

Магазин приложений Marketplace

Инструкция по
добавлению сервисов

Версия 1.0 от 27.10.23

Оглавление

1. Общая информация	3
1.1. Типы сервисов	3
1.2. Типы тарифных опций	5
1.3. Стоимость тарифных планов и опций	5
1.4. Матрица тарифных планов	7
1.5. Мастер конфигурации тарифного плана	8
2. Добавление image-based сервиса в Магазин	9
2.1. Порядок действий	9
2.2. Создание образа сервиса	9
2.3. Структура сервисного пакета	15
2.4. Конфигурация image-based сервиса	17
2.5. Конфигурация инфраструктуры image-based сервиса	58
2.6. Загрузка image-based сервиса в Магазин	70
3. Обновление image-based сервиса	89
4. Перенос image-based сервиса в архив	91
5. Добавление SaaS-сервиса в Магазин	93
5.1. Порядок действий	93
5.2. Брокер для SaaS-сервиса	93
5.3. Конфигурация SaaS-сервиса	96
5.4. Загрузка конфигурации сервиса в брокер	115
5.5. Загрузка SaaS-сервиса в Магазин	116
6. Обновление SaaS-сервиса	122
A. Примеры YAML-файлов для image-based сервиса	124
A.1. Пример файла service.yaml	124
A.2. Пример файла plan.yaml	124
B. Провайдер iVK CS	125
B.1. Ресурсы	125
B.2. Источники данных	134
C. Пример манифеста для сервиса Kafka	136
D. Пример JSON-файла для SaaS-сервиса	144
E. Термины и сокращения	149

1. Общая информация

Полное наименование системы: Магазин приложений Marketplace (XaaS).

Краткое наименование системы: Магазин.

Магазин интегрирован с облачной платформой VK Cloud. Магазин обеспечивает доступ пользователей облачной платформы к сервисам, размещённым в нём.

Роли пользователей, взаимодействующих с Магазином:

- Поставщик — добавляет сервис в Магазин. Доступны тестовые и открытые пространства имён Магазина.
- Потребитель — использует сервис, размещённый в Магазине. Доступны только открытые пространства имён Магазина.

Настоящий документ предназначен для поставщиков.

Настоящий документ описывает порядок добавления сервисов в Магазин.

1.1. Типы сервисов

Магазин позволяет размещать следующие типы сервисов:

- SaaS — централизованно установленный мультитенантный продукт. Инфраструктура для работы сервиса предоставляется поставщиком. Поставщик разворачивает сервис либо на сторонней инфраструктуре, либо на инфраструктуре облачной платформы в своём проекте. Пользователю предоставляется доступ к экземпляру сервиса через отдельный тенант/аккаунт.
- Image-based — продукт, который разворачивается на основе образов виртуальных машин в облачном проекте пользователя. Для работы сервиса может использоваться дополнительная инфраструктура облачной платформы: виртуальные сети, балансировщики нагрузки, кластеры DBaaS, S3, резервное копирование. Инфраструктура для работы сервиса предоставляется облачной платформой. Экземпляры сервиса разворачиваются в проекте пользователя.

Услуги, предоставляемые сервисом, описываются с помощью тарифных планов и опций. Тарифная опция — это конкретный параметр плана.

Взаимодействие Магазина с сервисами осуществляется по протоколу VK OSB с помощью брокеров (рисунок 1). Брокер обеспечивает доставку конфигурации сервиса в Магазин: Магазин периодически опрашивает брокера о текущей конфигурации сервиса.

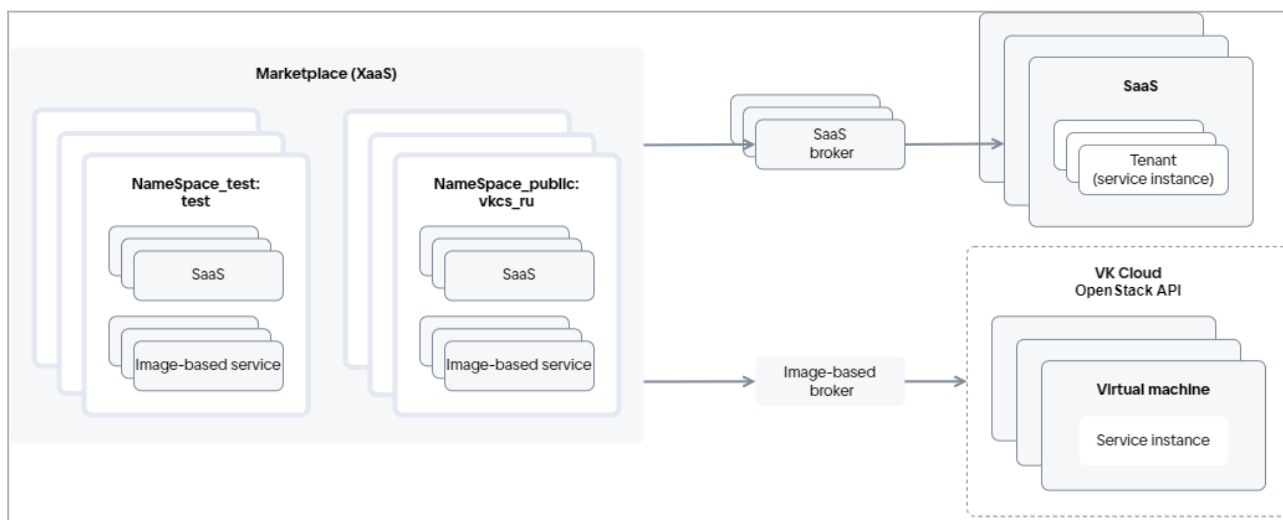


Рисунок 1 — Взаимодействие Магазины с сервисами

SaaS-брокер обеспечивает взаимодействие конкретного SaaS-сервиса с Магазином. Image-based брокер обеспечивает взаимодействие image-based сервисов с Магазином. Image-based брокер внутри себя имеет tenants, объединяющие image-based сервисы одного поставщика.



Поставщик SaaS-сервиса разрабатывает SaaS-брокер.

Поставщик image-based сервиса разрабатывает сервисный пакет для image-based брокера.

Брокер для image-based сервисов разработан VK.

Магазин включает в себя тестовые (`NameSpace_test`) и открытые (`NameSpace_public`) пространства имён.

Тестовые пространства имён предназначены для проверки загруженной конфигурации сервиса перед её публикацией в Магазины. В тестовых пространствах имён доступны все ревизии сервиса, описанные в его конфигурации. Ревизии доступны только пользователям, у которых есть доступ в тестовые пространства имён. Тестовые пространства имён недоступны потребителям.

Открытые пространства имён обеспечивают доступ потребителей к сервисам, размещённым в Магазины.



Потребителям доступна только последняя опубликованная ревизия сервиса.

Сущности, связанные с сервисом и используемые при интеграции с Магазином, приведены в таблице 1.

Таблица 1 — Сущности, связанные с сервисом

Имя	Описание
Инстанс сервиса (Service instance)	Экземпляр сервиса, развёрнутый у пользователя. Для SaaS-сервиса инстанс сервиса — это тенант/аккаунт SaaS-сервиса
Сервисная привязка (Service binding)	Сущность, которая создаётся после создания инстанса сервиса на основании запроса от облачной платформы к брокеру и связывается с инстансом сервиса. Например, сервисную привязку можно использовать для создания ссылки с учётными данными для доступа к инстансу сервиса (ссылка с учётными данными — это сервисная привязка) или чтобы отправить сервису логи, сгенерированные облачной платформой

1.2. Типы тарифных опций

Поддерживаются типы тарифных опций, приведённые в таблице 2.

