VxLAN, метками DSCP маркируются только внутренние заголовки пакетов, и при инкапсуляции автоматического добавления меток во внешние заголовки не происходит. Вручную это можно сделать двумя путями: установив конкретное значение DSCP для пакетов или используя механизм наследования.

6.11.8.1.1 Конкретное значение

Чтобы установить конкретное значение для внешних заголовков, добавьте опцию `dscp` в секцию `[agent]` конфига `/etc/neutron/plugin.ini`:

```
[agent]
dscp = 8
```

6.11.8.1.2 Наследование

Чтобы значение DSCP копировалось из внутреннего заголовка пакета во внешний, добавьте опцию `dscp_inherit = true в секцию [agent] конфига /etc/neutron/plugin.ini.`

При включённом наследовании опция `dscp` будет проигнорирована.

6.11.9. Управление правилами

Правила в политиках QoS можно менять в любой момент «на лету»: изменения сразу же применятся ко всем портам, ассоциированным с политикой:

```
$ openstack network qos rule set --max-kbps 2000 --max-burst-kbits 1600 \
    --ingress bw-limiter 09d72cf4-36a1-40eb-ae01-e38e2b53d602

$ openstack network qos rule show \
    bw-limiter 09d72cf4-36a1-40eb-ae01-e38e2b53d602
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| direction      | ingress                              |
| id             | 09d72cf4-36a1-40eb-ae01-e38e2b53d602 |
| max_burst_kbps | 1600                                 |
| max_kbps       | 2000                                 |
| name           | None                                 |
| project_id     |                                      |
+----------------+--------------------------------------+
```

6.11.9.1. Множественные правила

Каждая политика может совмещать в себе несколько правил при условии, что правила одного типа настроены на разные направления трафика.

```
# openstack network qos policy create bandwidth-control
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| description |                                      |
| id          | 87613a59-4fcb-47b5-aefc-37e62dd4e11a |
| is_default  | False                                |
| name        | bandwidth-control                    |
| project_id  | 6112b477a55f4139b1d53feb071c169c     |
| rules       | []                                   |
| shared      | False                                |
| tags        | []                                   |
+-------------+--------------------------------------+

# openstack network qos rule create --type bandwidth-limit \
    --max-kbps 50000 --max-burst-kbits 50000 --egress bandwidth-control
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| direction      | egress                               |
| id             | bd24df9e-6e48-4733-afe0-5b89c7bd79da |
| max_burst_kbps | 50000                                |
| max_kbps       | 50000                                |
| name           | None                                 |
| project_id     |                                      |
+----------------+--------------------------------------+

# openstack network qos rule create --type bandwidth-limit \
    --max-kbps 10000 --max-burst-kbits 10000 --ingress bandwidth-control
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| direction      | ingress                              |
| id             | 57eedfe1-6e23-494a-9f64-7fc7142153b9 |
| max_burst_kbps | 10000                                |
| max_kbps       | 10000                                |
| name           | None                                 |
| project_id     |                                      |
+----------------+--------------------------------------+

# openstack network qos rule create --type minimum-bandwidth \
    --min-kbps 1000 --egress bandwidth-control
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| direction  | egress                               |
| id         | 24df2ad7-48df-49ca-983a-ce6d4105e86a |
| min_kbps   | 1000                                 |
| name       | None                                 |
| project_id |                                      |
+------------+--------------------------------------+

# openstack network qos policy show bandwidth-control
+-------------+-------------------------------------------------------------------------------------------------------
-+
| Field       | Value                                                                                                 |
+-------------+-------------------------------------------------------------------------------------------------------
-+
| description |                                                                                                       |
| id          | 87613a59-4fcb-47b5-aefc-37e62dd4e11a                                                                  |
| is_default  | False                                                                                                 |
| name        | bandwidth-control                                                                                     |
| project_id  | 6112b477a55f4139b1d53feb071c169c                                                                      |
| rules       | [{'max_kbps': 50000, 'max_burst_kbps': 50000, 'direction': 'egress', 'id':                            |
|             | 'bd24df9e-6e48-4733-afe0-5b89c7bd79da', 'qos_policy_id':                                              |
|             | '87613a59-4fcb-47b5-aefc-37e62dd4e11a', 'type': 'bandwidth_limit'},                                   |
|             | {'max_kbps': 10000, 'max_burst_kbps': 10000, 'direction': 'ingress', 'id':                            |
|             | '57eedfe1-6e23-494a-9f64-7fc7142153b9', 'qos_policy_id':                                              |
|             | '87613a59-4fcb-47b5-aefc-37e62dd4e11a', 'type': 'bandwidth_limit'},                                   |
|             | {'min_kbps': 1000, 'direction': 'egress', 'id': '24df2ad7-48df-49ca-983a-ce6d4105e86a',               |
|             | 'qos_policy_id': '87613a59-4fcb-47b5-aefc-37e62dd4e11a', 'type': 'minimum_bandwidth'}]                |
| shared      | False                                                                                                 |
| tags        | []                                                                                                    |
```

```
+------------+-------------------------------------------------------------------------------------------
-+
```

6.11.10. Удаление политики QoS

Чтобы удалить политику с порта, нужно знать только его ID:

```
# openstack port unset --qos-policy 26615f9c-bd7c-4b01-988b-9b055f13627d
```

Чтобы полностью удалить политику:

```
# openstack network qos policy delete bw-limiter
```

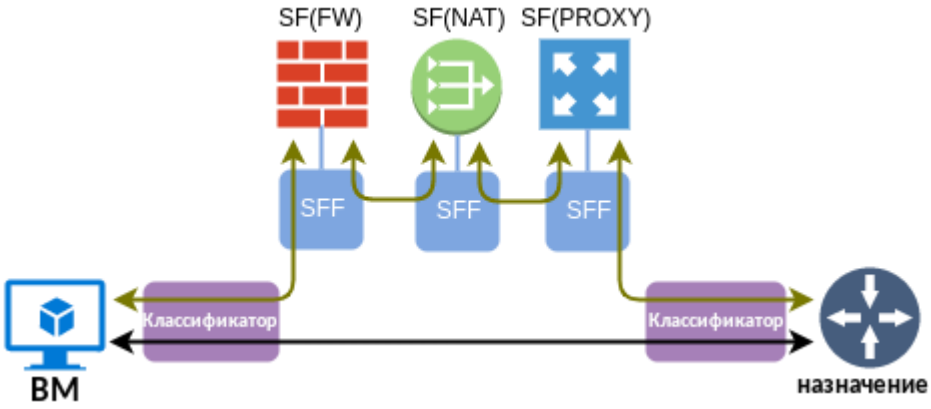
6.12. Service Function Chaining

6.12.1. Описание

Цепи Служебных Функций (Service Function Chaining, SFC) — метод перенаправления трафика, который направляет пакеты через определенный набор конечных точек — служебных функций, отменяя стандартную переадресацию на основе пункта назначения. Отталкиваясь от работы с традиционными сетями, SFC можно рассматривать как набор экземпляров маршрутизации на основе политик, управляемых центральным элементом (контроллером SDN). Типичными вариантами использования SFC могут быть такие вещи, как брандмауэр, IDS/IPS, прокси, NAT, мониторинг.

Распространенным представлением SFC является ориентированный (ациклический) граф. Первый и последний элементы графа являются источником и получателем, которых нужно связать, а каждая вершина внутри графа представляет собой служебную функцию. IETF RFC7665 определяет эталонную архитектуру для реализаций SFC и устанавливает некоторые основные термины. Упрощенная архитектура SFC состоит из следующих основных компонентов:

- Классификатор — элемент сети, который сопоставляет и перенаправляет пакеты в цепь служебных функций;
- Служебная функция (SF) — элемент, отвечающий за обработку пакетов;
- Маршрутизатор служебных функций (SFF) — элемент сети, который перенаправляет трафик в/из напрямую подключенной служебной функции.



Одним из важных свойств служебных функций является эластичность. В пул служебных функций можно добавить больше экземпляров одного и того же типа, и маршрутизатор служебных функций будет распределять трафик между ними. По этой причине маршрутизатор служебных функций рассматривает подключения к служебным функциям как группу портов, а не как один порт.

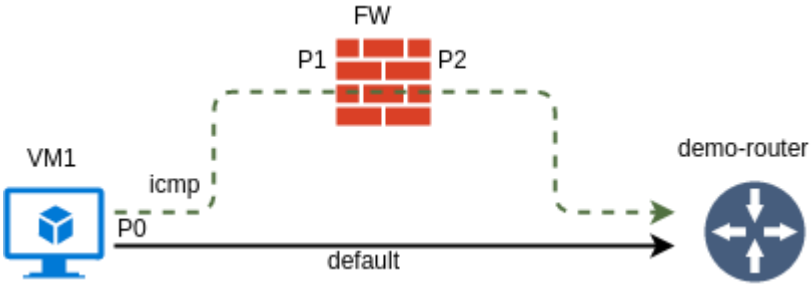
Для передачи информации по цепи служебных функций используется RFC7665, который определяет требования к заголовку «SFC Encapsulation», который можно использовать для уникальной

идентификации экземпляра цепи и указания общих метаданных для всех её элементов. Neutron API называет инкапсуляцию SFC *корреляцией*, поскольку её основной функцией является идентификация пути конкретной служебной функции. В ПВ РУСТЭК используется реализация инкапсуляции SFC — MPLS, которая используется текущим драйвером агента OVS.

6.12.2. Ограничения

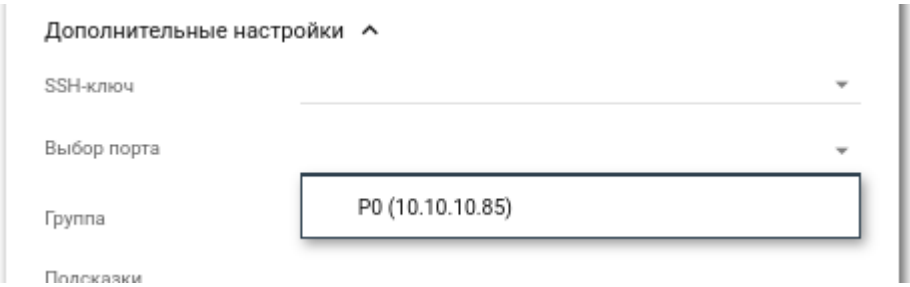
SFC работает только в сетях VXLAN.

6.12.3. Пример настройки SFC в ПВ РУСТЭК



А) Создать 3 порта: P0 и P1 P2 для ВМ FW без "безопасности портов";

Б) Создать ВМ с именем VM1 и при создании указать порт P0;



В) Далее необходимо создать ВМ с именем FW и указать порты: P1 и P2;

Г) Затем нам нужно создать пару входных и выходных портов и назначить её группе пар портов;

```
    openstack sfc port pair create --ingress P1 --egress P2 PPAIR
    openstack sfc port pair group create --port-pair PPAIR PPGROUP
```

Группа пар портов также позволяет указать заголовки L2-L4, которые следует использовать для балансировки нагрузки в группах OpenFlow, переопределяя поведение по умолчанию, описанное в следующем разделе;

Д) Еще одним обязательным элементом является классификатор потока. Мы будем перенаправлять ICMP-трафик, поступающий с порта P0 VM1;

```
    openstack sfc flow classifier create --protocol icmp --logical-source-port P0
FLOW-ICMP
```

Е) Наконец, мы можем связать классификатор потоков с ранее созданной группой пар портов. Значение по умолчанию для параметра корреляции в этом случае равно `mpls`. Это означает, что каждая цепь будет иметь свою уникальную метку MPLS, которая будет использоваться в качестве инкапсуляции SFC;

```
    openstack sfc port chain create --port-pair-group PPGROUP --flow-classifier
FLOW-ICMP PCHAIN
```

Если после этих действий вы войдете в консоль VM1 и попытаетесь пропинговать шлюз по умолчанию, это не удастся, т.к. пакеты icmp перенаправлены на порт BM FW. Если на BM FW запустить анализатор трафика, то можно увидеть пакеты icmp с BM VM1.

6.12.4. Реализация SFC в конвейере пересылки OVS

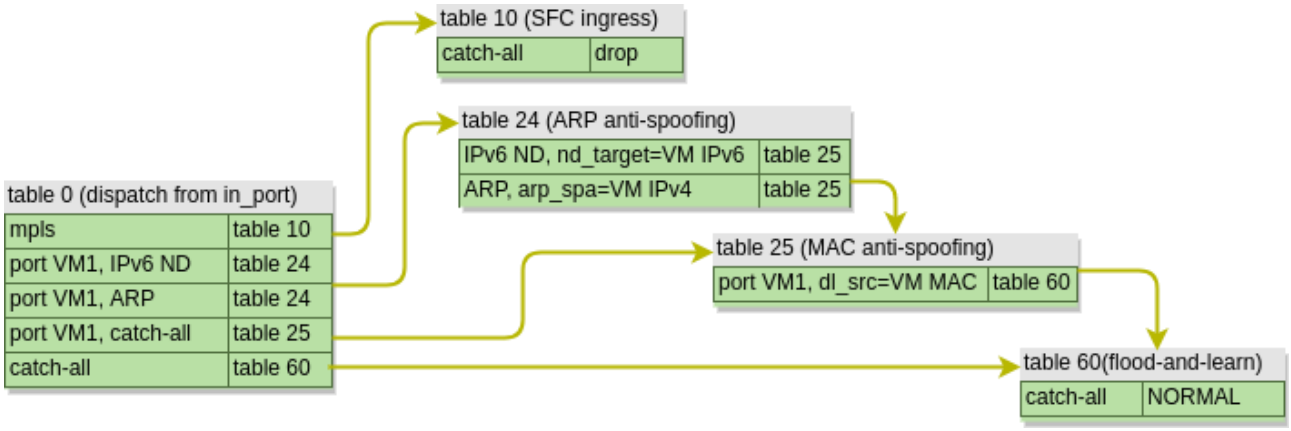
В примере выше VM1, и FW находятся на одном физическом узле и затрагивает flows на br-int. Для отображения всех правил для пакетов для br-int можно воспользоваться командой

```
ovs-ofctl dump-flows br-int --no-stats
```

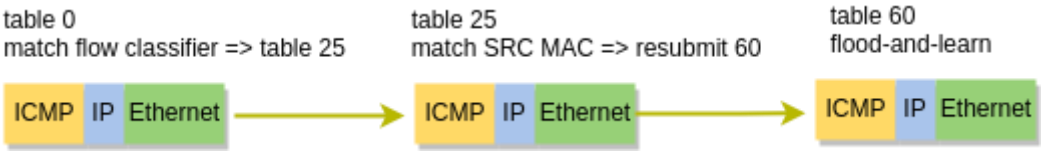
Узнать имена и номера интерфейсов внутри OVS для BM VM1 и FW по их MAC можно командой:

```
ovs-ofctl dump-ports-desc br-int
```

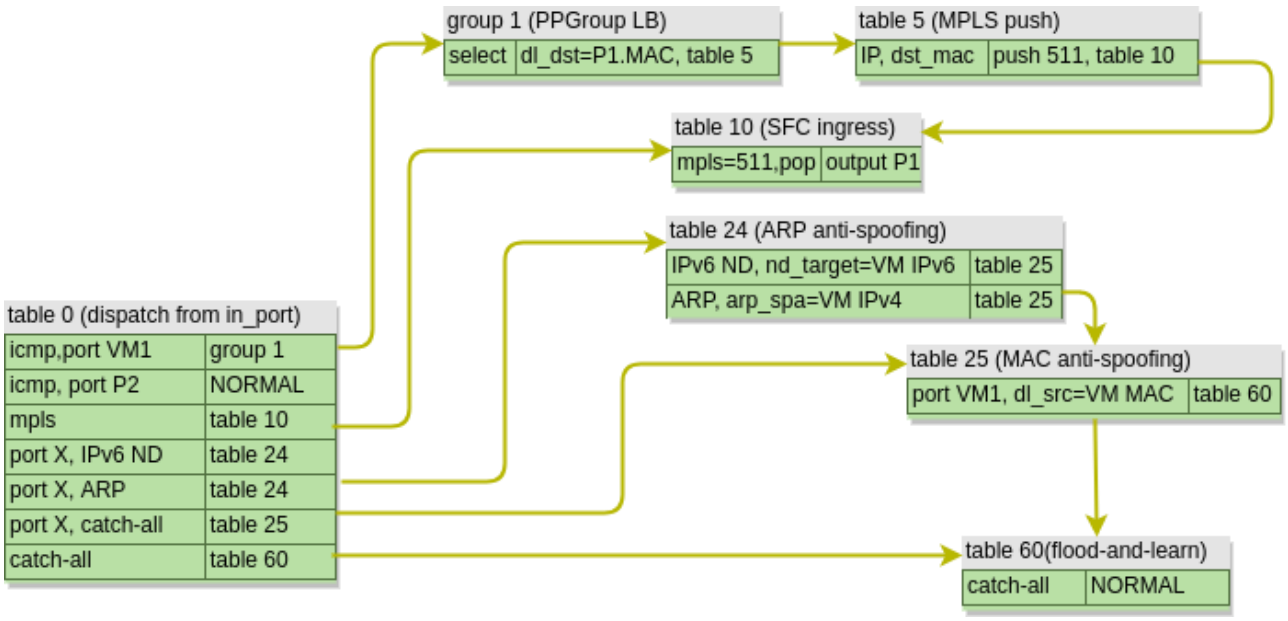
На схеме ниже (схема нарисована по выводу правил: ovs-ofctl dump-flows br-int --no-stats) отображены правила OpenFlow в br-int до настройки SFC для BM VM1.