Таблица 2 — Типы тарифных опций

Тип тарифной опции	SaaS-сервис	Image-based сервис
Числовой	+	+
Строковый	+	+
Логический	+	+
Datasource	—	+

Тип опции **datasource** используется для описания опций, связанных с сущностями облачной платформы. Такой тип позволяет использовать данные облачной платформы в режиме реального времени. На основании этих данных формируются возможные значения опции во время подключения сервиса или обновления тарифного плана.

Поддерживаемые типы **datasource**:

- Типы VM, доступные в проекте пользователя облачной платформы.
- Зоны доступности облачной платформы.
- Виртуальные сети, доступные в проекте облачной платформы.
- Тип диска.
- SSH-ключ для доступа к VM.

1.3. Стоимость тарифных планов и опций


Тарифные планы могут быть платными или бесплатными.

Тарифные планы могут иметь платные тарифные опции. На текущий момент поддерживаются платные опции числового типа.

Стоимость тарифных планов и опций описывается в конфигурации сервиса.

Способы списания денежных средств приведены в таблице 3.

Таблица 3 — Способы списания денежных средств

Способ	Описание
Предоплатный	Стоимость тарифного плана и опций списывается раз в отчётный период в дату подключения тарифного плана. Стоимость тарифных опций фиксирована
Постоплатный (рисунок 2)	<p>Стоимость тарифных опций рассчитывается исходя из фактически использованного количества ресурсов сервиса за месяц (например, количество используемого объёма хранилища).</p> <p>Оплата использованных ресурсов сервиса (тарифных опций) осуществляется на основании отчёта, полученного от сервиса.</p> <p>Периодичность списания равна периодичности, с которой сервис формирует отчёт. Например, если сервис формирует отчёт раз в месяц, то стоимость будет списываться раз в месяц.</p> <p> Постоплатный способ списания применяется только для тарифных опций в бесплатном тарифном плане.</p>



Текущая версия Магазина не поддерживает постоплатный способ списания денежных средств для image-based сервисов.



Рисунок 2 — Передача отчёта между Магазином и брокером

Магазин периодически опрашивает брокера на наличие необработанных отчётов о фактически использованных ресурсах сервиса (постоплатные тарифные опции):

1. Если в брокере есть необработанные отчёты, то брокер передаёт их Магазину.
2. Если в брокере нет необработанных отчётов, то выполняются следующие действия:
 - а. Брокер запрашивает отчёт у сервиса.

- б. По запросу брокера сервис передает ему отчет по всем существующим инстансам сервиса.
 - в. Брокер присваивает отчету идентификатор `batch_id`.
 - г. Брокер в ответ на запрос Магазина передает ему необработанный отчет.
3. Магазин рассчитывает стоимость использованных ресурсов на основании полученного отчета.
 4. После обработки отчета (списание денежных средств по этому отчету произошло или запланировано) Магазин сообщает брокеру `batch_id` этого отчета. В дальнейшем обработанные отчеты брокер не будет передавать Магазину.

1.4. Матрица тарифных планов

Матрица тарифных планов (рисунок 3) отображается на странице тарифов при просмотре информации о сервисе. На основании матрицы пользователь может сравнить тарифы сервиса между собой: какие опции в какой тарифный план входят.

Тарифы и условия	Бесплатный	Базовый
Стоимость	Бесплатно	2 000 ₽ в месяц
	Выбрать тариф	Выбрать тариф
Доступ к открытому API ⓘ	1000	5000
Количество групп пользователей ⓘ	0	5
Количество продуктов ⓘ	5	50
Количество участников ⓘ	5	20
Объем загружаемых сборок ⓘ	5	25
Выгрузка отчетов в CSV ⓘ	×	✓
Перенос отчетов в Jira и GitHub Issues ⓘ	×	✓
Сопровождение менеджером ⓘ	×	×
Уведомления о новых отчетах ⓘ	×	✓

Рисунок 3 — Матрица тарифных планов

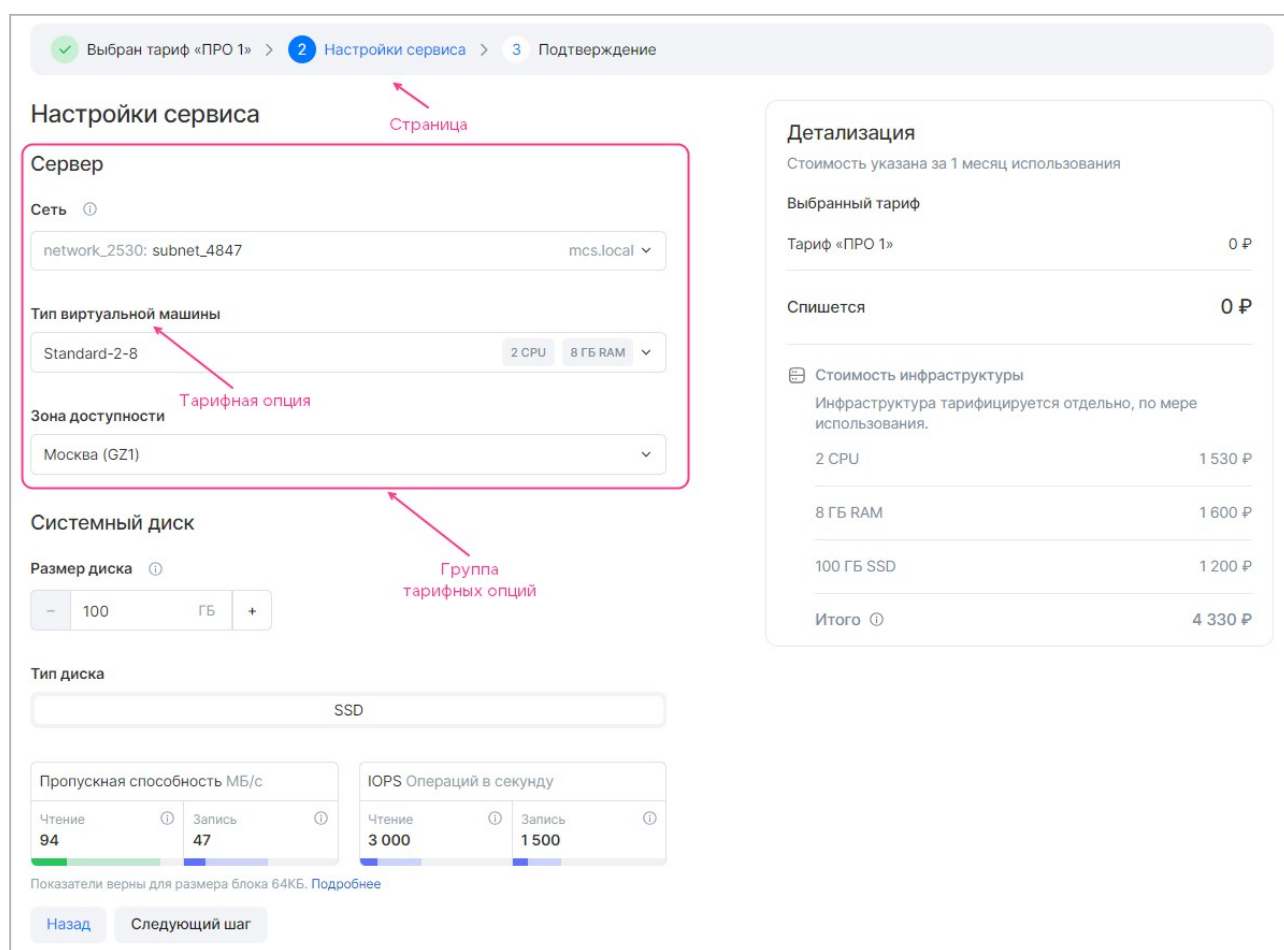
Матрица тарифных планов описывается в конфигурации сервиса. В ней указываются тарифные опции сервиса, на основании которых пользователь будет выбирать тарифный план.

1.5. Мастер конфигурации тарифного плана

При подключении сервиса и при обновлении тарифного плана пользователю отображается мастер конфигурации тарифного плана.

Мастер конфигурации тарифного плана позволяет пользователю настроить тарифные опции для конкретного тарифного плана.

В интерфейсе Магазина мастер конфигурации тарифного плана разбит на отдельные страницы (рисунок 4). Каждая страница описывает конкретный этап настройки тарифного плана. На странице отображаются тарифные опции, настраиваемые на этом этапе. Опции объединены в группы.



Выбран тариф «ПРО 1» > 2 Настройки сервиса > 3 Подтверждение

Настройки сервиса

Сервер

Сеть ⓘ

network_2530: subnet_4847 mcs.local ▾

Тип виртуальной машины

Standard-2-8 2 CPU 8 ГБ RAM ▾

Зона доступности

Москва (GZ1) ▾

Системный диск

Размер диска ⓘ

100 ГБ +

Тип диска

SSD

Пропускная способность МБ/с

Чтение ⓘ Запись ⓘ

94 47

ИOPS Операций в секунду

Чтение ⓘ Запись ⓘ

3 000 1 500

Показатели верны для размера блока 64КБ. Подробнее

Назад Следующий шаг

Детализация

Стоимость указана за 1 месяц использования

Выбранный тариф

Тариф «ПРО 1» 0 ₽

Спишется 0 ₽

Стоимость инфраструктуры

Инфраструктура тарифицируется отдельно, по мере использования.

2 CPU	1 530 ₽
8 ГБ RAM	1 600 ₽
100 ГБ SSD	1 200 ₽
Итого ⓘ	4 330 ₽

Рисунок 4 — Мастер конфигурации тарифного плана

Первая и последняя страницы мастера конфигурации тарифного плана добавляются автоматически. Остальные страницы описываются в конфигурации сервиса.

В мастере конфигурации тарифного плана отображается стоимость тарифного плана и платных тарифных опций. Для image-based сервисов дополнительно отображается стоимость инфраструктуры облачной платформы, которая рассчитывается автоматически в соответствии с тарифами облачной платформы. Параметры для расчёта описываются в конфигурации image-based сервиса.

2. Добавление image-based сервиса в Магазин

2.1. Порядок действий

Чтобы добавить image-based сервис в Магазин:

1. [Создайте образ сервиса и загрузите его в облачную платформу.](#)
Рекомендуемый способ — с помощью Packer.
2. [Создайте структуру файлов сервисного пакета.](#)
3. [Подготовьте файлы, описывающие конфигурацию сервиса \(тарифные планы, опции\).](#)
4. [Подготовьте файлы, описывающие конфигурацию инфраструктуры сервиса.](#)
5. [Загрузите и опубликуйте сервис в Магазине.](#)

2.2. Создание образа сервиса

2.2.1. Требования к образу сервиса

Образ сервиса должен быть на базе ОС семейства Linux с архитектурой x86.

Образ сервиса должен содержать следующие программные пакеты:

- Curl.
- Systemd.
- Cloud-init.


2.2.2. Создание с помощью Packer


Чтобы создать образ сервиса с помощью Packer:

1. Установите Packer:
 - а. Скачайте Packer с [официального зеркала VK Cloud](#).
 - б. Распакуйте архив и в переменной среды `Path` укажите путь к распакованному файлу.
 - в. Выполните команду `packer`, чтобы убедиться в успешной установке Packer.

Подробная инструкция приведена в [официальной документации Packer](#).

2. Установите OpenStack CLI (подробнее — в [официальной документации VK Cloud](#)).
3. Скачайте базовый образ ОС, поддерживающий работу с облачными платформами.

 ОС должна удовлетворять [требованиям к образу](#).

 На [официальном сайте OpenStack](#) размещены ссылки на образы некоторых ОС, поддерживающие работу с облачными платформами.

4. Конвертируйте базовый образ в формат **RAW**:

- а. Установите утилиту [qemu-img](#) на локальный компьютер.

Пример установки qemu-img для образов на базе RHEL (CentOS, AlmaLinux, RockyLinux)

```
$ sudo yum install qemu-img
```

Пример установки qemu-img для образов на базе Debian (Ubuntu)

```
$ sudo apt install qemu-utils
```

- б. Выполните команду:

```
$ qemu-img convert -f qcow2 -O raw <INITIAL_IMAGE_NAME> <IMAGE_NAME>
```

где:

- **<INITIAL_IMAGE_NAME>** — имя исходного базового образа. Например, **alt-p9-cloud-x86_64.qcow2**.
- **<IMAGE_NAME>** — имя базового образа в формате **RAW**. Например, **alt-p9-cloud-x86_64.raw**.

5. Загрузите базовый образ в формате **RAW** в облачную платформу:

```
openstack image create --private --container-format bare --disk-format raw --property store=s3 --file <IMAGE_PATH> <IMAGE_NAME>
```

где:

- **<IMAGE_PATH>** — путь к базовому образу.
- **<IMAGE_NAME>** — имя базового образа в формате **RAW**.

ID созданного образа будет отображаться в ЛК облачной платформы.

Подробная инструкция приведена в [официальной документации VK Cloud](#).

6. Сконфигурируйте packer-файл:

а. В переменные окружения запишите ID сети и ID базового образа:

```
$ export NETWORK_ID=<NETWORK_ID>
$ export SOURCE_IMAGE=<IMAGE_ID>
```

где:

- **<NETWORK_ID>** — ID сети. Значение отображается в ЛК облачной платформы на странице со списком сетей.
- **<IMAGE_ID>** — ID базового образа. Значение отображается в ЛК облачной платформы на странице со списком образов.

б. Создайте файл **<FILE_NAME>.pkr.hcl**. Например, **altlinux.pkr.hcl**.

в. В файле опишите конфигурацию VM, на базе которой будет создан образ сервиса (синтаксис — на [официальном сайте Packer](#)).

Пример конфигурации VM с ОС Alt Linux P9 и разворачиванием image-based сервиса из плейбука Ansible

```
# ID сети
variable "network_id" {
  type = string
  default = "${env("NETWORK_ID")}"
  validation {
    condition     = length(var.network_id) > 0
    error_message = <<EOF
The NETWORK_ID environment variable must be set.
EOF
  }
}

# ID базового образа
variable "source_image" {
  type = string
  default = "${env("SOURCE_IMAGE")}"
  validation {
    condition     = length(var.source_image) > 0
    error_message = <<EOF
The SOURCE_IMAGE environment variable must be set.
EOF
  }
}


# Создание VM из базового образа
source "openstack" "altlinux" {
```

```

flavor      = "Standard-2-6"
image_name  = "Alt-Linux-P9-Starter-Kit"
source_image = "${var.source_image}"
config_drive = "true"
networks    = ["${var.network_id}"]
security_groups = ["default", "ssh"]
ssh_username = "altlinux"
use_blockstorage_volume = "false"
volume_availability_zone = "MS1"
}

# Настройка VM
build {
  sources = ["source.openstack.altlinux"]
  provisioner "shell" {
    execute_command = "sudo {{ .Path }}"
    inline = [
      "apt-get update",
      "apt-get install -y irqbalance bash-completion bind-utils qemu-guest-agent",
      "cloud-utils-growpart",
      "systemctl enable qemu-guest-agent"
    ]
  }
  provisioner "ansible" {
    playbook_file = "provision.yml"
    user          = "altlinux"
    sftp_command  = "/usr/libexec/openssh/sftp-server -e"
    use_proxy     = false
  }
}

```

 В конфигурации VM укажите имя пользователя ОС в зависимости от ОС. Список имён приведён в [официальной документации VK Cloud](#).