Структура таблицы проста, поскольку интеграционный мост br-int в основном опирается на изучение MAC уровня данных. Несколько таблиц защиты от спуфинга MAC и ARP проверят действительность пакета и отправят его в таблицу 60, где действие `NORMAL` вызовет поведение «наводни и изучи». Таким образом, ICMP-пакет, исходящий от VM1, будет проходить по следующему пути:

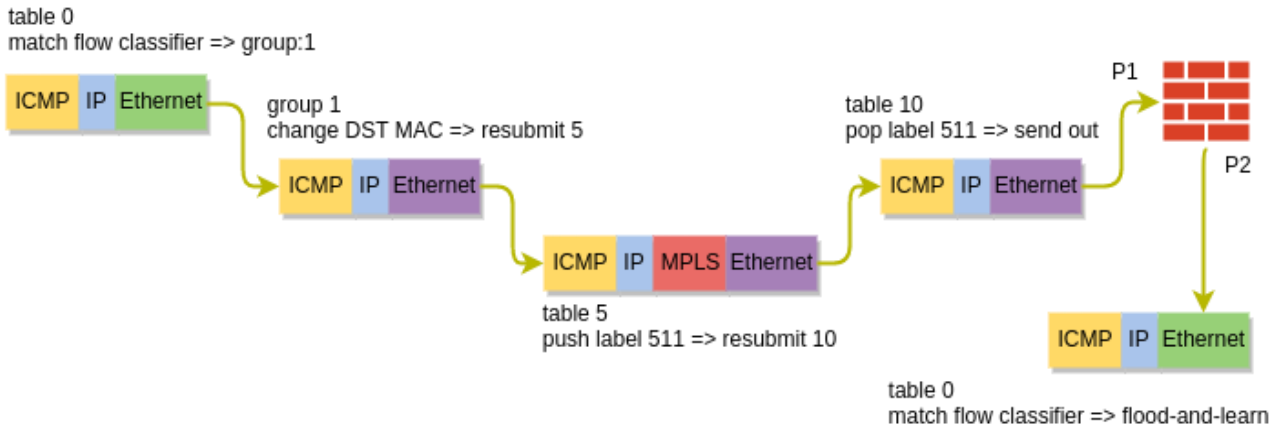


После того, как мы настроили SFC, конвейер пересылки изменился и теперь выглядит так:



Во-первых, мы видим, что таблица 0 действует как классификатор, перенаправляя пакеты, которые в примере выше мы создавали командой **openstack sfc flow classifier create** в сторону group 1. Эта группа представляет собой группу OpenFlow типа select, которая распределяет трафик между несколькими пунктами назначения. По умолчанию OVS будет использовать комбинацию заголовков L2-L4, для вычисления хэша, определяющего выходной блок, аналогично тому, как балансировка нагрузки для каждого потока работает в традиционных маршрутизаторах и коммутаторах. Это поведение можно переопределить с помощью определенного набора заголовков в lb_fields настройках группы пар портов.

В нашем случае у нас есть только одна служебная функция, поэтому MAC-адрес назначения пакета обновляется до MAC-адреса входного порта служебной функции и перенаправляется в новую таблицу 5. В таблице 5 все пакеты, предназначенные для служебной функции, объединяются с одной меткой MPLS, однозначно идентифицирующей путь служебной функции. Затем пакет перенаправляется в таблицу 10. Именно здесь пакеты распределяются по входным портам служебной функции на основе назначенной метки MPLS.



После обработки служебной функцией пакет покидает исходящий порт P2 и снова входит в br-int. Эта временная таблица 0 знает, что пакет уже обработан служебной функцией, и, поскольку правила защиты от спуфинга отключены, просто рассылает пакет через все порты в той же VLAN. Пакет пересылается на br-tun, где он реплицируется и доставляется на узел контроллера в соответствии с поведением по умолчанию.

6.12.5. Описание команд создания правил SFC

6.12.5.1. openstack sfc port pair create

Данной командой мы объединяем пару портов во входной (--ingress) и выходной (--egress).

6.12.5.2. openstack sfc port pair group create

Включение в группы пары портов.

Группа пар портов может содержать одну или несколько пар портов. Несколько пар портов обеспечивают балансировку/распределение нагрузки по набору функционально эквивалентных служебных функций.

Опции:

`--port-pair` — пара портов, данный параметр можно указывать несколько раз для включения в группу нескольких пар;

`--port-pair-group-parameters lb-fields` — разделенный список полей балансировки нагрузки.

6.12.5.3. openstack sfc flow classifier create

Правила определения источника потока, определяет назначение потока. Атрибут `l7_parameters` является заполнителем, который может использоваться для поддержки классификации потоков с использованием полей уровня 7, таких как URL-адрес. Опции `logical_source_port` и `logical_destination_port` по умолчанию имеют значение `none`, опция `ethertype` по умолчанию имеет значение `IPv4`.

Опции:

`--ethertype` — IPv4/IPv6;

`--protocol` — IP-протокол;

`--source_port_range_min` — минимальный порт исходного протокола;

`--source_port_range_max` — максимальный порт исходного протокола;

`--destination_port_range_min` — минимальный порт протокола назначения;

`--destination_port_range_max` — максимальный порт протокола назначения;

`--source_ip_prefix` — исходный IP-адрес или префикс;

`--destination_ip_prefix` — IP-адрес назначения или префикс;

`--logical_source_port` — исходный порт;

`--logical_destination_port` — порт назначения;

`--l7_parameters` — словарь параметров L7.

6.12.5.4. openstack sfc port chain create

Данной командой мы связываем правила потоков с ранее созданной группой пар портов.

`--project_id` — идентификатор проекта;

`--port_pair_groups` — группа пар портов;

`--flow_classifiers` — имя или ID правила потока;

--chain_parameters — параметры цепи, содержит один или несколько параметров для цепи портов (в настоящее время поддерживается один параметр correlation, который по умолчанию имеет значение mpls для согласованности с возможностями OVS).

6.12.6. Внешние источники

Описание всех консольных команд и опций: https://docs.openstack.org/python-neutronclient/6.4.0/cli/osc/v2/networking-sfc.html

Настройка и примеры: https://docs.openstack.org/neutron/latest/admin/config-sfc.html

OpenStack SDN — Skydiving Into Service Function Chaining https://networkop.co.uk/blog/2017/09/15/os-sfc-skydive/

7. Образы

7.1. Свойства

При создании образа указываются свойства, которые могут полезными для создаваемых ВМ. Задать эти свойства можно в форме **Создание образа**. При нажатии на кнопку **Дополнительные настройки** в форме появятся дополнительные настройки.

Описание свойств в форме **Создание образа** для заполнения представлено в **Руководстве пользователя**, раздел 5.1. Ниже представлено дополнительное описание некоторых свойств:

- **Имя ОС** – наименование операционной системы образа. Используется для привязки к образу инит-скрипта, отрабатывающего при создании ВМ. Для добавления инит-скрипта данное свойство должно быть обязательно заполнено. После добавления к какому-либо образу инит-скрипта все образы, имеющие такое же значение свойства **Имя ОС** как в этом образе, будут использовать тот же инит-скрипт. При удалении всех образов с одинаковым свойством **Имя ОС**, инит-скрипт остаётся привязанным к имени и после добавления нового образа с этим именем, он будет сразу иметь инит-скрипт;
- **RAM (МБ)** – минимальный размер оперативной памяти в мегабайтах, который должен выделяться ВМ, создаваемой из образа с таким свойством. Данное свойство учитывается при создании ВМ: в окне создания ВМ будут доступны только конфигурации, которые имеют размер оперативной памяти равный или больше заданного свойства. При оставлении нулевым значением ограничение размера не задаётся и для создания ВМ будут доступны все конфигурации;
- **Размер диска (ГБ)** – минимальный размер диска в гигабайтах, создаваемому из образа с таким свойством. Данное свойство учитывается при создании диска или ВМ: при выбора в окне создания диска или ВМ образа с таким свойством параметр **Размер диска** будет автоматически установлен этим значением. При оставлении нулевым значением ограничение размера не задаётся и размер диска может быть задан любым;
- **Сетевой адаптер** – выбор из списка сетевого адаптера, используемого при создании ВМ. Для выбора доступны следующие сетевые адаптеры:
    - **virtio** – паравиртуализированный сетевой адаптер. Он используется для достижения максимальной доступной пропускной способности. Рекомендуемый для использования. Все операционные системы на основе Linux поставляются с драйверами VirtIO. Для Windows VirtIO драйвер может быть загружен с веб-сайтов http://www.linux-kvm.org/page/WindowsGuestDrivers/Download_Drivers или https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/, для Mac OS — с веб-сайта https://github.com/pmj/virtio-net-osx. В некоторых облачных образах Windows драйвер VirtIO уже встроен, например, в образе, который можно скачать отсюда https://cloudbase.it/windows-cloud-images/. Используется по умолчанию;
    - **e1000** – сетевой гигабитный адаптер Intel 82545EM. Он является драйвером ядра Linux, применяемым для виртуализации виртуальных сетевых интерфейсов на основе архитектуры Intel. Поддерживается всеми основными операционными системами, такими как Linux, Windows и Mac OS X без необходимости установки дополнительных драйверов;
    - **rtl8139** – сетевой 100-мегабитный адаптер Realtek RTL8139;
- **Дисковый контроллер** – выбор из списка дискового контроллера, используемого при создании ВМ. Доступны для выбора следующие дисковые контроллеры:
    - **virtio** – паравиртуализированный дисковый контроллер. Он более производительный, чем ide и sata. Диск virtio внутри ВМ виден как устройство PCI Express. Диски в Linux при использовании данного контроллера именуются как /dev/vda, /dev/vdb и т. д. Используется по умолчанию;
    - **virtio-scsi** – паравиртуализированный дисковый контроллер SCSI. Он предоставляет адаптер шины SCSI для ВМ. SCSI предлагает более богатый набор команд, чем virtio, и поддерживает больше вариантов использования. Virtio-SCSI разработан для замены virtio, позволяет напрямую подключиться к SCSI LUNам и значительно улучшает масштабируемость по сравнению с virtio. Так, virtio-scsi поддерживает сотни устройств, в то время как virtio поддерживает не более 28 устройств. Кластерные диски можно создавать только с данным контроллером, так как virtio не поддерживает команды постоянного резервирования SCSI-3. Имеет полную поддержку горячего подключения дисков, тогда как для дисков virtio это не всегда возможно, потому что для них должна быть поддержка горячего подключения устройств PCI Express со стороны гостевой

ОС. Диски в Linux при использовании данного контроллера именуются как /dev/sda, /dev/sdb и т. д.;
    - **ide** – дисковый контроллер IDE. Он наименее производительный из всех доступных. Для корректного подключения диска, созданного из образа с данным контроллером, к ВМ, необходимо в меню образа **Дополнительные настройки** в списке **Тип ВМ** выбрать **pc-i440fx-5.2**. Диски в Linux при использовании данного контроллера именуются как /dev/sda, /dev/sdb и т. д.;
    - **sata** – дисковый контроллер SATA. Замена контроллеру ide для ВМ с чипсетом q35, используемым по умолчанию. Он менее производительный, чем virtio и virtio-scsi, но более производительный, чем ide. Диски в Linux при использовании данного контроллера именуются как /dev/sda, /dev/sdb и т. д.;
- **Публичный** – включение свойства делает образ доступным для использования пользователями всех проектов. Если свойство выключено, то образ будет доступен только в проекте, в котором он был создан, а также администраторам любых проектов;
- **Улучшения Windows** – после включения в свойства образа добавляется параметр **os_type** равный **windows**. Для создаваемых ВМ с использованием такого образа в конфигурационный файл XML будут внесены изменения, способствующие улучшению их работы на загруженных гипервизорах, что более всего актуально для Windows 7 и Windows Server 2008 R2:

```
<features>
    <acpi/>
    <apic/>
    <hyperv>
      <relaxed state='on'/>
      <vapic state='on'/>
      <spinlocks state='on' retries='8191'/>
    </hyperv>
  </features>
  ....
  <clock offset='localtime'>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='hpet' present='no'/>
    <timer name='hypervclock' present='yes'/>
  </clock>
```

Также будет создан раздел подкачки на базе FAT32 вместо раздела подкачки Linux, а имя узла будет ограничено 15 символами;

- **Защищенный** – включение свойства делает невозможным удаление образа, до тех пор, пока оно не будет выключено;
- **Загрузчик UEFI** – свойство включается при необходимости создания ВМ с использованием образа, поддерживающего загрузку по UEFI;
- **QEMU агент** – включение поддержку гостевого агента QEMU. Он обеспечивает доступ между физическими узлами и гостевыми ОС через сокет, используя протокол QMP. Например, может использоваться для создания консистентных резервных копий ВМ. Без включения данного свойства агент не может быть запущен в гостевой ОС;
- **Действия при зависании гостевой ОС** – выбор из списка поведения сторожевого таймера для ВМ Linux, созданного из образа (задаётся через параметр hw_watchdog_action):
    - **Нет** – свойство не задаётся (параметр hw_watchdog_action отсутствует). Используется по умолчанию;
    - **Отключено** – сторожевой таймер не эмулируется. Позволяет отключить таймер для ВМ, созданного из такого образа, даже если оно было включено в конфигурации;
    - **Перезагрузка** – принудительная перезагрузка ВМ;
    - **Выключение** – принудительное выключение ВМ;
    - **Приостановка** – приостановка ВМ;
    - **Ничего не делать** – эмулирование сторожевого таймера и ничего не делать, если ВМ завис;
- **Тип ВМ** – выбор из списка чипсета, используемого при создании ВМ:
    - **pc-i440fx-5.2** – эмулирование чипсета i440FX. Имеет поддержку шины Legacy PCI;
    - **q35** – эмулирование более современного чипсета Q35 (ICH9). Имеет поддержку шины PCI Express. Используется по умолчанию.

7.2. Использование готовых образов

Большинство популярных операционных систем (ОС) имеют готовые образы, предназначенные для создания ВМ, работающих под управлением гипервизора KVM:

- Arch Linux – https://geo.mirror.pkgbuild.com/images/ (логин по умолчанию arch);
- Astra Linux – https://dl.astralinux.ru/images/ (логин по умолчанию astra);
- CentOS – https://cloud.centos.org/centos/ (логин по умолчанию centos, в CentOS Stream 9 – cloud-user);
- Cirros – http://download.cirros-cloud.net/ (логин по умолчанию cirros);
- Debian – https://cdimage.debian.org/cdimage/cloud/ (логин по умолчанию debian);
- Fedora – https://alt.fedoraproject.org/cloud/ (логин по умолчанию fedora);
- FreeBSD, OpenBSD и NetBSD – https://bsd-cloud-image.org/ (логины по умолчанию: freebsd для FreeBSD, openbsd для OpenBSD и netbsd для NetBSD);
- OpenSUSE – https://download.opensuse.org/repositories/Cloud:/Images:/ (логин по умолчанию opensuse);
- SLES – https://www.suse.com/download/sles/ (логин по умолчанию opensuse);
- Ubuntu – https://cloud-images.ubuntu.com/ (логин по умолчанию ubuntu);
- Windows – https://cloudbase.it/windows-cloud-images/ (логин по умолчанию Administrator).

Это образы с настроенной копией ОС. Из готового образа можно быстро создать ВМ с типовой конфигурацией. Рекомендуется использовать 64-битные образы в формате qcow2 (файл образа может иметь расширения qcow2, qcow2c, img), подготовленные для платформы OpenStack. Например, для ОС CentOS в названии образа должно быть **GenericCloud**, а для ОС Debian – **openstack** или **genericcloud**. Такие образы содержат инструмент cloud-init, с помощью которого во время загрузки ВМ получает параметры конфигурации из метаданных и запускает инит-скрипт. Инит-скрипт, привязанный к образу, для ВМ доступен по адресу http://169.254.169.254/latest/user-data или находится на специальном диске с метаданными (подробнее читать в разделе 7.4).

Для большинства готовых образов пароль не установлен. Поэтому, чтобы аутентифицироваться в ОС, нужно использовать SSH-ключ, который должен быть указан при создании ВМ.

Как настроить аутентификацию в ОС с помощью логина и пароля для готовых образов Linux читайте в разделе 7.4.

7.3. Создание

Создание образов разных операционных систем (ОС) на платформе виртуализации РУСТЭК выполняется одной и той же последовательностью действий, однако имеет небольшие различия.

7.3.1. Windows

В данном разделе описывается последовательность действий на примере создания образа ОС Windows 10.

7.3.1.1. Создание образа для установки Windows

Для создания образа нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- нажать кнопку **Создать** ➕ на панели инструментов;
- заполнить поля формы **Создание образа** следующим образом:
  - **Имя** – ввести имя Windows 10 ISO;
  - **Проект** – выбрать проект admin;
  - **Имя ОС** – ввести имя windows10_iso;
  - **Контейнер** – выбрать bare;
  - **Формат диска** – iso;
  - **RAM** – 0;

- **Размер диска** – 0;
- **Сетевой адаптер** – virtio;

Для образов Windows в поле **Дисковый контроллер** рекомендуется выбрать virtio-scsi и установить флажок в чекбоксе **Улучшения Windows**.

- **Публичный** – установить флажок;
- **Метод загрузки** – выбрать Файл;

- нажать кнопку **Создать**.

Созданный образ отобразится в разделе меню **Копии и образы – Образы**.

7.3.1.2. Загрузка образа ISO для установки Windows

Для загрузки образа ISO нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- выбрать созданный образ из предыдущего раздела;
- нажать кнопку **Загрузить образ** на панели инструментов;
- в открывшейся форме **Загрузка образа** добавить файл образа ISO Windows 10);
- после загрузки кнопка **ЗАГРУЗИТЬ** станет активна, нажать на неё.

7.3.1.3. Создание образа для диска драйверов

Образ ISO с драйверами VirtIO доступен для скачивания по ссылке https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso.

Для создания образа нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- нажать кнопку **Создать** на панели инструментов;
- заполнить поля формы **Создание образа** следующим образом:
  - **Имя** – ввести имя VirtIO ISO;
  - **Проект** – выбрать проект admin;
  - **Имя ОС** – ввести имя virtio_iso;
  - **Контейнер** – выбрать bare;
  - **Формат диска** – iso;
  - **RAM** – 0;
  - **Размер диска** – 0;
  - **Сетевой адаптер** – virtio;
  - **Дисковый контроллер** – virtio-scsi;
  - **Публичный** – установить флажок;
  - **Улучшения Windows** – убрать флажок;
  - **Метод загрузки** – выбрать URL;
  - **URL** – ввести https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso;
- нажать кнопку **Создать**.

Созданный образ отобразится в разделе меню **Копии и образы – Образы**.