г. Проверьте созданную конфигурацию:

```
$ packer validate <FILE_NAME>.pkr.hcl
```

где: **<FILE_NAME>.pkr.hcl** — имя packer-файла.

7. Создайте образ сервиса в облачной платформе:

а. Запустите создание образа сервиса:

```
$ packer build <PACKER_FILE>
```

где: **<PACKER_FILE>** — имя packer-файла.

- б. Дождитесь появления сообщения об успешном создании. В сообщении будет указан ID образа сервиса.

Пример сообщения об успешном создании образа сервиса

```
==> Builds finished. The artifacts of successful builds are:
--> openstack.altlinux: An image was created: c6320138-035f-40d8-XXXX-
e814edb2ce5f
```


где: **c6320138-035f-40d8-XXXX-e814edb2ce5f** — ID образа сервиса.

- в. Убедитесь, что образ сервиса отображается в ЛК облачной платформы.

2.2.3. Создание через ЛК облачной платформы

Чтобы создать образ сервиса через ЛК облачной платформы:

1. Войдите в ЛК облачной платформы.
2. Создайте VM:
 - а. В меню слева перейдите в подраздел **Облачные вычисления** → **Виртуальные машины**.
 - б. Нажмите кнопку **Добавить**.
 - в. В форме создания VM задайте параметры VM.

 ОС должна удовлетворять [требованиям к образу](#).

- г. Подтвердите создание VM.
3. Установите ПО на VM:
 - а. Перейдите на страницу созданной VM.
 - б. Назначьте внешний IP-адрес с помощью кнопки **Назначить внешний IP**.
 - в. Подключитесь к VM через SSH:

```
$ ssh -i <KEY_PATH> <USER_NAME>@<FLOATING_IP>
```

где:

- **<KEY_PATH>** — путь к файлу с закрытым ключом доступа к VM.
- **<USER_NAME>** — имя пользователя ОС. Укажите имя в зависимости от ОС. Список имён приведён в [официальной документации VK Cloud](#).

- `<FLOATING_IP>` — внешний IP-адрес ВМ.

г. Если на ВМ не установлены программные пакеты, указанные в [требованиях к образу сервиса](#), то установите их.

Пример команды, чтобы установить `curl` на ОС Ubuntu

```
$ sudo apt install curl
```

д. Установите ПО image-based сервиса на ВМ.

4. Очистите ВМ от чувствительных данных:


- Для ОС RHEL (CentOS, AlmaLinux, RockyLinux) выполните команды:

```
# Очистить логи cloud-init
$ sudo cloud-init clean --log --seed
# Очистить ssh-ключи
$ sudo rm -f /etc/ssh/*host*key*
# Очистить логи
$ sudo find /var/log -maxdepth 5 -type f -exec rm -fv {} \;
# Удалить tmp-файлы
$ sudo rm -rf /tmp/* /var/tmp/*
# Очистить системные файлы
$ sudo truncate -s 0 /etc/machine-id /etc/resolv.conf /var/log/audit/audit.log
/var/log/wtmp \
/var/log/lastlog /var/log/btmp /var/log/cron /var/log/maillog
/var/log/messages /var/log/secure \
/var/log/spooler
# Удалить системные файлы
$ sudo rm -rf /etc/hostname /etc/machine-info
/var/lib/systemd/credential.secret /var/lib/cloud /var/log/tuned \
/var/log/qemu-ga /var/log/anaconda /var/lib/systemd/random-seed
# Инициализировать диск, заполнить нулями
$ sudo dd if=/dev/zero of=/zeroed_file bs=1M oflag=direct || sudo rm -f
/zeroed_file
# Очистить историю команд
$ history -c
# Синхронизировать файловую систему
$ sudo sync
```

- Для ОС Debian (Ubuntu) выполните команды:

```
# Очистить логи cloud-init
$ sudo cloud-init clean --log --seed
# Очистить ssh-ключи
$ sudo rm -f /etc/ssh/*host*key*
# Очистить логи
$ sudo find /var/log -maxdepth 5 -type f -exec rm -fv {} \;
# Удалить tmp-файлы
$ sudo rm -rf /tmp/* /var/tmp/*
# Очистить файл machine-id
```

```
$ sudo truncate -s 0 /etc/machine-id /var/lib/dbus/machine-id
# Инициализировать диск, заполнить нулями
$ sudo dd if=/dev/zero of=/zeroed_file bs=1M oflag=direct || sudo rm -f /zeroed_file
# Очистить историю команд
$ history -c
# Синхронизировать файловую систему
$ sudo sync
```

 Перед публикацией сервиса образ будет сделан публичным. На основе публичного образа будут разворачиваться инстансы сервиса у пользователей облачной платформы. Данные образа будут общедоступными.

5. Остановите VM:

```
$ sudo /sbin/shutdown -hP now
```

6. Создайте образ сервиса на основе диска VM:

а. В меню слева перейдите в подраздел **Облачные вычисления** → **Образы**.

б. Нажмите кнопку **Создать образ**.

в. В открывшемся окне выберите источник **Диск**.

г. Выберите диск созданной и настроенной VM.

Имя диска VM отображается на её странице на вкладке **Диски**.

д. Укажите имя образа.

е. Подтвердите создание образа. ID созданного образа сервиса будет отображаться в строке этого образа.

2.3. Структура сервисного пакета

Структура файлов сервисного пакета определяется версией генератора JSON-файла. Генератор преобразует файлы из сервисного пакета в файл в формате **JSON**, который необходим брокеру для интеграции image-based сервиса с Магазином. Версия генератора указывается в файле **version.yaml**.

2.3.1. Структура файлов для генератора версии 1.0.0

Чтобы создать файловую структуру для генератора версии **1.0.0**:

1. Создайте директорию сервиса `<SERVICE_NAME>`.
2. В директории сервиса создайте файл `version.yaml`. Внутри файла в параметре `version` укажите версию генератора `1.0.0`:

```
version: 1.0.0
```

3. В директории сервиса создайте следующие директории и файлы:
 - Файл `full_description.md` — содержит полное описание сервиса.
 - Файл `service.yaml` — описывает параметры сервиса и его тарифные планы.
 - Директория `images` — содержит файл с изображением иконки сервиса в Магазине.
 - Файл с расширением `png` или `svg`, использующийся как иконка сервиса.
 - Директория `parameters` — содержит файлы, описывающие опции всех тарифных планов сервиса:
 - Файлы `<OPTION_NAME>.yaml` — описывают опции тарифных планов. Каждый отдельный файл соответствует одной опции. Эти опции будут использоваться при формировании файла, описывающего конкретный тарифный план (файл `plan.yaml` в директории конкретного плана).
 - Директория `plans` — содержит все тарифные планы сервиса. Внутри находятся отдельные директории для каждого плана:
 - Отдельные директории для каждого плана `<PLAN_NAME>` — содержат файлы, описывающие конкретный тарифный план. Одна директория соответствует одному плану:
 - Файл `plan.yaml` — описывает параметры конкретного тарифного плана.
 - Файл `display.yaml` — описывает мастер конфигурации тарифного плана. Мастер конфигурации тарифного плана будет отображаться в Магазине при подключении сервиса и обновлении тарифного плана.
 - Директория `deployment` — содержит манифест Terraform.
 - Файл `deploy.tf` — манифест, описывающий инфраструктуру и процесс разворачивания конкретного тарифного плана сервиса в облачной платформе.

⚠ Для имён файлов и директорий используйте латинские буквы и знаки нижнего подчеркивания в качестве разделителей. В именах не рекомендуется использовать пробелы.