7.3.1.4. Создание эталонного ВМ Windows

Для создания эталонного ВМ Windows нужно:

- перейти в раздел **Виртуальные Машины**;
- нажать кнопку **Создать** на панели инструментов;
- заполнить поля формы **Создание ВМ** следующим образом:
  - **Имя** – ввести имя Windows install;
  - **Проект** – выбрать проект admin;
  - **ОС** – выбрать Windows 10 ISO;
  - **Конфигурация** – выбрать конфигурацию (например, medium (2 CPU/4 Гб RAM));
  - **Размер диска** – 40;
  - **Тип диска** – выбрать тип (например, nfs);
  - **Сети** – выбрать сеть (например, int-net);
  - **Профили безопасности** – выбрать профили (например, default);
  - **ISO образ** – выбрать образ VirtIO ISO;
  - **ип контроллера для CD** – выбрать SATA;

Нужно обязательно снять флажок в чекбоксе **Удалять диск вместе с ВМ**.

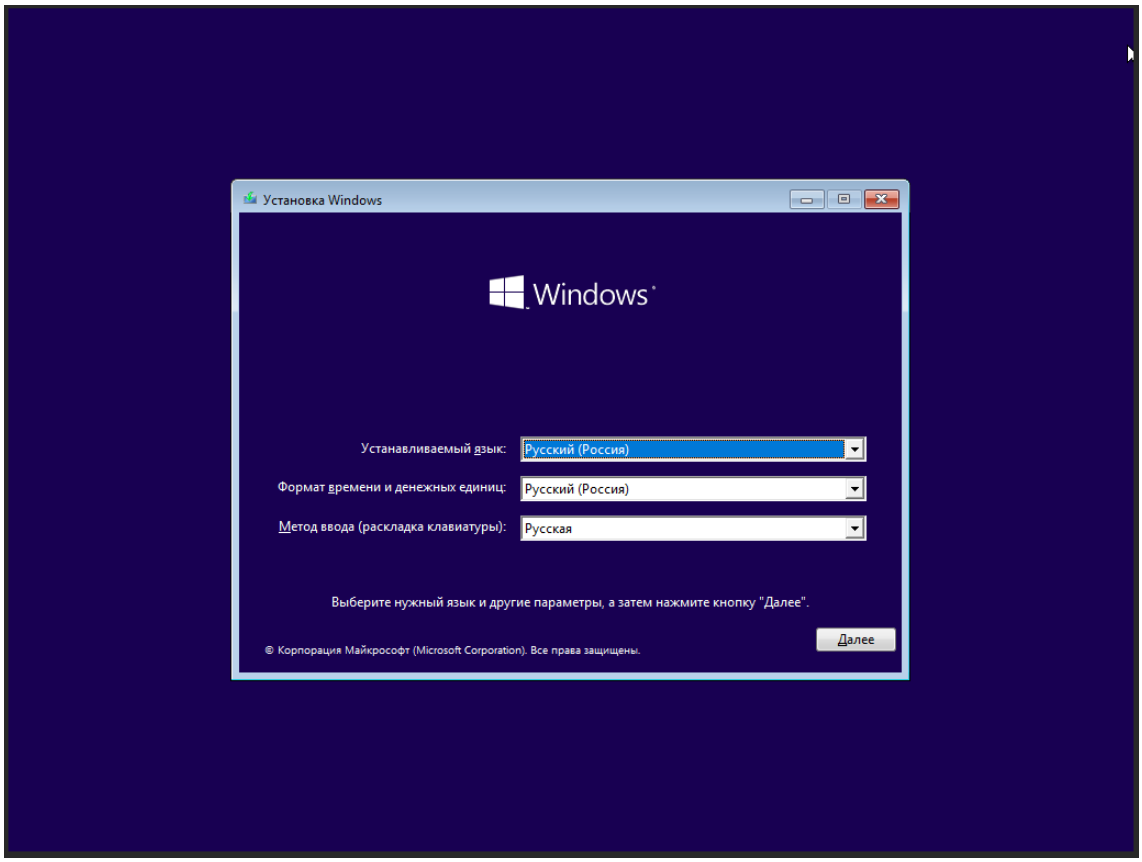
Указываемый размер диска на данном этапе будет являться минимально возможным для будущих ВМ. Если планируется использовать скрипты автоматизации (инит-скрипты), то рекомендуется указывать минимально необходимый размер для установки ОС и требуемого дополнительного программного обеспечения.

- нажать кнопку **Создать**.

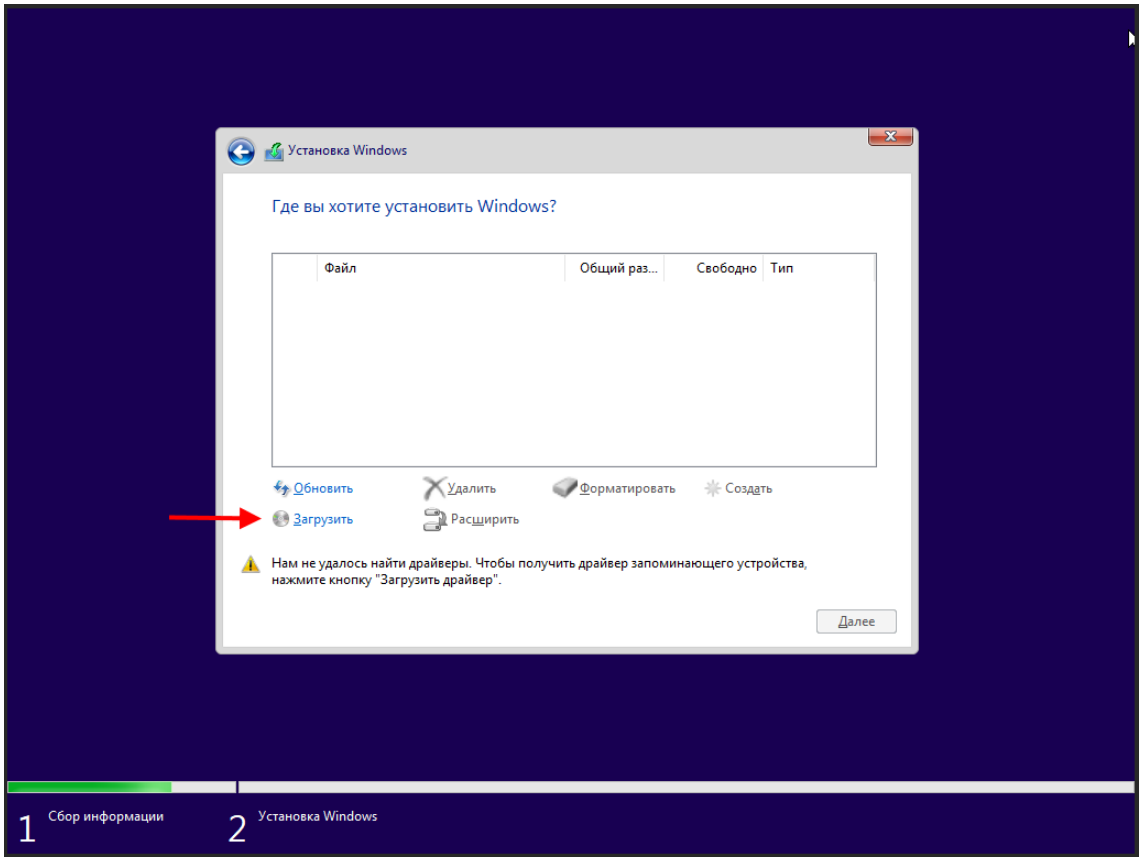
Созданный ВМ отобразится в разделе меню **Виртуальные Машины** со статусом **Собирается**. После завершения его подготовки статус изменится на **Запущен**, далее нужно открыть консоль ВМ для установки гостевой ОС.

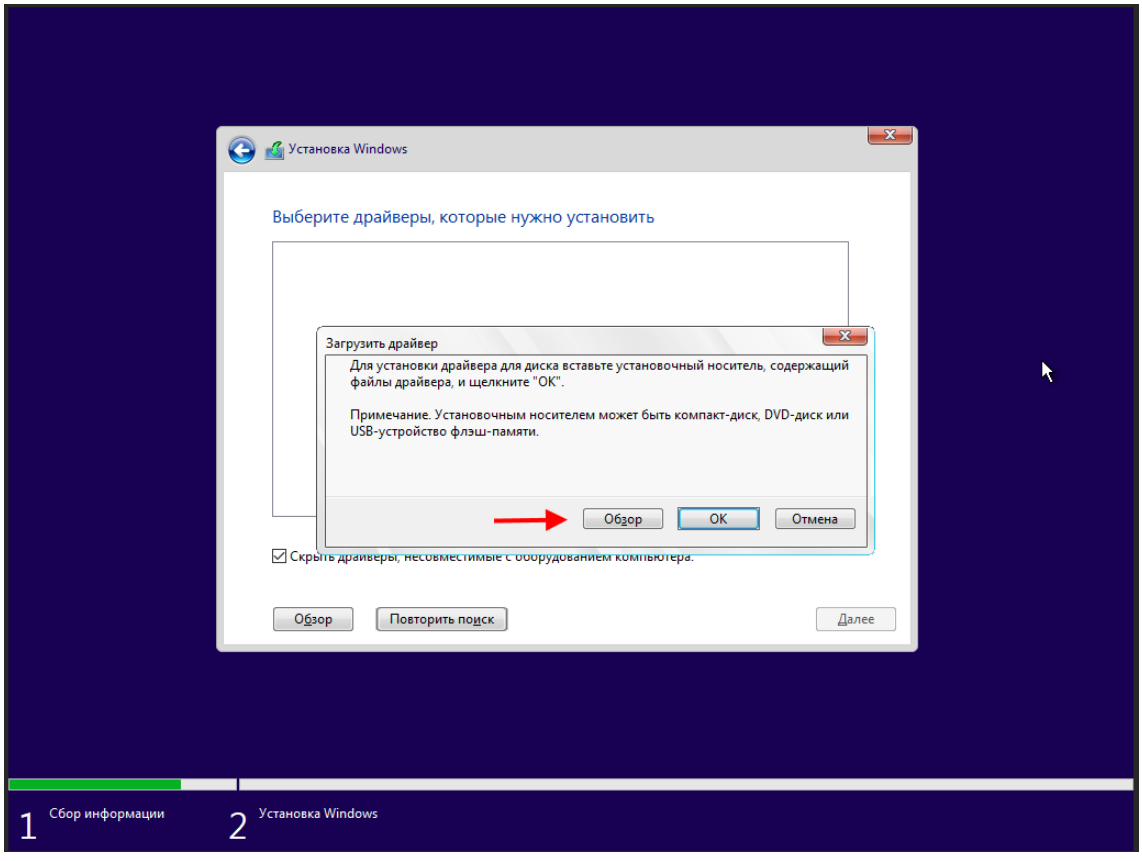
7.3.1.5. Установка Windows в консоли

После открытия консоли можно выполнить установку ОС Windows 10.

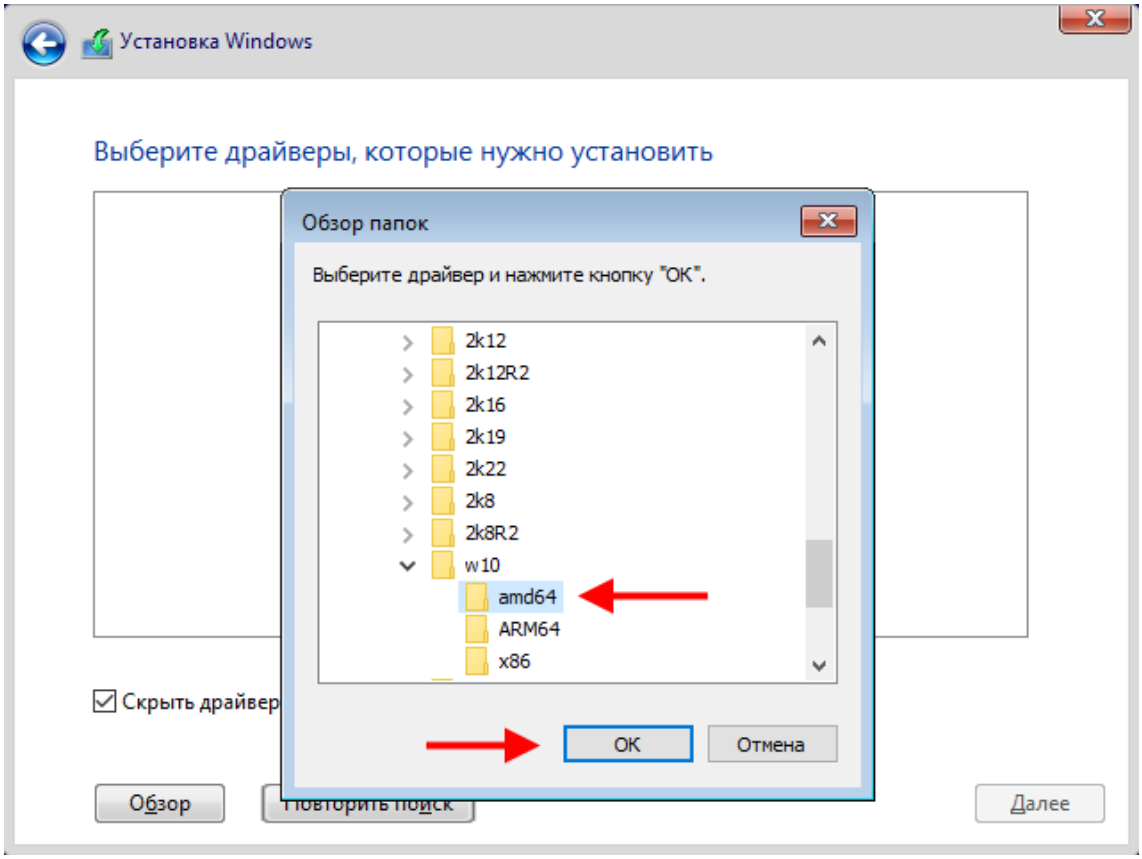


Дисковый контроллер не распознается по умолчанию, поэтому для него необходимо установить драйвер VirtIO.

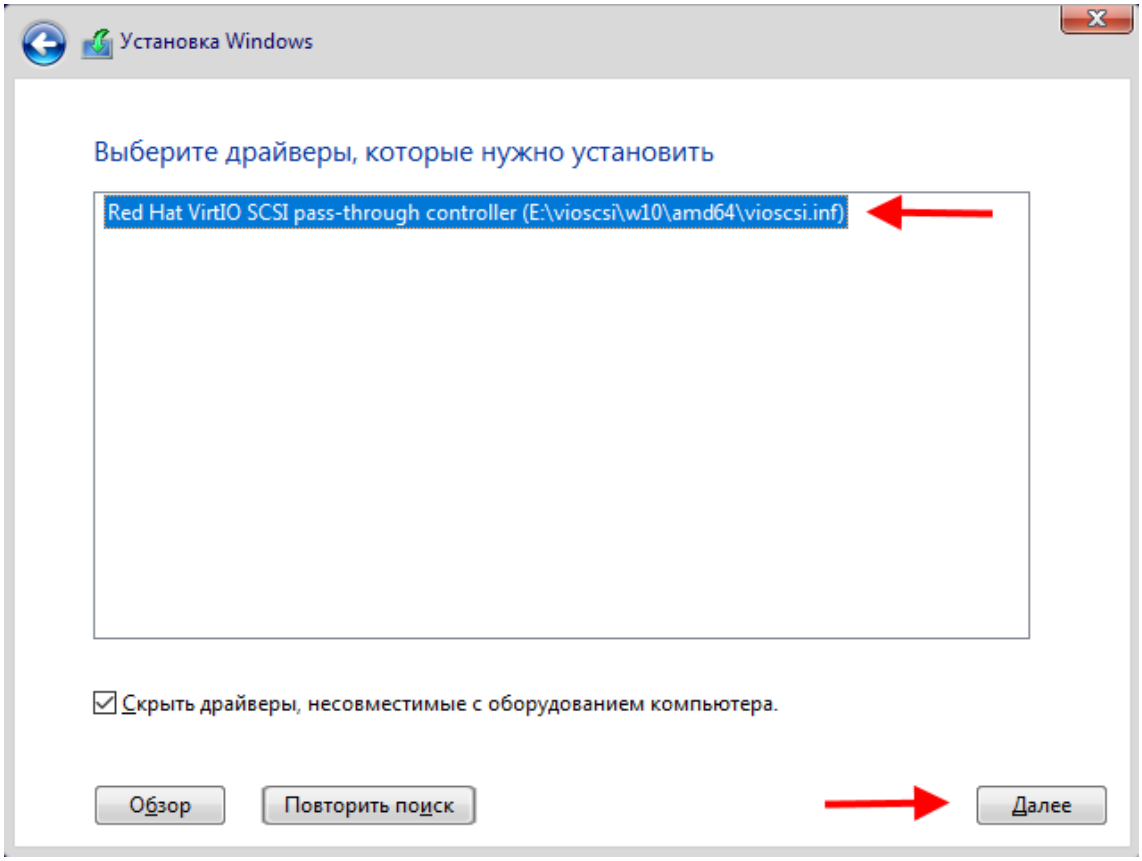




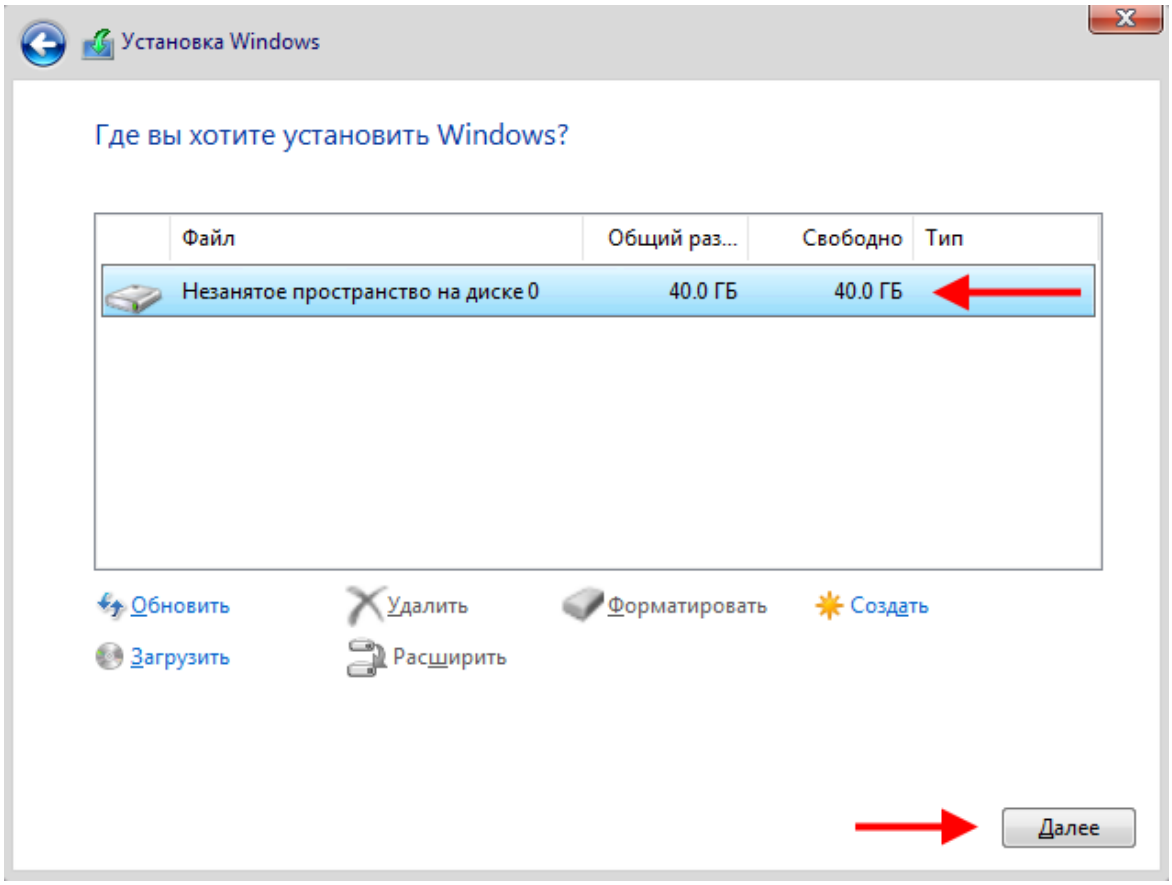
Здесь нужно выбрать CD-дисковод с драйверами virtio-win, далее директорию vioscsi → w10 → amd64 и нажать кнопку **OK**.