Пример файловой структуры для генератора версии **1.0.0** приведён на рисунке 5.

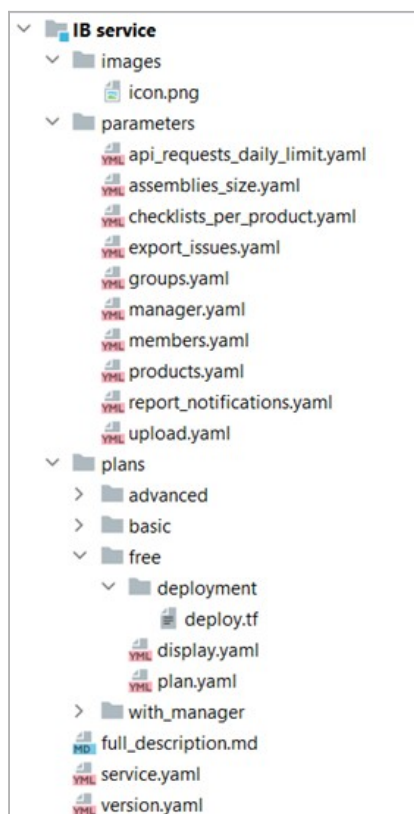


Рисунок 5 — Структура файлов сервисного пакета для генератора версии 1.0.0

2.4. Конфигурация image-based сервиса

Конфигурация сервиса для брокера описывается с помощью файлов формата **YAML** и **MD**. YAML-файлы должны иметь определенную структуру и параметры, которые приведены в следующих подразделах.

Подготовьте следующие конфигурационные файлы:

1. Файл `service.yaml`.
2. Файл `full_description.md`.
3. Файлы тарифных опций `parameters/<OPTION_NAME>.yaml` (подробнее — в подразделе [Заполнение YAML-файлов тарифных опций](#)).
4. Файлы тарифных планов `plans/<PLAN_NAME>/plan.yaml`.

5. Файлы мастера конфигурации тарифного плана для каждого тарифного плана `plans/<PLAN_NAME>/display.yaml`.

2.4.1. Файл `service.yaml`

В файле `service.yaml` укажите параметры и секции, приведённые в таблице 4.

Таблица 4 — Параметры и секции в файле `service.yaml`

Имя	Описание	Формат	Обязательный
<code>id</code>	Идентификатор сервиса UUID4 (ID), сформированный с помощью генератора UUID4	string (UUID4)	Да
<code>revision</code>	Ревизия сервиса. Сочетание ревизии и ID сервиса определяет его уникальность в Магазине. Остальные параметры описывают характеристики конкретной ревизии сервиса	string, до 255 символов	Да
<code>name</code>	Название сервиса	string, до 255 символов	Да
<code>short_description</code>	Краткое описание сервиса, которое будет отображаться в его карточке в Магазине. Полное описание сервиса заполняется в файле <code>full_description.md</code>	string, до 255 символов	Да
<code>singleton</code>	Определяет, есть ли ограничение в один инстанс сервиса на один проект облачной платформы	boolean	Нет
<code>auto_bind</code>	Определяет, нужно ли после разворачивания сервиса автоматически создавать сервисную привязку	boolean	Нет
<code>icon</code>	Определяет файл с изображением для иконки сервиса. Укажите имя файла, находящегося в директории <code>images</code>	string, до 512 символов	Нет
<code>help</code>	URL документации сервиса	string, до 512 символов	Нет
<code>bindable</code>	Определяет, можно ли создавать сервисные привязки для этого сервиса. Значение должно быть <code>true</code>	boolean	Да
<code>plan_updateable</code>	Определяет, может ли пользователь переходить с одного тарифного плана на другой без удаления сервиса. Значение параметра применяется для всех планов сервиса. Для image-based сервисов значение должно быть <code>false</code>	boolean	Да

Имя	Описание	Формат	Обязательный
deactivatable	Определяет, можно ли временно приостановить использование сервиса	boolean	Да
bindings_retrievable	Определяет, нужно ли повторять попытку создания сервисной привязки в течение определенного времени, если предыдущая попытка не удалась	boolean	Да
instances_retrievable	Определяет, нужно ли повторять попытку создания инстанса сервиса в течение определенного времени, если предыдущая попытка не удалась	boolean	Да
plans	Определяет тарифные планы сервиса (подробнее — в подразделе Тарифные планы сервиса)	Массив	Да
preview	Определяет тарифные опции для матрицы тарифных планов (подробнее — в подразделе Тарифные опции для матрицы тарифных планов)	Массив	Да

⚠ Сочетание ID и ревизии сервиса должно быть уникальным в рамках Магазина. Если сервис с такими же идентификатором и ревизией уже существует в Магазине, то сервисный пакет не будет обновлен.

На основании тарифных планов и опций, указанных в секциях `plans` и `preview`, формируется матрица тарифных планов (подробнее — в подразделе [Матрица тарифных планов](#)).

Пример файла `service.yaml` приведен в приложении [Пример файла service.yaml](#).

2.4.1.1. Тарифные планы сервиса

Чтобы указать все тарифные планы сервиса:

1. Укажите массив `plans`.
2. Внутри массива перечислите имена тарифных планов `<PLAN_NAME>` в ключах `name`. Имена тарифных планов должны соответствовать именам директорий этих планов.

⚠ Пользователю будут доступны только тарифные планы, указанные в массиве `plans`.

Перечисление тарифных планов сервиса

```
plans:
  - name: <PLAN_NAME1>
  - name: <PLAN_NAME2>
```

2.4.1.2. Тарифные опции для матрицы тарифных планов

Чтобы указать тарифные опции для матрицы тарифных планов (из всех возможных, описанных в директории `parameters`):

1. Укажите секцию `preview` с массивом `parameters` внутри.
2. Внутри массива перечислите имена тарифных опций `<OPTION_NAME>` с помощью ключей `name`. Имена тарифных опций должны соответствовать именам YAML-файлов `parameters/<OPTION_NAME>.yaml`.

Массив может быть пустым.

В матрице тарифных планов опции будут отображаться с именами, заданными в YAML-файлах.

Перечисление тарифных опций сервиса

```
preview:
  parameters:
    - name: <OPTION_NAME1>
    - name: <OPTION_NAME2>
```

2.4.2. Файл `full_description.md`

В файле `full_description.md` укажите полное описание сервиса. Поддерживается формат Markdown.

2.4.3. Заполнение YAML-файлов тарифных опций

В файлах `parameters/<OPTION_NAME>.yaml` опишите все тарифные опции (подробнее — в подразделе [Типы тарифных опций](#)), которые будут использоваться хотя бы в одном тарифном плане.

Каждый отдельный YAML-файл соответствует одной тарифной опции. В нём описываются настройки тарифной опции, а для платных опций — также её стоимость.

Подробное описание параметров, используемых для описания тарифных опций, приведено в подразделе [YAML-файл тарифной опции](#).

2.4.3.1. Заполнение файла для тарифной опции-константы типа integer

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `integer`.
 - `const` — значение тарифной опции.

Пример описания опции-константы типа `integer`, формат `YAML`

```
actions:
- create
- update

schema:
  description: Размер системного диска
  hint: В ГБ
  type: integer
  const: 20
```

На рисунке 6 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

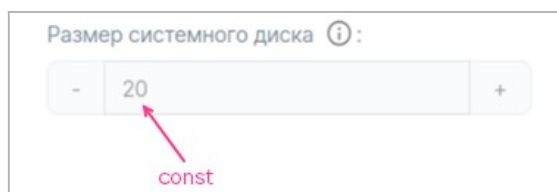


Рисунок 6 — Тарифная опция-константа типа integer

2.4.3.2. Заполнение файла для тарифной опции типа integer с выбором значения из списка

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).