Драйвер отобразится в списке, после чего нужно нажать кнопку **Далее**.



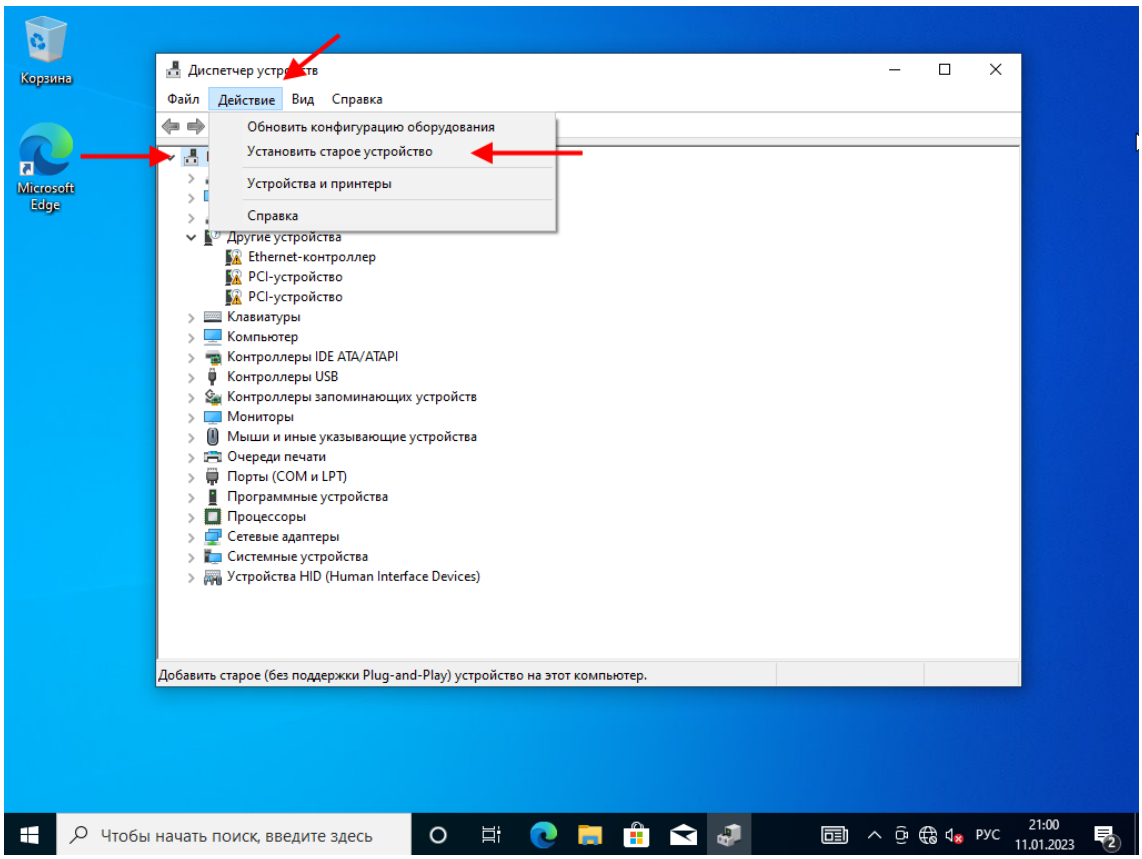
После установки драйвера будет доступен к выбору диск и можно продолжить установку гостевой ОС, нажав кнопку **Далее**.



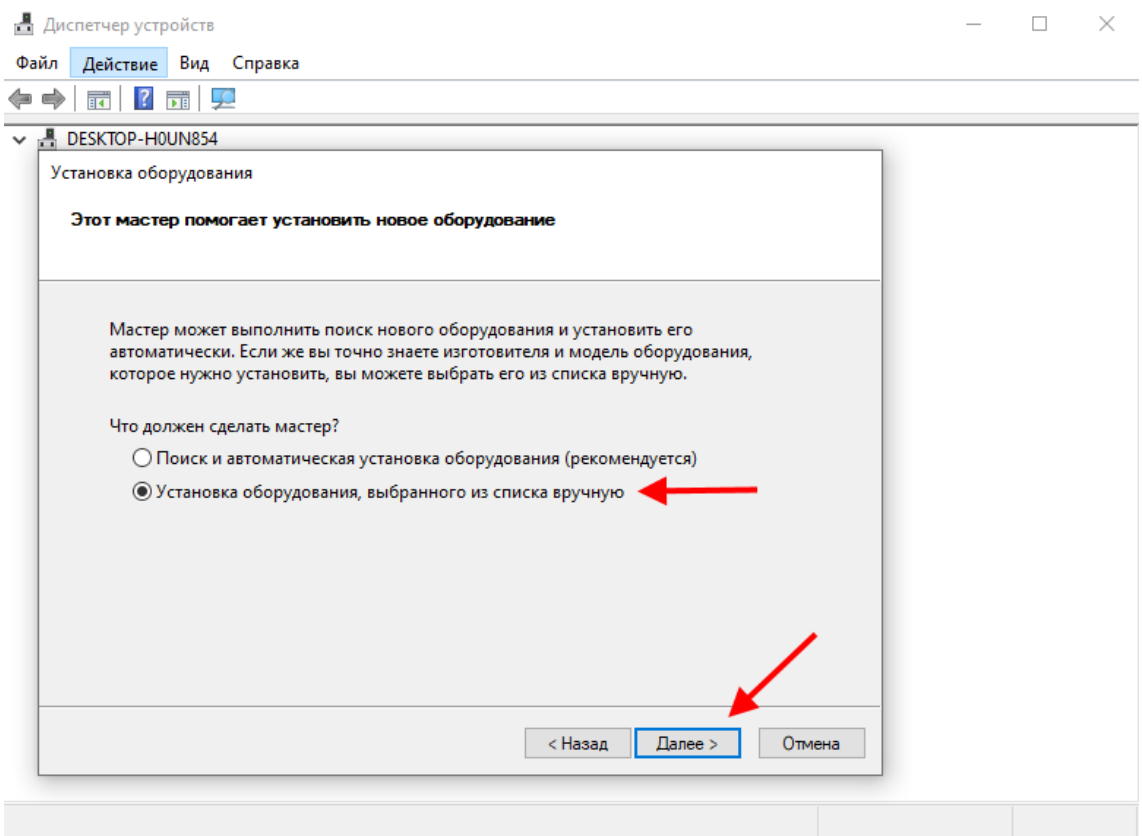
Далее выполняется установка гостевой ОС. По завершению необходимо установить дополнительные драйверы для сетевой карты и устройств PCI.

Драйверы сетевой карты находятся в директории NetKVM, а устройств PCI – в директории Balloon.

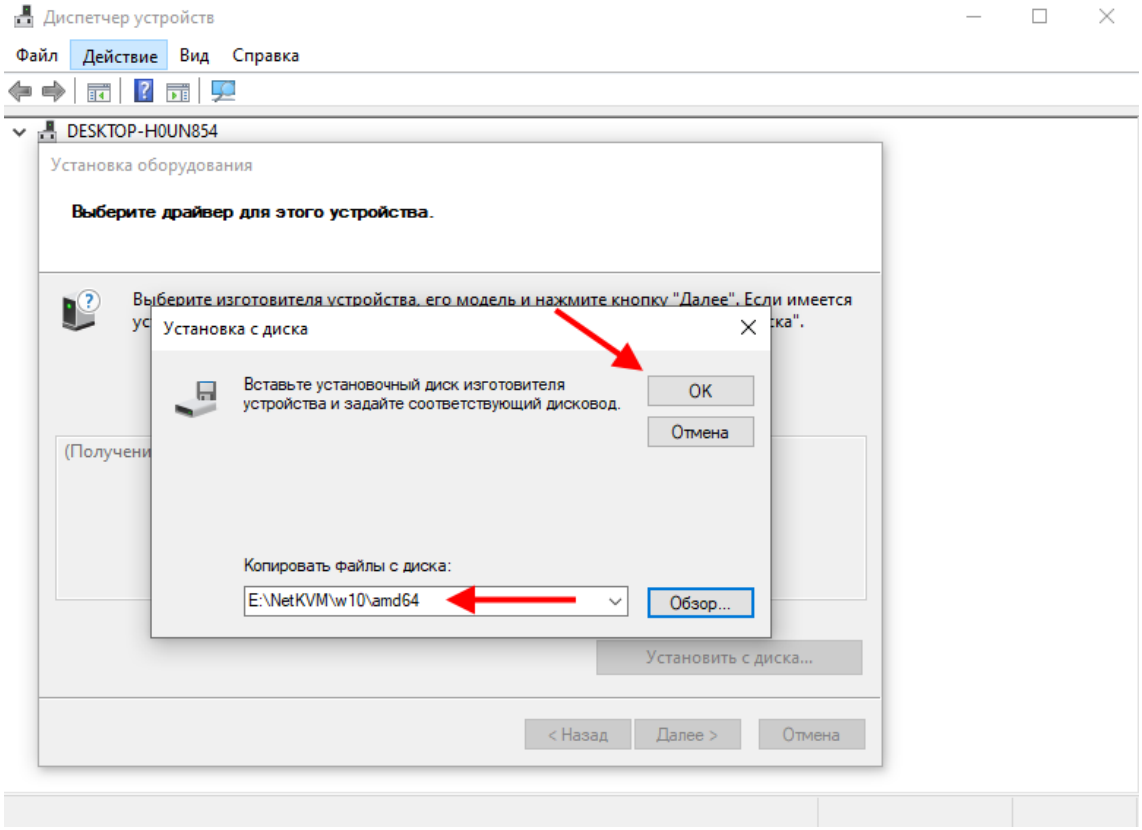
Для установки драйверов необходимо открыть **Диспетчер устройств**, выбрать ваш компьютер, нажать **Действие** → **Установить старое устройство**



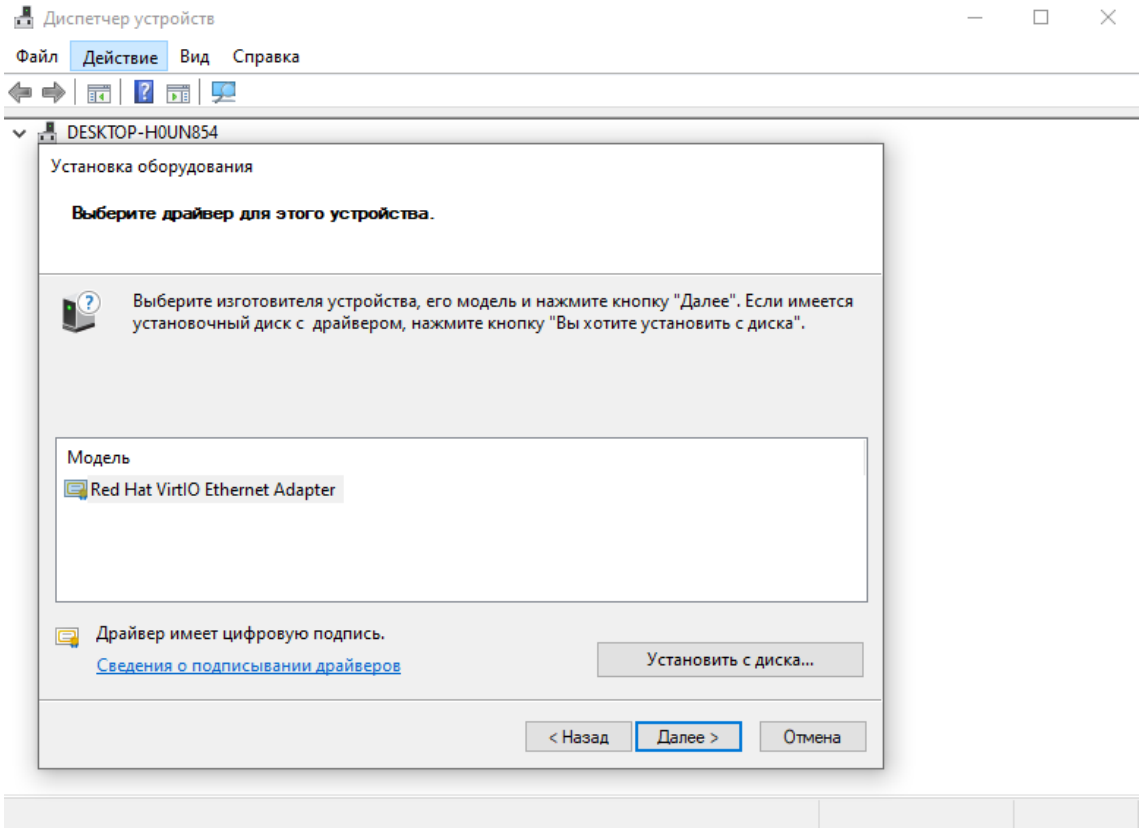
После запуска **Мастера установки оборудования** нажать **Далее** → **Установка оборудования, выбранного из списка вручную** → **Далее**.



После чего выбрать **Показать все устройства** → **Далее** → **Установить с диска** → **Обзор** и указать путь до драйвера сетевой карты (располагается в директории E:\NetKVM\w10\amd64\netkvm.inf на CD с драйверами virtio-win) и нажать кнопку **ОК**.



Драйвер отобразится в списке готовых к установке. Нужно нажать **Далее** → **Далее**.



После завершения установки необходимо нажать кнопку **Готово**.

После завершения установки драйвера будет выведено предупреждение: *Это устройство работает неправильно, т.к. Windows не удается загрузить для него нужные драйверы. (Код 31)*. Это не является ошибкой — драйвер инициализируется после перезагрузки системы.

Аналогично сетевому драйверу NetKVM выполнить установку драйвера для устройств PCI – Balloon (располагается в директории E:\Balloon\w10\amd64\balloon.inf на CD с драйверами virtio-win).

После этого можно выполнить необходимые настройки для будущего эталонного образа. Например, отключить брандмауэр Windows и управлять межсетевым экраном только через панель управления ПВ РУСТЭК, настроить удалённый доступ, а также обновить систему и установить необходимый софт. Если для установки обновлений требуется наличие сети и выход в интернет, то перезагрузите ВМ, чтобы сетевой драйвер инициализировался.

После выполнения всех необходимых настроек рекомендуется убрать привязку ВМ к оборудованию и выполнить очистку, запустив утилиту sysprep (находится в директории C:\Windows\System32\Sysprep\sysprep.exe) или использовать программное обеспечение Cloudbase-Init (https://www.cloudbase.it/downloads/CloudbaseInitSetup_x64.msi), а после этого выключить ВМ. Более подробно об этом и других видах кастомизации образов можно прочитать в разделе 7.4.

7.3.1.6. Удаление эталонного ВМ Windows

После выключения эталонного ВМ Windows install, его необходимо удалить. Смена статуса ВМ на **Выключен** отобразится в поле **Статус** панели управления в разделе **Виртуальные Машины**. Для удаления следует выбрать ВМ и нажать кнопку **Удалить** на панели инструментов, после этого ВМ перестанет отображаться в списке.

7.3.1.7. Создание образа на основе диска эталонного ВМ Windows

Для создания образа нужно:

- перейти в раздел **Диски**;
- выбрать Disk for Windows install размером 40 ГБ (он должен находиться в статусе **Доступен**);
- нажать кнопку **Загрузить как образ** на панели инструментов;
- в форме **Создание образа из диска** ввести имя, например, Windows 10 cloud;
- нажать кнопку **ПОДТВЕРДИТЬ**.

Статус диска изменится на **Загружается**. После завершения операции статус диска снова будет отображаться как **Доступен,** а созданный образ можно увидеть в разделе **Копии и образы – Образы**.