- `type` — тип тарифной опции. Укажите `integer`.
- `enum` — возможные значения тарифной опции.
- `default` — значение по умолчанию.

Пример описания опции типа `integer` с выбором значения из списка, формат `YAML`

```
actions:
- create
- update

schema:
  description: Количество серверов в кластере
  type: integer
  enum: [3, 5, 7]
  default: 5
```

На рисунке 7 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

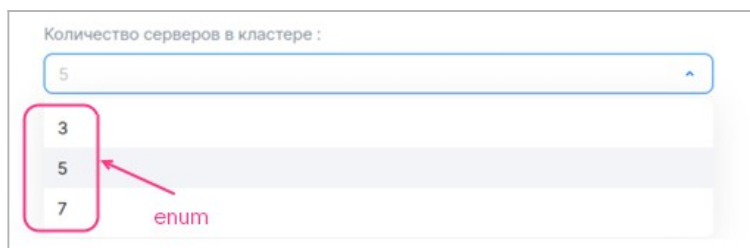


Рисунок 7 — Тарифная опция типа `integer` с выбором значения из списка

2.4.3.3. Заполнение файла для тарифной опции типа `integer` с шагом изменения

2.4.3.3.1. Бесплатная тарифная опция с шагом изменения 1

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `integer`.
 - `default` — значение по умолчанию (опционально).

Если параметр не задан, то значение по умолчанию будет равно `0`.

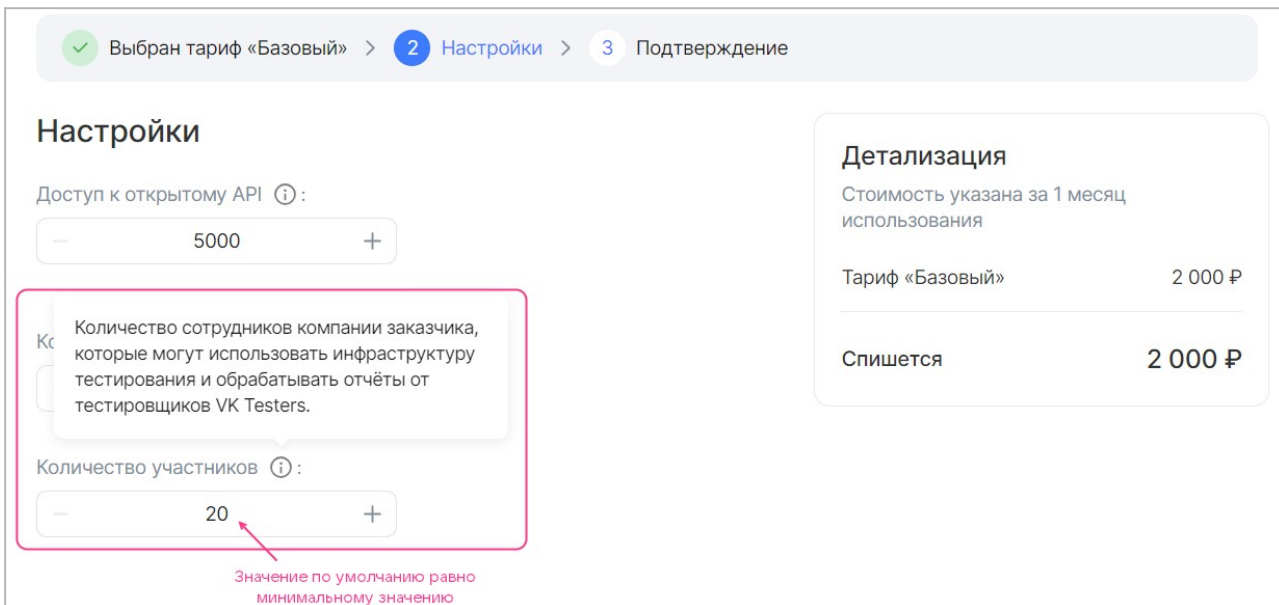
- **minimum** и **maximum** — максимальное и минимальное значения тарифной опции (опционально).

Пример описания бесплатной опции типа **integer** с шагом изменения 1, формат **YAML**

```
actions:
- create
- update

schema:
  description: Количество участников
  hint: Количество сотрудников компании заказчика, которые могут использовать
инфраструктуру тестирования и обрабатывать отчёты от тестировщиков VK Testers.
  type: integer
  default: 20
  minimum: 20
```

На рисунках 8, 9 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ :

— 5000 +

Количество сотрудников компании заказчика, которые могут использовать инфраструктуру тестирования и обрабатывать отчёты от тестировщиков VK Testers.

Количество участников ⓘ :

— 20 +

Значение по умолчанию равно минимальному значению

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый»	2 000 ₽
Спишется	2 000 ₽

Рисунок 8 — Бесплатная тарифная опция типа **integer** с шагом изменения 1

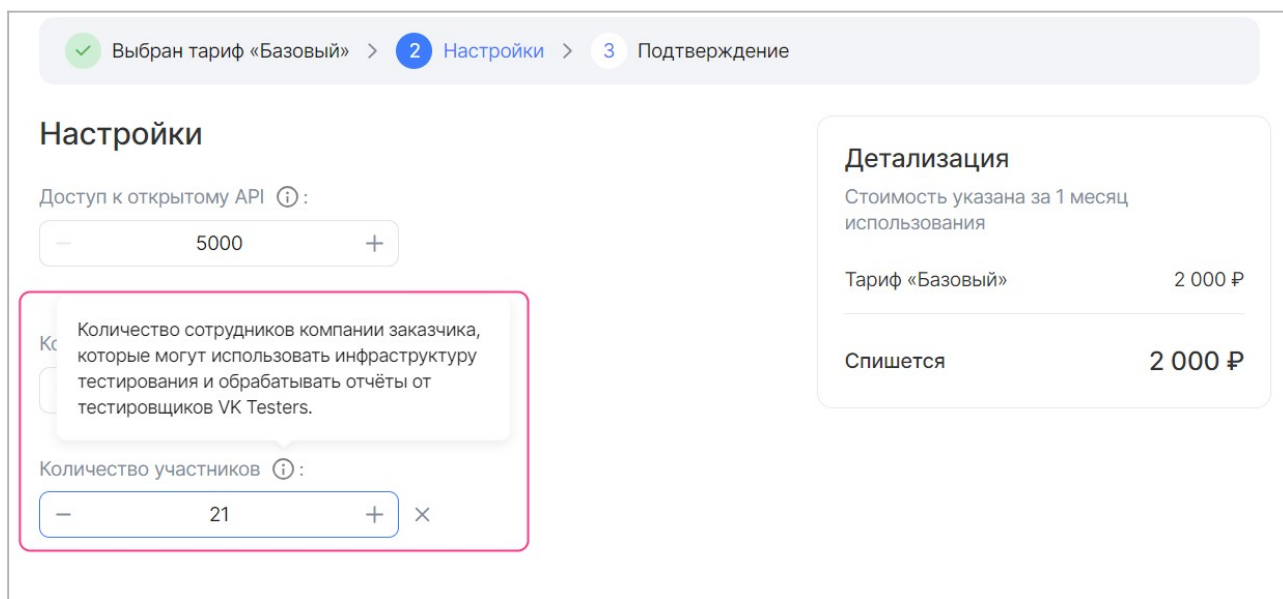


Рисунок 9 — Бесплатная тарифная опция типа `integer` с шагом изменения 1, значение увеличено на 1 шаг

2.4.3.3.2. Бесплатная тарифная опция с пользовательским шагом изменения

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `billing` задайте следующие параметры:
 - `base` — стандартное значение.
 - `cost` — стоимость шага изменения. Укажите `0`.
 - `unit.size` — размер шага изменения.
 - `unit.measurement` — единицы измерения тарифной опции (опционально).