7.3.1.8. Редактирование созданного образа Windows

Созданный образ необходимо отредактировать. Для этого нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- выбрать образ Windows 10 cloud;
- нажать кнопку **Редактировать** на панели инструментов;
- заполнить поля формы **Редактирование образа** следующим образом:
  - **Имя** – ввести имя Windows 10 cloud;
  - **Имя ОС** – ввести имя windows10;
  - **RAM** – 0;
  - **Размер диска** – 40;
  - **Сетевой адаптер** – virtio;
  - **Дисковый контроллер** – virtio-scsi;

При редактировании образа необходимо указывать тот же дисковый контроллер, который использовался при создании образа ISO.

  - **Публичный** – установить флажок;
  - **Улучшения Windows** – установить флажок;
- нажать кнопку **Сохранить**.

7.3.1.9. Создание ВМ на основе созданного образа

После этого можно создать ВМ на основе готового образа. Для этого нужно:

- перейти в раздел **Виртуальные Машины**;
- нажать кнопку **Создать** на панели инструментов;
- заполнить поля формы **Создание ВМ** следующим образом:
  - **Имя** – ввести имя windows10-test;
  - **Проект** – выбрать проект admin;
  - **ОС** – выбрать Windows 10 cloud;
  - **Конфигурация** – выбрать конфигурацию (например, medium (2 CPU/4 Гб RAM));
  - **Размер диска** – 40;
  - **Тип диска** – выбрать тип (например, nfs);
  - Установить флажок **Удалять диск вместе с ВМ**;
  - **Сети** – выбрать сеть (например, int-net);
  - **Профили безопасности** – выбрать профили (например, default);
- нажать кнопку **Создать**.

После изменения статуса ВМ с **Собирается** на **Запущен**, можно проверить готовность ВМ к работе с помощью консоли.

7.3.2. Linux

В данном разделе описывается последовательность действий на примере создания образа ОС CentOS 9.

7.3.2.1. Создание образа для установки Linux

Для создания образа нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- нажать кнопку **Создать** на панели инструментов;
- заполнить поля формы **Создание образа** следующим образом:
  - **Имя** – ввести имя CentOS 9 ISO;
  - **Проект** – выбрать проект admin;
  - **Имя ОС** – ввести имя centos9_iso;
  - **Контейнер** – выбрать bare;
  - **Формат диска** – iso;
  - **RAM** – 0;
  - **Размер диска** – 0;
  - **Сетевой адаптер** – virtio;
  - **Дисковый контроллер** – virtio-scsi;

Для образов Linux рекомендуется выбрать **Дисковый контроллер** virtio-scsi.

  - **Публичный** – установить флажок;
  - **Улучшения Windows** – убрать флажок;
  - **Метод загрузки** – выбрать Файл;
- нажать кнопку **Создать**.

Созданный образ отобразится в разделе меню **Копии и образы – Образы**.

7.3.2.2. Загрузка образа ISO для установки Linux

Для загрузки образа ISO нужно:

- перейти в раздел меню **Копии и образы – Образы**;
- выбрать созданный образ из предыдущего раздела;
- нажать кнопку **Загрузить образ** на панели инструментов;
- в открывшейся форме **Загрузка образа** добавить файл образа ISO CentOS 9;
- после этого кнопка **ЗАГРУЗИТЬ** станет активна, нажать на неё.

7.3.2.3. Создание эталонного ВМ Linux

Для создания эталонного ВМ Linux нужно:

- перейти в раздел **Виртуальные Машины**;

- нажать кнопку **Создать** на панели инструментов;
- заполнить поля формы **Создание ВМ** следующим образом:
  - **Имя** – ввести имя CentOS install;
  - **Проект** – выбрать проект admin;
  - **ОС** – выбрать CentOS 9 ISO;
  - **Конфигурация** – выбрать конфигурацию (например, medium (2 CPU/4 Гб RAM));
  - **Размер диска** – 10;
  - **Тип диска** – выбрать тип (например, nfs);
  - **Сети** – выбрать сеть (например, int-net);
  - **Профили безопасности** – выбрать профили (например, default);

Нужно обязательно снять флажок в чекбоксе **Удалять диск вместе с ВМ**.

Указываемый размер диска на данном этапе будет являться минимально возможным для будущих ВМ. Если планируется использовать скрипты автоматизации (инит-скрипты), то рекомендуется указывать минимально необходимый размер для установки ОС и требуемого дополнительного программного обеспечения.

- нажать кнопку **Создать**.

Созданный ВМ отобразится в разделе меню **Виртуальные Машины** со статусом **Собирается**. После завершения его подготовки статус изменится на **Запущен**, далее нужно открыть консоль ВМ для установки гостевой ОС.

7.3.2.4. Установка Linux в консоли

После открытия консоли можно выполнить установку ОС CentOS 9.

После завершения установки ОС можно загрузиться с локального диска (при загрузке выбрать **Troubleshooting → Boot from local drive**) и выполнить дополнительные настройки или установить необходимое программное обеспечение для будущего эталонного образа, а после этого выключить ВМ. Также можно прочитать о кастомизации образов в разделе 7.4.

7.3.2.5. Удаление эталонного ВМ Linux

После выключения эталонного ВМ CentOS install, его необходимо удалить. Смена статуса ВМ на **Выключен** отобразится в поле **Статус** панели управления в разделе **Виртуальные Машины**. Для удаления следует выбрать ВМ и нажать кнопку **Удалить** на панели инструментов. После этого ВМ перестанет отображаться в списке.

7.3.2.6. Создание образа на основе диска эталонного ВМ Linux

Для создания образа нужно:

- перейти в раздел **Диски**;
- выбрать Disk for CentOS install размером 10 ГБ (он должен находиться в статусе **Доступен**);
- нажать кнопку **Загрузить как образ** на панели инструментов;
- в форме **Создание образа из диска** ввести имя, например, CentOS 9 cloud;
- нажать кнопку **ПОДТВЕРДИТЬ**.

Статус диска поменяется на **Загружается**. После завершения операции статус диска снова будет отображаться как **Доступен,** а созданный образ можно увидеть в разделе **Копии и образы – Образы**.

7.3.2.7. Редактирование созданного образа Linux

Созданный образ необходимо отредактировать. Для этого нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- выбрать образ CentOS 9 cloud;
- нажать кнопку **Редактировать** на панели инструментов;
- заполнить поля формы **Редактирование образа** следующим образом:
  - **Имя** – ввести имя CentOS 9 cloud;

- **Имя ОС** – ввести имя centos9;
- **RAM** – 0;
- **Размер диска** – 40;
- **Сетевой адаптер** – virtio;
- **Дисковый контроллер** – virtio-scsi;

При редактировании образа необходимо указывать тот же дисковый контроллер, который использовался при создании образа ISO.

- **Публичный** – установить флажок;
- **Улучшения Windows** – убрать флажок;

- нажать кнопку **Сохранить**.

7.3.2.8. Создание ВМ на основе созданного образа

После этого можно создать ВМ на основе готового образа. Для этого нужно:

- перейти в раздел **Виртуальные Машины**;
- нажать кнопку **Создать** на панели инструментов;
- заполнить поля формы **Создание ВМ** следующим образом:
  - **Имя** – ввести имя centos9-test;
  - **Проект** – выбрать проект admin;
  - **ОС** – выбрать CentOS 9 cloud;
  - **Конфигурация** – выбрать конфигурацию (например, medium (2 CPU/4 Гб RAM));
  - **Размер диска** – 10;
  - **Тип диска** – выбрать тип (например, nfs);
  - Установить флажок **Удалять диск вместе с ВМ**;
  - **Сети** – выбрать сеть (например, int-net);
  - **Профили безопасности** – выбрать профили (например, default);
- нажать кнопку **Создать**.

После изменения статуса ВМ с **Собирается** на **Запущен**. Проверить готовность ВМ к работе можно с помощью консоли.

7.4. Кастомизация

Кастомизация образов представляет собой использование дополнительного программного обеспечения (ПО) для расширения функционала и улучшения производительности ВМ.

7.4.1. Cloud-init

Cloud-init – это мощный инструмент кастомизации ВМ, создаваемых из образов, который является общепринятым стандартом. В большинстве дистрибутивов семейства операционных систем (ОС) Linux cloud-init находится в репозиториях. В облачных сборках Linux установлен по умолчанию. Cloudbase-init – это Windows-эквивалент проекта cloud-init.

После создания ВМ происходит обращение к ВМ метаданных или к специальному диску с метаданными и выполняется настройка ВМ. Возможно расширение диска и файловой системы, установка ПО, создание пользователей, настройка доступа и паролей и многое другое. Диск с метаданными используется, если в сети для ВМ отсутствует DHCP-сервер или есть необходимость создать уникальный инит-скрипт для ВМ.

Об использовании cloud-init читать здесь: https://cloudinit.readthedocs.io/en/latest/index.html

Об использовании cloudbase-init читать здесь: https://cloudbase-init.readthedocs.io/en/latest/index.html

7.4.1.1. Доступ к инит-скрипту

По умолчанию в ПВ РУСТЭК для доступа к инит-скрипту, привязанному к свойству образа **Имя ОС**, используется ВМ метаданных, доступный по адресу 169.254.169.254 для IPv4 или fe80::a9fe:a9fe для IPv6. Можно использовать один инит-скрипт для разных образов с одинаковым свойством **Имя ОС**

(подробнее читайте в разделе 7.1). Инит-скрипт для ВМ доступен по адресу http://169.254.169.254/latest/user-data. Данный способ работает только в том случае, если у сети ВМ есть DHCP-сервер. При отсутствии DHCP-сервера при создании ВМ в дополнительных настройках нужно установить флажок в чекбоксе **Создать диск с метаданными** и выбрать что будет использоваться в качестве источника: образ или файл.

При выборе образа в качестве источника будет создан диск с инит-скриптом, который привязан к образу через свойство **Имя ОС**.

При выборе файла в качестве источника также нужно выбрать файл с инит-скриптом, с которым будет создан диск.

7.4.1.2. Написание собственных инит-скриптов

Список всех доступных модулей cloud-init, которые можно использовать для написания собственных инит-скриптов, находится здесь: https://cloudinit.readthedocs.io/en/latest/reference/modules.html

Список всех доступных опций cloudbase-init, которые можно использовать для написания собственных инит-скриптов, находится здесь: https://cloudbase-init.readthedocs.io/en/latest/config.html

Инит-скрипт может быть как в формате YAML, так и в виде скриптов shell и PowerShell (доступен в cloudbase-init). Это основные, но не единственные форматы. Полный список для cloud-init находится здесь: https://cloudinit.readthedocs.io/en/latest/explanation/format.html, а для cloudbase-init – здесь: https://cloudbase-init.readthedocs.io/en/latest/userdata.html.

При использовании инит-скрипта в формате YAML первая строка должна начинаться с **#cloud-config**.

При использовании инит-скрипта в виде скрипта shell первая строка должна начинаться с **#!** (в случае использования cloud-init) или **#!/bin/bash** (в случае использования cloudbase-init).

При использовании инит-скрипта в виде скрипта PowerShell первая строка должна начинаться с **#ps1** (для Windows x64) или **#ps1_sysnative** (для Windows x64) или **#ps1_x86** (для Windows x32).

7.4.1.2.1 Макросы

В ПВ РУСТЭК есть несколько макросов, которые можно использовать при написании собственных инит-скриптов:

- [user] – содержит имя пользователя, создавшего ВМ;
- [host] – содержит имя создаваемого ВМ;
- [password] – содержит сгенерированный пароль в открытом виде;
- [crypto_pass] – содержит сгенерированный пароль в зашифрованном виде, совместимом с файлом паролей ОС Linux.

7.4.1.2.2 Автоматическая генерация пароля в ОС Linux

Для аутентификации в ОС с помощью логина и пароля для готовых образов Linux нужно:

- перейти в подраздел **Образы** раздела **Копии и образы**;
- выбрать образ;
- нажать на кнопку **Инит-скрипт**  на панели инструментов (предварительно задав свойство **Имя ОС** для образа);
- использовать следующий инит-скрипт, представленный в виде скрипта shell:

```
    #!/bin/sh
    /usr/sbin/useradd -d /home/[user] -s /bin/bash -G adm -p
'[crypto_pass]' [user]
    echo "%adm ALL=(ALL)ALL" > /etc/sudoers.d/firstboot
    chmod 440 /etc/sudoers.d/firstboot
    chown [user] /var/log/inithooks.log
```

Или в формате YAML:

```
#cloud-config
users:
  - default
  - name: [user]
    primary_group: adm
    groups: users
    lock_passwd: false
    home: /home/[user]
    shell: /bin/bash
    sudo: ALL=(ALL) NOPASSWD:ALL
    passwd: '[crypto_pass]'
```

- нажать на кнопку **Сохранить**.

Теперь при создании ВМ будет отображаться сообщение, содержащее пароль для ОС, который следует сохранить. Имя логина для аутентификации будет идентично имени пользователя, которым был создан ВМ.

7.4.1.2.3 Примеры некоторых операций

Содержимое инит-скрипта в формате YAML:

```
#cloud-config
disable_root: false
mounts:
  - [ sdc, /opt/data ]
packages:
  - pwgen
  - pastebinit
packages_update: true
runcmd:
  - systemctl restart cloud-init-local
  - [ ls, -l, / ]
ssh_pwauth: true
timezone: Europe/Moscow
```

disable_root разрешает пользователя root.
mounts подключает дополнительный диск.
packages устанавливает необходимое ПО.
packages_update обновляет установленное ПО.
runcmd выполняет произвольные команды**.**
ssh_pwauth разрешает доступ по SSH с использованием паролей.
timezone настраивает временную зону.

7.4.2. Sysprep (Windows)

До создания эталонного образа в эталонной ОС Windows рекомендуется использовать утилиту Sysprep, которая подготавливает ОС Windows для создания образов. Это замедлит создание нового ВМ на основе эталонного образа, но будет создана чистая ОС Windows. Для создания образа нужно:

- Выполнить команду %WINDIR%\system32\sysprep\sysprep.exe;
- В открывшемся окне выбрать "Переход в окно приветствия системы (OOBE)";
- Установить флажок в чекбоксе "Подготовка к использованию";
- В параметрах завершения работы указать "Завершение работы" и нажать **OK**.

Или можно использовать инструмент cloudbase-init из предыдущего раздела. На последнем шаге выполнения будет предложен запуск утилиты Sysprep. Нужно отметить флажком оба чекбокса и нажать **Finish**.

7.4.3. Windows OS Optimization Tool for VMware Horizon

Windows OS Optimization Tool for VMware Horizon – полезный инструмент для оптимизации образов с ОС Windows. Он оптимизирует эталонные образы путём автоматического удаления ненужных настроек и отключения ненужных функций, обычно присутствующих на физической машине Windows. В результате повышается производительность виртуального рабочего стола.

7.5. Утилиты

Существует несколько полезных утилит, облегчающих работу с образами.

- Libguestfs
- Diskimage-builder
- Windows Imaging Tools

Пакеты, включающие представленные утилиты, не входят в пакетную базу операционной системы ПВ РУСТЭК.

7.5.1. Libguestfs

libguestfs – это набор инструментов для доступа и изменения образов дисков. Его можно использовать для просмотра и редактирования файлов внутри дисков, конвертации P2V и V2V, форматирования дисков, изменения размера дисков и многого другого.

Об изменении образов с помощью данного инструмента читать здесь: https://docs.openstack.org/image-guide/modify-images.html

7.5.2. Diskimage-builder

Diskimage-builder – это инструмент автоматического создания образов дисков, поддерживающий различные дистрибутивы Linux (Fedora, Red Hat Enterprise Linux, Ubuntu, Debian, CentOS и openSUSE) и архитектуры.

Об использовании данного инструмента читать здесь: https://docs.openstack.org/diskimage-builder/latest/

7.5.3. Windows Imaging Tools

Windows Imaging Tools – это инструмент автоматического создания образов дисков, поддерживающий различные версии Windows и архитектуры.

Об использовании данного инструмента читать здесь: https://github.com/cloudbase/windows-imaging-tools

8. Высокая доступность виртуальных машин

Высокую доступность виртуальных машин в ПВ РУСТЭК реализует модуль `Virtual Appliance High Availability, VAHA`.

8.1. Режимы работы

VAHA работает на множестве хостов, на каждом — в одном из трёх режимов:

• **контроллер** — работает на узлах с ролью «Управление ВМ», следит за состоянием агентов и выполняет необходимые операции при обнаружении отказа;

• **агент** — работает на вычислительных узлах, пишет дисковые хартбиты (heartbeats) и отслеживает изоляцию узла, на котором установлен;

• **смешанный** — используется, если на узле включены обе роли и работает соответственно, выполняя функции как агента, так и контроллера.

8.2. Принцип работы

Общий принцип прост: VAHA следит за состоянием вычислительных узлов, каждую минуту получая хартбиты от агентов по одному или двум путям: только по сети или по сети и через общее дисковое хранилище. При обнаружении отказа узла, узел принудительно выключается, а все виртуальные машины с него эвакуируются и запускаются на других вычислительных узлах инсталляции.

8.2.1. Проверка по сети

Сетевой хартбит реализован пингом с контроллера по сети управления до каждого из узлов. Используется всегда, не отключается.

8.2.2. Проверка по общему хранилищу

Каждый агент раз в минуту пишет текущую временную метку в файл на общем хранилище, который используется контроллерами для определения последнего времени активности узла. По умолчанию дисковый хартбит выключен, включается в разделе РУСТЭК.Конфигуратора «Настройки высокой доступности виртуальных машин».

Рекомендуется включать только при использовании отдельной сети для СХД или при подключении общего хранилища по протоколу Fibre Channel. При использовании протоколов NFS или iSCSI через основную сеть управления дисковый хартбит не имеет смысла, поскольку использует ту же самую сеть и перестаёт работать вместе с ней.

8.3. Определение отказа или изоляции узла

Узел может быть в нескольких состояниях:

- OK — в последнюю проверку узел был доступен по всем включенным хартбитам;
- PARTIAL FAILED — в последнюю проверку узел был недоступен по одному из хартбитов, применимо только если настроен дисковый хартбит;
- FAILED — в последнюю проверку узел был недоступен по всем включенным хартбитам;
- EVACUATED — время недоступности хоста превысило таймаут, он был выключен и все ВМ были эвакуированы на другие вычислительные узлы.