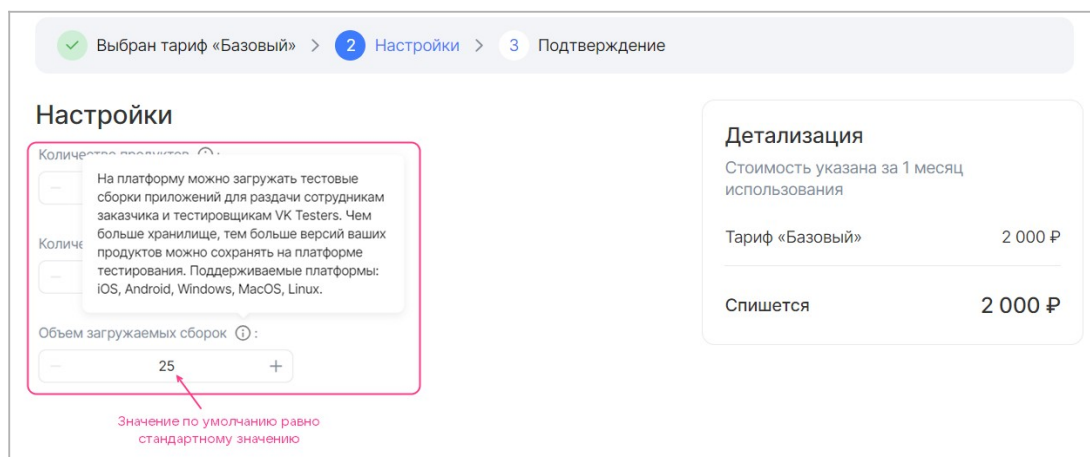
Пример заполнения секции `billing`

```
billing:
  base: 25
  cost: 0
  unit:
    size: 100
```

3. В секции `schema` задайте следующие параметры:

- `description` — имя тарифной опции.
- `hint` — описание тарифной опции (опционально).

- **type** — тип тарифной опции. Укажите **integer**.
- **default** — значение по умолчанию (опционально). Задаётся относительно стандартного значения опции:
 - Не указывайте параметр **default** или укажите **0**, чтобы значение по умолчанию было равно стандартному значению (рисунок 10).



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Количество загрузок: 25

На платформу можно загружать тестовые сборки приложений для раздачи сотрудникам заказчика и тестировщикам VK Testers. Чем больше хранилище, тем больше версий ваших продуктов можно сохранять на платформе тестирования. Поддерживаемые платформы: iOS, Android, Windows, MacOS, Linux.

Объем загружаемых сборок: 25

Значение по умолчанию равно стандартному значению

Детализация

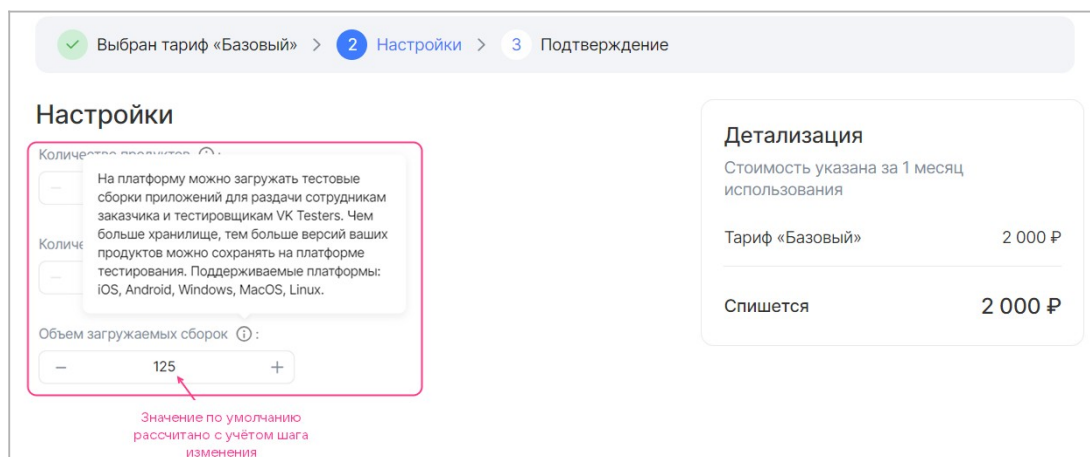
Стоимость указана за 1 месяц использования

Тариф «Базовый»	2 000 ₽
Спишется	2 000 ₽

Рисунок 10 — Бесплатная тарифная опция типа integer с пользовательским шагом изменения (billing.base = 25, schema.default = 0, billing.unit.size = 100)

- Укажите **n**, чтобы значение по умолчанию рассчитывалось по формуле на основе стандартного значения и значения **n** (рисунки 11, 12):

$$\text{billing.base} + n * \text{billing.unit.size}$$



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Количество загрузок: 125

На платформу можно загружать тестовые сборки приложений для раздачи сотрудникам заказчика и тестировщикам VK Testers. Чем больше хранилище, тем больше версий ваших продуктов можно сохранять на платформе тестирования. Поддерживаемые платформы: iOS, Android, Windows, MacOS, Linux.

Объем загружаемых сборок: 125

Значение по умолчанию рассчитано с учётом шага изменения

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый»	2 000 ₽
Спишется	2 000 ₽

Рисунок 11 — Бесплатная тарифная опция типа integer с пользовательским шагом изменения (billing.base = 25, schema.default = 1, billing.unit.size = 100)

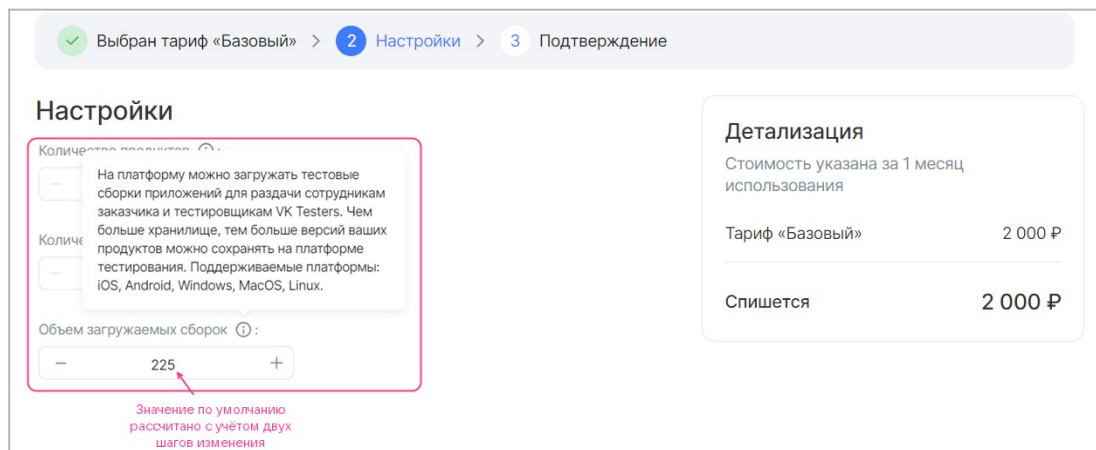


Рисунок 12 — Бесплатная тарифная опция типа integer с пользовательским шагом изменения ($\text{billing.base} = 25$, $\text{schema.default} = 2$, $\text{billing.unit.size} = 100$)


- **minimum** и **maximum** — максимальное и минимальное значения тарифной опции (опционально). Задаётся так же, как это было сделано для значения по умолчанию.