Помимо состояния, на выполнение действий будет влиять опция «Эвакуировать частично недоступные узлы» в настройках высокой доступности в РУСТЭК.Конфигураторе. Подробнее в таблице:

проверка по сети	проверка по общему хранилищу	дисковый хартбит	состояние	эвакуировать частично недоступные узлы	действие
✅	✅	вкл	OK	любое	нет
✅	---	выкл	OK	вкл	нет
✅	❌	вкл	PARTIAL FAILED	вкл	выключение узла и эвакуация
❌	✅	вкл	PARTIAL FAILED	вкл	выключение узла и эвакуация
✅	❌	вкл	PARTIAL FAILED	выкл	нет
❌	✅	вкл	PARTIAL FAILED	выкл	нет
❌	❌	вкл	FAILED	любое	выключение узла и эвакуация
❌	---	выкл	FAILED	любое	выключение узла и эвакуация

8.3.1. Контроллер

Контроллеры определяют отказавший узел по последней метке времени его активности любым из доступных путей.

8.3.2. Агент

Агенты определяют собственную изоляцию по нескольким критериям:

1. Недоступность остальных узлов через сервис `consul`.
2. Отсутствие пинга до шлюза.
3. Недоступность общего хранилища, если включены дисковые хартбиты.

Таймаут изоляции до начала активных действий всегда на 30 секунд меньше, чем основной таймаут.

8.4. Действия при отказе

Отказавший хост будет выключен вне зависимости от настроек VAHA, чтобы не допустить повреждения данных виртуальных машин. Это может произойти двумя способами: либо его выключит контроллер через IPMI, либо агент вызовет состояние kernel panic на узле.

8.4.1. Контроллер

Если один из узлов признан отказавшим, VAHA эвакуирует виртуальные машины, опционально — выключает узел через IPMI.

8.4.2. Агент

Если узел, на котором выполняется агент, признан изолированным, VAHA выключает все ВМ, которые расположены на нём, и вызывает kernel panic, после чего контроллер эвакуирует виртуальные машины с отказавшего узла.

Проверка изоляции выполняется только в том случае, если не включена опция «Использовать IPMI для выключения изолированных узлов».

8.5. Настройка

VAHA настраивается при инсталляции ПВ РУСТЭК через РУСТЭК.Конфигуратор в разделе «Настройки высокой доступности виртуальных машин». Для гибкой настройки можно использовать конфигурационные файлы, однако, следует помнить, что:

• во-первых, изменения просуществуют до следующего запуска РУСТЭК.Конфигуратора;

• во-вторых, изменения нужно вносить на всех узлах.

В таблице ниже приведены основные опции, их соответствие настройкам в РУСТЭК.Конфигураторе и описание:

/etc/vaha/vaha.conf	РУСТЭК.Конфигуратор	Описание
`compute`	---	Используется ли агент, определяется состоянием роли «Вычислительный узел»
`control`	---	Используется ли контроллер, определяется состоянием роли «Управление ВМ»
`datastore_heartbeat`	Включить дисковый хартбит	
`datastore_heartbeat_path`	Путь для хартбит-директории	В эту директорию будет примонтирован общее хранилище для обмена хартбитами
`enabled_only`	Проверять только разрешённые узлы	Не проверять хартбиты и состояние, если узел запрещён после эвакуации, из-за режима обслуживания или вручную
`ping_count`	---	Количество пингов для проверки сетевого хартбита, по умолчанию 2
`ping_wait_sec`	---	Время ожидания для каждого пинга, по умолчанию 3
`connect_attempts`	---	Количество попыток подключиться по IPMI, по умолчанию 4
`connect_timeout_sec`	---	Таймаут соединения IPMI, по умолчанию 10
`recheck_on_error`	---	Проверять повторно при ошибке подключения IPMI, по умолчанию `False`
`use_ipmi`	Использовать IPMI для выключения узлов	
`threshold_minutes`	Порог срабатывания в минутах	Основной таймаут: через сколько минут выключать узел и эвакуировать ВМ
`attempts`	Количество попыток выключения узла	Сколько раз пытаться выключить и эвакуировать, если используется IPMI
`attempts_interval_sec`	Интервал между попытками в секундах	

`disable_on_evacuate`	Запрещать узел после эвакуации	
`partial_failed_evacuate`	Эвакуировать частично недоступные узлы	

9. Мультитенантность

9.1. Домены

Существуют следующие области видимости:

- **Платформа** — глобальный уровень;
- **Домен** — совокупность входящих в него проектов и пользователей;
- **Проект** — логическая группа ресурсов внутри домена.

Домен — это высший уровень абстракции для ресурсов, проектов и пользователей. Использование доменов позволяет разграничить доступ к проектам в рамках одного домена.

В качестве домена может использоваться локальный домен, либо внешняя служба каталогов.

Администратор домена может создавать проекты и пользователей в домене, назначать роли пользователям в домене.

При задании названий важно учитывать:

- **Доменное имя** — глобально уникальное во всех доменах;
- **Название проекта** — уникальное в домене-владельце;
- **Название группы** — уникальное в домене-владельце;
- **Имя пользователя** — уникальное в домене-владельце.

9.2. Проекты

Проект — это контейнер, который группирует или изолирует ресурсы или объекты идентификации. Проект является базовой единицей владения ресурсами: ресурсы принадлежат конкретному проекту, проект принадлежит определенному домену.

9.3. Роли

Роли в ПВ РУСТЭК связаны с областью видимости. Роль связывает пользователя с объектом доступа и определяет его привилегии.

Администратор ПВ РУСТЭК может назначать пользователям в домене или проекте следующие роли:

- **reader** — роль с доступом "только на чтение", обладает наименьшими привилегиями, применяется для просмотра используемых ресурсов в заданной области видимости.
- **member** — роль участника проекта, позволяет просматривать и создавать ресурсы в заданной области видимости.
- **admin** — роль администратора, позволяет просматривать, создавать и удалять ресурсы в заданной области видимости.
- **service** — системная роль, предназначена исключительно для сервисов и не может применяться для пользователей. Роль позволяет служебным пользователям сервисов проверять токены, аутентифицировать и авторизовывать пользовательские запросы, взаимодействовать между собой. Сервисные пользователи имеют данную роль в проекте service.

Изменение или добавление ролей созданному пользователю осуществляется в портале в разделе **Доступы** → **Проекты,** в блоке **Пользователи.**

9.3.1. Роли Octavia

Данные роли предназначены для использования балансировщиков нагрузки.

Контроль доступа на основе ролей в Octavia обеспечивает детальную политику управления доступом. Данные роли так же, как и основные, управляются сервисом Keystone. Пользователи должны иметь одну из следующих ролей, чтобы иметь доступ к балансировщикам нагрузки:

- **load-balancer_admin** — роль администратора всех балансировщиков нагрузки.
- **load-balancer_global_observer** — роль с доступом "только на чтение", позволяет просматривать все доступные балансировщики нагрузки.

- **load-balancer_member** — роль пользователя, позволяет просматривать и создавать балансировщики.
- **load-balancer_observer** — роль с доступом "только на чтение", позволяет просматривать указанные доступные балансировщики нагрузки.
- **load-balancer_quota_admin** — роль администратора управления квотами балансировщиков.

9.4. Пользователи

У каждого пользователя есть свои полномочия.

Иерархия пользователей:

- Администратор платформы;
- Администратор домена;
- Администратор проекта;
- Пользователь проекта;
- Пользователь с доступом на просмотр.

Администратор платформы осуществляет управление платформой и создаёт пользователей, может видеть все проекты и не связан с ними, имеет наивысшие полномочия.

Администратор домена осуществляет управление внутри проектов, находящимися в данном домене.

У пользователя могут быть заданы разные роли в разных проектах: в одном — admin, в другом — reader, в третьем — member в зависимости от его полномочий. До создания пользователя следует, при необходимости, создать проект для него, затем создать пользователя.

Пользователи по умолчанию создаются с ролью member.

9.5. Квоты

Квоты — это операционные лимиты, которые позволяют ограничивать ресурсы на проект, чтобы предотвратить исчерпание системных ресурсов. Например, можно устанавливать количество виртуальных машин в проекте.

В случае, если назначена квота на проект, то во время создания или изменения ресурса будет выполняться проверка путём подсчета текущего использования предоставленных ресурсов, после чего будет выполнена ещё одна проверка, чтобы убедиться, что первоначальная проверка действительна. Использование ресурсов подсчитывается с использованием API и БД.

Управление квотами в веб-портале выполняется в разделе меню **Квоты**.

Существуют несколько категорий квот. Внутри каждой категории содержатся свои параметры, на которые можно устанавливать ограничения.

Категории квот

Название квоты	Описание
compute	Квоты на вычислительные ресурсы
dns	Квоты на DNS
key-manager	Квоты на управление секретными данными
load-balancer	Квоты на балансировщики нагрузки
network	Квоты на сеть
volumev3	Квоты на ресурсы хранения

Минимальное значение, которое может быть установлено на ресурс: -1. Это будет означать, что квота на ресурс выключена и не используется.

10. Оптимизация нагрузки платформы

10.1. Основные термины

Аудит — запрос на оптимизацию платформы. Результатом аудита является план действий.

План действий — набор миграций, необходимых для перераспределения нагрузки.

С помощью аудитов можно контролировать нагрузку физических узлов по CPU. Применение плана действий опускает нагрузку каждого физического узла ниже указанного порога. Для перераспределения нагрузки осуществляется миграция ВМ с перегруженных узлов на свободные.

Ожидаемое поведение: ВМ мигрируют с перегруженного физического узла только в том случае, если после миграции нагрузка свободных узлов не станет превышать порог. В случае, если невозможно опустить нагрузку всех узлов до порогового значения, будет смигрирована только часть ВМ.

Чтобы опустить нагрузку физического узла ниже заданного порога, надо:

- создать аудит;
- запустить аудит, получить план действий;
- выполнить план, если это необходимо.

10.2. Создание аудита

10.2.1. Типы аудитов

Разовый: выполняется однократно при создании.

Постоянный: выполняется с заданным интервалом времени в секундах. Можно настроить время начала и завершения.

10.2.2. Область аудита

При создании аудита или шаблона аудита можно включить или исключить сущности:

- агрегаты;
- области видимости;
- физический узел;
- метаданные ВМ.

10.2.3. Параметры

- Порог — значение в процентах, до которого будет снижена нагрузка каждого узла, если это возможно. Рекомендуемые значения: 70-85.
- Погрешность — допустимые отклонения от заданного порога. После применения миграции свободные узлы будут загружены на значение ниже, чем порог + погрешность. Рекомендуемые значения: 1-3.

10.2.4. Создание без шаблона

Обязательное поле: тип аудита.

Если не написать имя, оно заполнится автоматически на основе выбранной стратегии и текущих даты и времени.

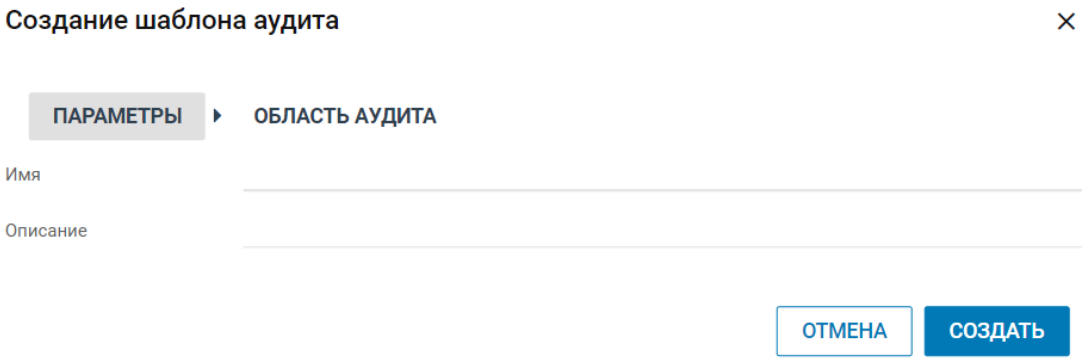
Чекбоксы:

- принудительно — аудит запустится, даже если применение предыдущего плана не завершилось;
- автозапуск — применять ли план действий автоматически.

10.2.5. Создание из шаблона

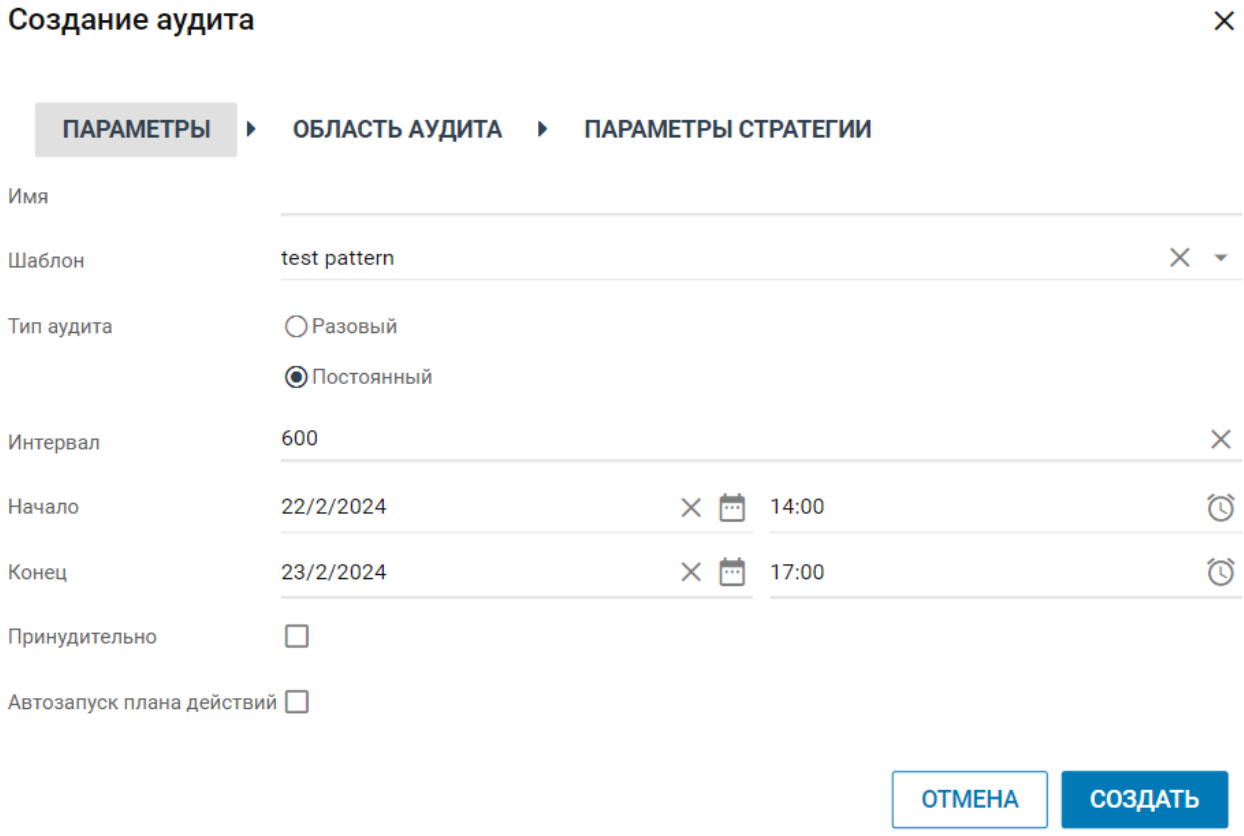
10.2.5.1. Создание шаблона

Шаблон аудита — способ сохранить области аудита для создания новых аудитов.



10.2.5.2. Пример создания аудита

Рассмотрим пример создания постоянного аудита с использованием шаблона.



Выбранные параметры означают, что:

- область аудита будет соответствовать указанной в шаблоне, перед созданием аудита её можно изменить;
- аудит будет запускаться каждые 10 минут, с 22.02.2024 14:00 до 23.02.2024 17:00;
- аудит не запустится повторно, пока не закончится выполнение предыдущего плана;
- план действий не будет применен автоматически, его можно будет применить в окне аудитов.

10.3. Нюансы работы

Аудит не обеспечивает равномерное распределение нагрузки по всем узлам, а только опускает нагрузку до указанного порогового значения.

При включенном оверкоммите если хотя бы один физический узел был загружен близко к 100%, после применения миграции может возникнуть ситуация, когда свободный узел стал перегруженным. Рекомендуется повторно запустить аудит после выполнения миграции, чтобы был сформирован план на основе "чистых" данных. По умолчанию данные обновляются раз в 5 минут.

11. Логи

11.1. Введение

Логи, они же «журналы событий» — важный элемент управления платформой. С их помощью можно диагностировать ошибки в работе системы, находить их причины и прослеживать цепочки событий. В разделе мы будем говорить о типах логирования, поверхностно рассмотрим основные настройки и разберёмся в чтении и анализе логов с помощью командной строки любого из узлов платформы ПВ РУСТЭК. Использовать в качестве примера будем компонент OpenStack Cinder и его сервисы.

11.2. Общая информация

В ПВ РУСТЭК практически каждый сервис использует два типа логирования событий: файлы журналов и запись логов в базу данных через сервис `syslog`. Рассмотрим их подробнее.

11.2.1. Файлы логов

Файлы используются для быстрой диагностики и мониторинга состояния. Каждый сервис использует свою именную директорию в системной директории `/var/logs`. В нашем примере cinder будет хранить свои журналы в директории `/var/log/cinder`. Обычно файл лога также называется по имени сервиса, но бывают случаи, когда компонент состоит из множества сервисов: тогда для каждого сервиса используется свой собственный файл, например `cinder-api.log`, с`inder-volume.log`, и так далее.

Важно помнить, что получить полную картину событий из логов на одном физическом ВМ может быть затруднительно и даже не всегда возможно, потому что в файлы на узле попадает информация только от того экземпляра сервиса, который запущен именно на нём.

11.2.2. Запись в базу через syslog

Хранение всех логов со всех узлов в единой базе данных позволяет получать полный срез информации и обычно используется для детального и комплексного анализа состояния платформы. Логи всех сервисов пишутся в таблицу `logs` базы данных `syslog` в СУБД PostgreSQL. Таблица имеет такую структуру:

Столбец	Значение
id	идентификатор записи
datetime	таймстамп события
host	имя узла, на котором произошло событие
program	название программы, которая отправила сообщение
data	содержимое сообщения в формате JSON
severity	уровень события

Для более удобного просмотра и анализа логов в базе был написан сервис `getlog`, о котором речь пойдёт ниже.

11.3. Настройка логирования

11.3.1. Oslo-log.conf

Все сервисы OpenStack и многие сервисы ПВ РУСТЭК используют в качестве системы логирования библиотеку `oslo_log`, которая основана на библиотеке logging, поставляемой вместе с Python. Это обеспечивает возможность использовать глобальные настройки логирования. В ПВ РУСТЭК они хранятся в файле `oslo_log.conf` в общей директории `/etc/openstack/`.

```
    [DEFAULT]
    log_dir = /var/log/$service_name
    debug = false
    logging_context_format_string = %(asctime)s %(process)d %(levelname)s %(name)s
[%(request_id)s %(global_request_id)s %(user_identity)s] %(instance)s%(message)s
    logging_default_format_string = %(asctime)s %(process)d %(levelname)s %(name)s
[-] %(instance)s%(message)s
    logging_exception_prefix = %(asctime)s %(process)d ERROR %(name)s %(instance)s
    logging_user_identity_format = %(user)s %(tenant)s
```

Наиболее важные параметры в этом файле:
log_dir — отвечает за путь к локальной директории с логами;
***_format_string** — шаблоны строк сообщения:

Поле	Значение
asctime	дата и время события до микросекунды
process	PID, идентификатор процесса в системе
levelname	уровень сообщения: DEBUG, INFO, WARNING, ERROR, CRITICAL
name	имя логгера
request_id	(контекст) идентификатор запроса внутри OpenStack
global_request_is	(контекст) глобальный идентификатор запроса, используется для объединения множества запросов в единую цепочку
user_identity	(контекст) информация о пользователе
instance	(опциональное поле) ID ВМ (виртуальной машины), к которому относится сообщение
message	непосредственно текст сообщения

Помимо этого, индивидуальные настройки логирования сервисов хранятся в файле logging.conf в служебных директориях: `/etc/<service_name>/logging.conf` на каждом узле.

11.3.2. Logging.conf

Настройки в этом файле — это стандартные опции логирования библиотеки logging. Подробно о них можно почитать в официальной документации: https://docs.python.org/3.7/library/logging.config.html#configuration-file-format. Здесь мы остановимся только на самых важных секциях и параметрах.

Основной настройкой, которая используется в траблшутинге, считается параметр level у логгеров и хендлеров. Именно он отвечает за уровень, начиная с которого сообщения будут попадать в логи. Уровней сообщения пять:

Уровень	Назначение
`DEBUG`	Детальная отладочная информация для диагностики проблем
`INFO`	Сообщения о том, что всё работает в штатном режиме
`WARNING`	Показывает, что происходит что-то неожиданное или что в будущем появятся проблемы, например «disk space low»
`ERROR`	Показывает, что из-за сбоя программа не может выполнять некоторые свои функции
`CRITICAL`	Сигнализирует о серьёзных ошибках, из-за которых программа не может продолжать работу

11.3.2.1. Loggers

Логгеры — инструменты, которые непосредственно выполняют отправку сообщений из программы. В названии секции указано их имя после «_», оно будет использовано в формате строки, определённом в `oslo_log.conf`. Разные имена логгеров нужны для разделения сообщений от разных компонентов программы внутри одного файла журнала. Например, все записи от очереди сообщений `rabbitmq`, которые генерируются в процессе работы, будут помечены именем `amqp`, а записи от основного сервиса — именем `cinder`.

```
###########
# Loggers #
###########

[logger_root]
level = INFO
handlers = syslog, file

[logger_cinder]
level = INFO
handlers =
qualname = cinder

[logger_amqp]
level = INFO
handlers =
qualname = amqp

[logger_sqlalchemy]
level = WARNING
handlers =
qualname = sqlalchemy
# "level = INFO" logs SQL queries.
# "level = DEBUG" logs SQL queries and results.
# "level = WARNING" logs neither.  (Recommended for production systems.)
```

Самыми важными настройками для логгеров являются:

level — уровень фиксируемых сообщений;

handlers — хендлеры, отслеживающие и обрабатывающие сообщения этого логгера. Пустое значение означает использование опции рутового логгера.

11.3.2.2. Handlers

Хендлеры — обработчики сообщений, которые отвечают за то, куда именно сообщение будет отправлено. Определено три типа хендлеров:

stderr — отправляет сообщения в стандартный консольный стрим `stderr` (обычно не используется);

syslog — отправляет сообщения в `syslog`, который, в свою очередь, записывает их в базу данных `syslog` в таблицу `logs`;

file — записывает сообщения в файл, указанный в аргументах хендлера.

```
############
# Handlers #
############

[handler_stderr]
class = StreamHandler
level = DEBUG
args = (sys.stderr,)
formatter = context

[handler_syslog]
class = oslo_log.handlers.OSSysLogHandler
level = INFO
args = ()
formatter = json

[handler_file]
class = handlers.WatchedFileHandler
level = INFO
args                                                                    =
('/var/log/cinder/{}.log'.format(os.getenv('SERVICE_NAME',os.path.basename(sys.argv
[0]))),)
formatter = context
```

Самыми важными настройками для хендлеров являются:

level — уровень фиксируемых сообщений: если уровень хендлера выше, чем уровень логгера, который его использует, все сообщения ниже этого уровня будут проигнорированы;

formatter — способ форматирования сообщений.

Как правило, используется два форматтера:

context — использует глобальные настройки форматирования строк, описанные в файле `oslo-log.conf`;

json — подготавливает сообщение в формате JSON для записи его в базу данных.

11.4. Getlog

`getlog` — это сервис для удобного просмотра консолидированных записей логов в базе данных. Это клиент-серверное приложение, его ВМ запущены на каждом узле с ролью `http` и отвечает только тот экземпляр, который расположен на узле с виртуальным IP-адресом. В директории `/var/logs/getlog` хранятся файлы журналов:

getlogserver.log — основной файл журнала сервиса;

getlogserver-uwsgi.log — журнал uwsgi-сервера;

access.log — журнал доступа веб-сервера nginx;

error.log — журнал ошибок веб-сервера nginx.

Как и в случае с другими распределёнными системами, в локальные файлы журналов попадает информация только от того экземпляра, который запущен на этом узле.

11.5. Клиенты getlog

У `getlog` есть два клиента: в панели управления платформой и в командной строке. Подробнее о том, как читать и фильтровать логи в панели управления см. **Руководство пользователя**, раздел **10. Логи**.

Консольный клиент реализован в качестве плагина к openstackclient и вызывается командой `openstack logs <subcommand> <options>`. Поговорим о доступных субкомандах.

11.5.1. Openstack logs list

Команда `openstack logs list` покажет доступные в базе сущности: узлы (hosts), столбцы таблицы логов (fields), программы (program) и уровни сообщений (severity):

```
+-------+----------+-----------------------------------------------+----------+
|  host |  fields  |                    program                    | severity |
+-------+----------+-----------------------------------------------+----------+
| hw-n2 |    id    |                                               |  alert   |
| hw-n3 | datetime |                    anacron                    |   crit   |
| hw-n4 |   data   |            ansible-authorized_key             |  emerg   |
| hw-n5 |   host   |             ansible-blockinfile               |   err    |
| hw-n6 | program  |               ansible-command                 |   info   |
|       | severity |                 ansible-copy                  |  notice  |
|       |          |                 ansible-cron                  | warning  |
|       |          |                 ansible-file                  |          |
|       |          |                 ansible-find                  |          |
|       |          |              ansible-gluster_peer             |          |
|       |          |             ansible-gluster_volume            |          |
|       |          |                ansible-hostname               |          |
|       |          |                ansible-ini_file               |          |
|       |          |               ansible-lineinfile              |          |
|       |          |                 ansible-mount                 |          |
|       |          |          ansible-openssl_certificate          |          |
|       |          |              ansible-openssl_csr              |          |
|       |          |           ansible-openssl_privatekey          |          |
|       |          |           ansible-openvswitch_bridge          |          |
|       |          |            ansible-openvswitch_port           |          |
|       |          |           ansible-os_keystone_domain          |          |
|       |          |        ansible-os_keystone_domain_info        |          |
|       |          |          ansible-os_keystone_endpoint         |          |
|       |          |            ansible-os_keystone_role           |          |
|       |          |          ansible-os_keystone_service          |          |
```

Подробная справка доступна по команде:

```
$ openstack logs list -h

usage: openstack logs list [-h] [-f {csv,json,table,value,yaml}] [-c COLUMN] [--quote
{all,minimal,none,nonnumeric}] [--noindent] [--max-width <integer>] [--fit-width] [--print-empty] [--sort-column
SORT_COLUMN]
                           [--sort-ascending | --sort-descending] [-s SELECT [SELECT ...]]

List database entities

optional arguments:
  -h, --help            show this help message and exit
```

```
      output formatters:
        output formatter options

        -f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                              the output format, defaults to table
        -c COLUMN, --column COLUMN
                              specify the column(s) to include, can be repeated to show multiple columns
        --sort-column SORT_COLUMN
                               specify the column(s) to sort the data (columns specified first have a priority,
non-existing columns are ignored), can be repeated
        --sort-ascending      sort the column(s) in ascending order
        --sort-descending     sort the column(s) in descending order

      CSV Formatter:
        --quote {all,minimal,none,nonnumeric}
                              when to include quotes, defaults to nonnumeric

      json formatter:
        --noindent            whether to disable indenting the JSON

      table formatter:
        --max-width <integer>
                               Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH
environment variable, but the parameter takes precedence.
        --fit-width           Fit the table to the display width. Implied if --max-width greater than 0. Set the
environment variable CLIFF_FIT_WIDTH=1 to always enable
        --print-empty         Print empty table if there is no data to show.

      This command is provided by the getlog plugin.
```

11.5.2. Openstack logs stat

Команда `openstack logs stat` покажет общую статистику по записям в базе: количество записей каждого уровня для каждой программы:

program	severity	total
	info	166
su	info	197
ntpd	info	12
pdns	err	15
pdns	warning	38
sshd	err	1
sshd	crit	1
sshd	info	547
sshd	notice	1
sudo	info	2772
sudo	notice	1386
vaha	crit	36
CROND	info	19844
crond	info	6
httpd	err	65
httpd	info	10191
nginx	err	85
nginx	info	109779
nginx	alert	69
nginx	emerg	28
redis	notice	19094
redis	warning	24
kernel	info	102
kernel	notice	103
kernel	warning	44

Посмотреть справку:

```
$ openstack logs stat -h

usage: openstack logs stat [-h] [-f {csv,json,table,value,yaml}] [-c COLUMN] [--quote
{all,minimal,none,nonnumeric}] [--noindent] [--max-width <integer>] [--fit-width] [--print-
empty] [--sort-column SORT_COLUMN]
                           [--sort-ascending | --sort-descending] [-s SELECT [SELECT ...]]

Show database statistics

optional arguments:
  -h, --help            show this help message and exit

output formatters:
  output formatter options

  -f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                       specify the column(s) to include, can be repeated to show multiple
columns
  --sort-column SORT_COLUMN
                        specify the column(s) to sort the data (columns specified first
have a priority, non-existing columns are ignored), can be repeated
  --sort-ascending      sort the column(s) in ascending order
  --sort-descending     sort the column(s) in descending order

CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric

json formatter:
  --noindent            whether to disable indenting the JSON

table formatter:
  --max-width <integer>
                          Maximum display width, <1 to disable. You can also use the
CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.
  --fit-width          Fit the table to the display width. Implied if --max-width greater
than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable
  --print-empty         Print empty table if there is no data to show.

This command is provided by the getlog plugin.
```

11.5.3. Openstack logs show: общая информация

Команда `openstack logs show` — это основная рабочая команда. Именно она покажет логи из базы по запросу пользователя. По умолчанию `getlog` фильтрует записи на основании `user_id` пользователя, поэтому для получения полной информации нужно определить системный контекст, выбрав нужный `os_cloud`. Системный клауд в ПВ РУСТЭК называется `rustack_system` и определяется тремя путями:

1. экспортом переменной окружения `OS_CLOUD` со значением `rustack_system`:

```
export OS_CLOUD=rustack_system
```

2. добавлением переменной окружения к команде:

```
OS_CLOUD=rustack_system openstack logs show <...>
```

3. использованием аргумента `--os-cloud`:

```
openstack logs show <...> --os-cloud rustack_system
```

Справка:

```
$ openstack logs show -h

usage: openstack logs show [-h] [-f {csv,json,table,value,yaml}] [-c COLUMN] [--quote
{all,minimal,none,nonnumeric}] [--noindent] [--max-width <integer>] [--fit-width] [--print-empty]
[--sort-column SORT_COLUMN]
                                [--sort-ascending | --sort-descending] [-d] [-n NUMBER] [-o
{id,datetime,host,program,severity} [{id,datetime,host,program,severity} ...]] [-r] [-t] [-s
SELECT [SELECT ...]]
                          [query ...]

Show logs by parameters

positional arguments:
  query         query to get log records from database

optional arguments:
  -h, --help            show this help message and exit
  -d, --data            full output for data field
  -n NUMBER, --number NUMBER
                        limitation output by line numbers
   -o {id,datetime,host,program,severity} [{id,datetime,host,program,severity} ...], --order
{id,datetime,host,program,severity} [{id,datetime,host,program,severity} ...]
                        output lines order by columns, default: datetime
  -r, --reverse         reversed output lines order by columns
  -t, --follow          tail -f functionality

output formatters:
  output formatter options

  -f {csv,json,table,value,yaml}, --format {csv,json,table,value,yaml}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated to show multiple columns
  --sort-column SORT_COLUMN
                          specify the column(s) to sort the data (columns specified first have a
priority, non-existing columns are ignored), can be repeated
  --sort-ascending      sort the column(s) in ascending order
  --sort-descending     sort the column(s) in descending order

CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric

json formatter:
  --noindent            whether to disable indenting the JSON

table formatter:
  --max-width <integer>
                    Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH
environment variable, but the parameter takes precedence.
  --fit-width           Fit the table to the display width. Implied if --max-width greater than 0.
Set the environment variable CLIFF_FIT_WIDTH=1 to always enable
  --print-empty         Print empty table if there is no data to show.

This command is provided by the getlog plugin.
```

11.5.4. Openstack logs show: основы

Когда нам нужно получить что-то из базы данных, мы используем язык `SQL`, запросы в котором строятся по такой упрощённой схеме:

`<ЧТО> <ОТКУДА> <КРИТЕРИИ ВЫБОРКИ> <ПОРЯДОК СОРТИРОВКИ> <КОЛИЧЕСТВО ЗАПИСЕЙ>.`

Например:

```
SELECT program, id FROM logs WHERE program = 'nginx' ORDER BY id LIMIT 3
```

В `getlog` реализован свой, более простой по синтаксису, чем `SQL`, язык — `SLQL`, Simple Log Query Language, а дополнительные условия, вроде сортировки и количества записей, настраиваются параметрами. Так, аналогичный запрос через `getlog` будет выглядеть так:

```
openstack logs show program = 'nginx' -o id -n3
```

Сравним:

SQL	openstack logs	значение	комментарий
`SELECT program, id`	`-c program, id`	ЧТО	столбцы program и id
`FROM logs`	-	ОТКУДА	из таблицы logs
`WHERE program = 'nginx'`	`program = 'nginx'`	КРИТЕРИИ ВЫБОРКИ	записи, у которых столбец program содержит значение 'nginx'
`ORDER BY id`	`-o id`	ПОРЯДОК СОРТИРОВКИ	сортировать по столбцу id, дефолтная сортировка — по убыванию
`LIMIT 3`	`-n3`	КОЛИЧЕСТВО ЗАПИСЕЙ	ограничить вывод тремя строками

Отдельно стоит обратить внимание на критерии выборки: именно эту часть представляет SLQL в getlog. Как правильно на нём писать, см. в разделе "Simple Log Query Language".

Не забывайте ограничивать выборку параметрами, иначе запрос будет выполняться долго, а вывод будет занимать очень много строк.

11.5.4.1. Столбцы и подробности

Отображаемые поля таблицы настраиваются параметром `-c, --columns`. По умолчанию отображаются поля с таймстампом, именем программы и сообщением об ошибке, но доступно для отображения несколько больше:

имя столбца	значение
`id`	идентификационный номер записи
`datetime`	таймстамп
`program`	имя программы
`host`	имя узла, с которого пришло сообщение
`severity`	уровень сообщения

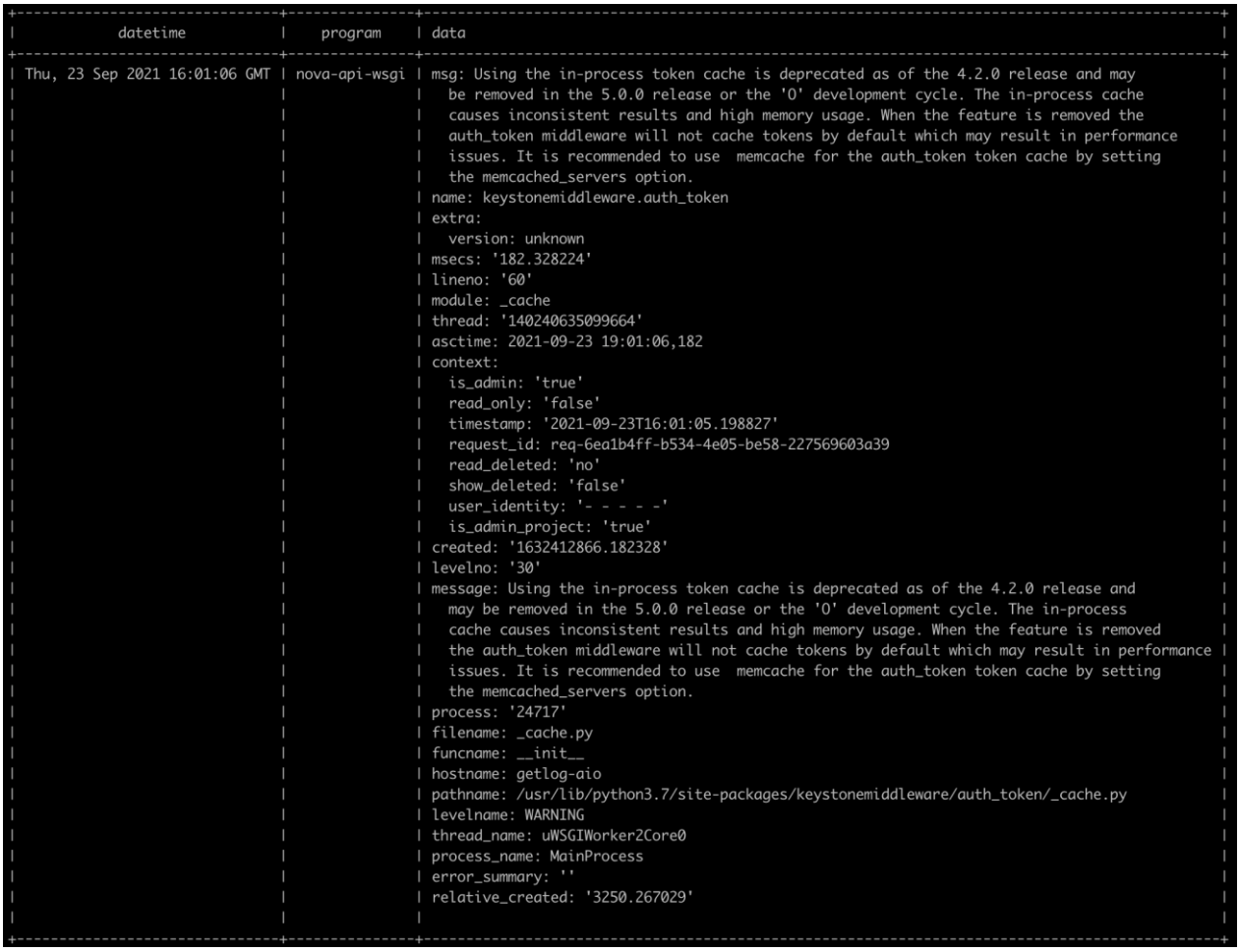
имя столбца	значение
`data`	непосредственное содержание записи

Всё это — поля оригинальной таблицы в базе данных, однако, это не единственное, что можно отобразить. Поле `data` содержит в себе данные в формате JSON, и по умолчанию из этого содержимого отображается только json-поле `message`. Просмотреть полный набор данных в столбце data можно, включив флаг `-d, --data`.

Помимо оригинальных столбцов из БД `getlog` позволяет отображать и содержимое конкретных json-полей из `data`. Например, по запросу:

```
openstack logs show program \~ ^nova* -d
```

В поле `data` мы увидим следующий JSON:

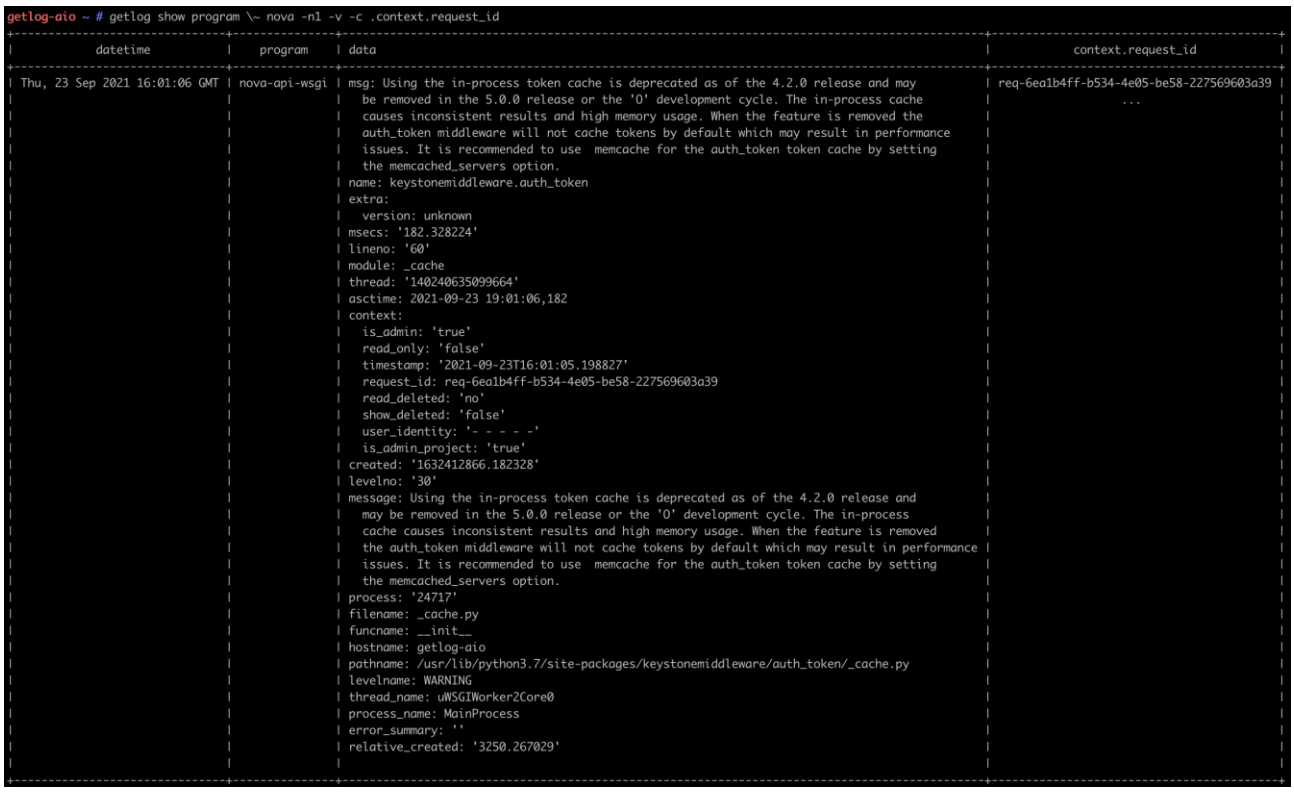


JSON — это древовидная структура, поэтому для доступа к вложенным объектам (строкам, словарям или спискам) нужно указать точный путь. Путь начинается от корня, который обозначается точкой, и точка же разделяет все последующие уровни. Например, отобразим в таблице конкретный request_id, к которому относится запись. Как мы видим на рисунке выше, он находится на втором уровне, внутри объекта context, так что итоговый путь будет выглядеть так: `.context.request_id`

Введём команду:

```
openstack logs show program \~ ^nova* -d -c .context.request_id
```

И получим следующий результат:



Таким же образом отображаются любые другие json-поля из столбца `data`.

11.5.4.2. Сортировка и порядок

Можно сортировать по любым полям, существующим в базе. Сортировка по json-полям невозможна. По умолчанию записи отсортированы по полю `datetime` от меньшего к большему, то есть, самые свежие записи появляются внизу таблицы. При необходимости можно развернуть порядок вывода, используя флаг `-r, --reverse`.

11.5.4.3. Ограничение вывода

11.5.4.3.1 Лимит

Записей логов в таблице достаточно много, поэтому имеет смысл показывать только последние несколько результатов. Для этого используется аргумент `-n, --number`.

> Важно помнить, что этот параметр работает точно так же, как `LIMIT` в `SQL`: он делает полную выборку и показывает только часть строк, ограниченную значением параметра. Поэтому, запросив все данные по определённому запросу, но ограничив вывод, мы имеем шансы получить не самые последние данные.

11.5.4.3.2 Безлимит

Другой распространённый случай — необходимость наблюдать за событиями в реальном времени. Всем известна команда `tail -f /path/to/logfile`, которая отображает записи в конкретном файле лога по мере его заполнения. `getlog` в таком режиме показывает все строки, соответствующие конкретному запросу. Для этого используется аргумент `-t, --follow`. При вызове он покажет последние десять строк, после чего будет добавлять новые записи в таблицу по мере появления в базе подходящих под критерии выборки сообщений.

11.6. Simple Log Query Language

`SLQL` — это язык запросов, позволяющий искать и фильтровать сообщения, хранящиеся в БД. В общем случае запрос состоит из условия сравнения, в котором участвуют поле, оператор сравнения и значение. Самый простой пример такого запроса:

```
program = nova-conductor
```

Этот запрос вернёт все записи, относящиеся к программе nova-conductor: сообщения всех уровней со всех узлов инсталляции, то есть, всё, что хранится в базе данных по этой программе.

11.6.1. Столбцы

Набор столбцов и принцип работы с ними уже знаком нам по разделу "Столбцы и подробности": тот же набор полей, существующих в базе, плюс возможность использовать json-поля из поля `data`. Это полезно, например, в случае, когда нужно отследить цепочку событий с одним и тем же `global_request_id`:

```
.context.global_request_id = req-bf57ca16-638a-4b91-ad09-e828146d13a8
```

11.6.2. Сравнения

Доступные для использования операторы сравнения:

значение	использование	строковый оператор
равно	== или =	eq
не равно	!= или <>	ne
больше	>	gt
меньше	<	lt
больше или равно	>=	ge
меньше или равно	<=	le

Также можно использовать операторы сравнения с регулярными выражениями, принятые в `PostgreSQL`:

значение	использование	строковый оператор
совпадение с регуляркой, с учётом регистра	~	re
совпадение с регуляркой, без учёта регистра	~*	ri
несовпадение с регуляркой, с учётом регистра	!~	nre
несовпадение с регуляркой, без учёта регистра	!~*	nri

И, разумеется, сами регулярные выражения. Например, запрос для поиска записей по всем сервисам компонента OpenStack Nova будет выглядеть так:

```
program ~* ^nova*
```

Здесь мы используем оператор сравнения ~*, чтобы найти все соответствия в именах программ, без учёта регистра.

Тот же самый запрос с использованием строкового оператора будет выглядеть так:

```
program ri ^nova*
```

11.6.3. Логические выражения

Условий сравнения может быть несколько. Тогда они объединяются между собой логическими операторами AND и OR, а подмножества условий можно группировать между собой, используя скобки. Например, запрос для поиска записей всех сервисов OpenStack Nova, кроме `nova-api-wsgi` и `nova-conductor`, будет выглядеть так:

```
program ~ 'nova' and (program != nova-api-wsgi and program != nova-conductor)
```

Для столбцов, в том числе, для json-полей, доступна проверка на существование и фильтрация по такому же критерию:

```
.context.user_id is not null
```

11.6.4. Нюансы использования в командной строке

Обратите внимание, что при работе из консоли лучше использовать строковые операторы сравнения, чтобы не экранировать спецсимволы в запросе обратным слэшом.

Из-за этого запрос вида:

```
program ~* nova
```

в стандартной оболочке должен выглядеть так:

```
program \~* nova
```

Или так:

```
program ri nova
```

12. Интеграция с внешними сервисами

12.1. Интеграция с сервисом каталога

12.1.1. Подготовка

12.1.1.1. Microsoft Active Directory

12.1.1.1.1 Пререквизиты

Нужно настроить Microsoft Active Directory (MS AD). В каталоге нужно создать:

1. Сервисного пользователя с правами на чтение в домене, который:
    a. может находиться в любом контейнере и быть членом любых групп;
2. Контейнер с пользователями, который:
    a. может быть простым `Container` или `Organizational Unit` и может находиться где угодно;
3. Группу безопасности:
    a. произвольная и должна находиться в выбранном контейнере из п.2;
4. Всех остальных пользователей, которым вы хотите дать доступ в ПВ РУСТЭК. При этом:
    a. пользователи должны находиться в контейнере из п. 2 и быть членами группы безопасности из п.3;
    b. если тип контейнера - `Organizational Unit (OU)`, то внутри него можно создавать другие OU (организационные единицы) для фрагментации пользователей, которых можно в них помещать.

Для настройки ПВ РУСТЭК вам понядобятся:

- сервисный пользователь:
    - `distinguishedName: cn=ServiceUser,cn=ContainerName,dc=yourdomain,dc=ru`;
    - пароль;
- контейнер с пользователями:
    - `distinguishedName: ou=RUSTACK_OU,dc=yourdomain,dc=ru`;
- группа безопасности:
    - `distinguishedName: cn=RUSTACK_USERS,ou=RUSTACK_OU,dc=yourdomain,dc=ru`;
- hostname или IP-адрес сервера каталога.

12.1.1.1.2 Проверка

Выполните команду в интерфейсе командной строки ПВ РУСТЭК:

```
ldapsearch -b <distinguishedName контейнера с пользователями> \
-H ldap://<hostname или IP сервера каталога> \
-D <distinguishedName сервисного пользователя> \
-w '<пароль сервисного пользователя>'
'(&(memberOf=<distinguishedName группы безопасности>)(objectClass=person)' cn
```

В выводе команды вы должны увидеть список пользователей. Если его там нет:

- проверьте `distinguishedName` сервисного пользователя и его пароль;
- проверьте hostname или IP-адрес сервера;
- проверьте имена контейнера и группы безопасности;
- убедитесь, что группа безопасности находится в контейнере;
- проверьте членство пользователей в группе безопасности.

12.1.1.2. FreeIPA

12.1.1.2.1 Пререквизиты

Настройка интеграции с сервисом каталога FreeIPA несколько проще, в связи с фиксированной структурой каталога.
Вам понадобятся:

- `user login` сервисного пользователя и его пароль;
- имя группы, в которую входят пользователи, которым вы хотите дать доступ в ПВ РУСТЭК.

12.1.1.2.2 Проверка

Выполните команду в интерфейсе командной строки ПВ РУСТЭК:

```
ldapsearch -b cn=users,cn=accounts,dc=yourdomain,dc=ru \
-H ldap://<hostname или IP сервера каталога> \
-D uid=<User login сервисного
пользователя>,cn=users,cn=accounts,dc=yourdomain,dc=ru  \
-w '<пароль сервисного пользователя>' \
'(memberOf=cn=<Имя группы>,cn=groups,cn=accounts,dc=yourdomain,dc=ru)' uid
```

В выводе команды вы должны увидеть список пользователей. Если его там нет, то:

- проверьте имя сервисного пользователя и его пароль;
- проверьте имя группы;
- проверьте адрес сервера.

12.1.2. Конфигурация ПВ РУСТЭК

В разделе «Интеграция с сервисом каталога»:

- отметьте чекбокс «Настроить интеграцию с сервисом каталога»;
- выберите нужный «Тип сервиса каталога»;
- заполните необходимую информацию:

Поле	Тип сервиса каталога	Значение	Комментарий
Полное имя домена (FQDN)	*	`yourdomain.ru`	
Короткое имя домена	*	`yourdomain`	
Hostname или IP-адрес сервера каталога	*	`dc.yourdomain.ru` или `192.168.0.100`	Если вы указываете hostname — укажите «*DNS платформы*» в разделе «*Сеть*», чтобы разрешать его имя.
Полное CN сервисного пользователя	`MS AD`	`cn=ServiceUser,cn=ContainerName,dc=yourdomain,dc=ru`	
	`FreeIPA`	`uid=ServiceUser,cn=users,cn=accounts,dc=yourdomain,dc=ru`	
Пароль сервисного пользователя	*	пароль	Нет ограничений по сложности пароля.
Фильтр пользователей	`MS AD`	`(memberOf=cn=RUSTACK_USERS,ou=RUSTACK_OU,dc=yourdomain,dc=ru)`	
	`FreeIPA`	`(memberOf=cn=RUSTACK USERS,cn=groups,cn=accounts,dc=yourdomain,dc=ru)`	
CN контейнера с пользователями	`MS AD`	`ou=RUSTACK_OU,dc=yourdomain,dc=ru`	
	`FreeIPA`	`cn=users,cn=accounts,dc=yourdomain,dc=ru`	
Размер пагинации			+100 за каждую полную 1000 пользователей в группе: 1 - 1000 = 0

Поле	Тип сервиса каталога	Значение	Комментарий
	`MS AD`	`0-1000`	1001-2000 = 100 ... >10000 = 1000.
	`FreeIPA`	`0`	Сервис каталога FreeIPA не поддерживает пагинацию, поэтому ее необходимо отключить, указав значение пагинации нулевым.

***** продолжите конфигурацию, или выберите «Применить конфигурацию РУСТЭК», если уже развернули платформу.

12.1.3. Управление пользователями сервиса каталога в ПВ РУСТЭК

12.1.3.1. Подготовка

Чтобы пользователи из сервиса каталога могли авторизоваться в портале — нужно заранее назначить им какую-либо роль в существующем или новом проекте ПВ РУСТЭК.
Для этого:

- зайдите в портал ПВ РУСТЭК под учетной записью Администратора Плафтормы;
- при необходимости — создайте отдельный проект в разделе «Доступы» → «Проекты»;
- перейдите в раздел «Доступы» → «Пользователи» и назначьте каждому из пользователей соответствующую роль (admin, member, reader) в выбранном проекте.

Также все эти операции можно выполнить в интерфейсе командной строки ПВ РУСТЭК:

Создание проекта:

```
openstack project create --os-cloud rustack_system <имя проекта>
```

Добавление всем пользователям сервиса каталога роли member в созданном проекте:

```
DOMAIN_USERS=$(openstack user list --os-cloud rustack_system --domain '<полное имя домена>' -f value -c ID)
for user in $DOMAIN_USERS;do openstack role add --user $user --project <имя проекта> member; done
```

12.1.3.2. Авторизация

При входе в портал имя пользователя нужно указывать в формате: `user@yourdomain.ru.`
При блокировке пользователя в сервисе каталога он так же блокируется в ПВ РУСТЭК.
При удалении пользователя из группы в сервисе каталога он не сможет авторизоваться в ПВ РУСТЭК.