Пример описания бесплатной опции типа **integer** с пользовательским шагом изменения, формат **YAML**

```
actions:
- create
- update

billing:
  base: 25
  cost: 0
  unit:
    size: 100

schema:
  description: Объем загружаемых сборок
  hint: На платформу можно загружать тестовые сборки приложений для раздачи сотрудникам заказчика и тестировщикам VK Testers. Чем больше хранилище, тем больше версий ваших продуктов можно сохранять на платформе тестирования. Поддерживаемые платформы: iOS, Android, Windows, MacOS, Linux.
  type: integer
  default: 0
```

 Подробное описание параметров секций **billing** и **schema** приведено в подразделах [Секция billing](#) и [Секция schema](#) соответственно.

2.4.3.3.3. Платная тарифная опция

Заполните файл **parameters/<OPTION_NAME>.yaml** следующим образом:

1. Укажите параметр **actions**.

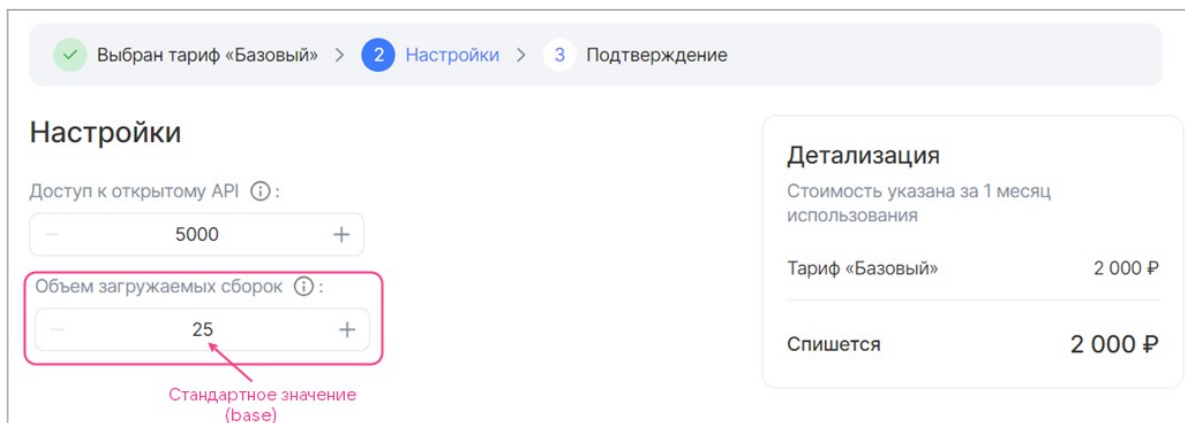
2. В секции **billing** задайте следующие параметры (рисунки 13, 14, 15):

- **base** — стандартное значение. Стандартное значение входит в стоимость тарифного плана.
- **cost** — стоимость шага изменения.
- **unit.size** — размер шага изменения.
- **unit.measurement** — единицы измерения тарифной опции (опционально).

Пример заполнения секции **billing**

```
billing:
  base: 25
  cost: 150
  unit:
    size: 100
```

В примере выше каждые 100 единиц опции, дополнительные к стандартному значению, стоят 150 денежных единиц.



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ:

— 5000 +

Объем загружаемых сборок ⓘ:

— 25 +

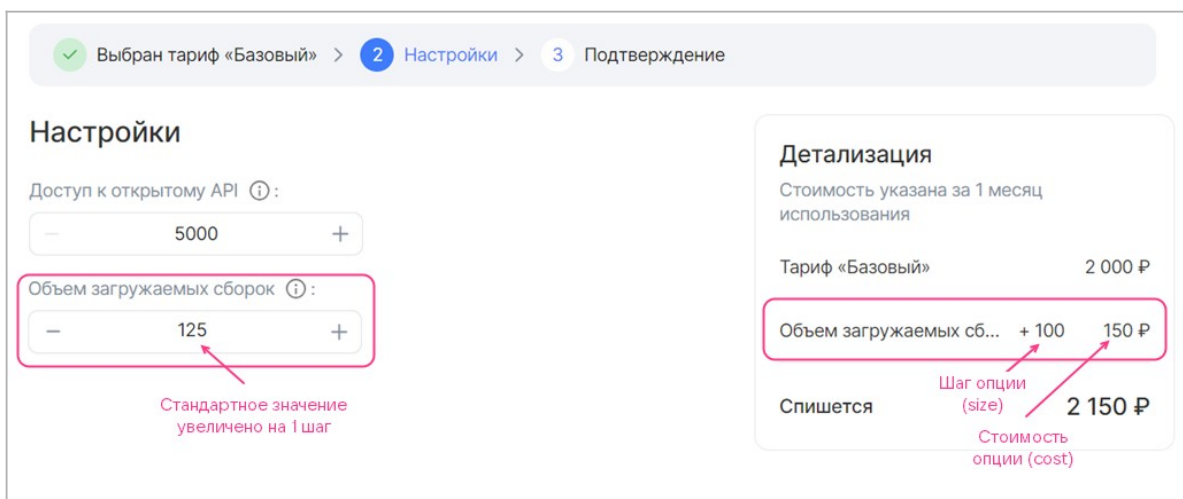
Стандартное значение (base)

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый»	2 000 ₽
Спишется	2 000 ₽

Рисунок 13 — Платная тарифная опция типа integer с шагом изменения (billing.base = 25, billing.cost = 150, billing.unit.size = 100)



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ:

— 5000 +

Объем загружаемых сборок ⓘ:

— 125 +

Стандартное значение увеличено на 1 шаг

Детализация

Стоимость указана за 1 месяц использования

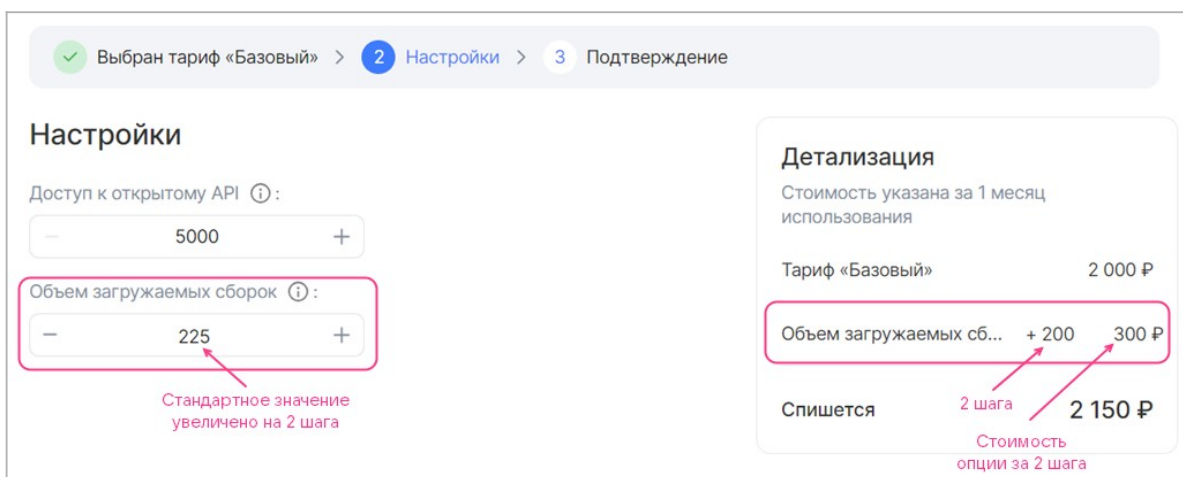
Тариф «Базовый» 2 000 ₽

Объем загружаемых сб... + 100 150 ₽

Спишется 2 150 ₽

Шаг опции (size) Стоимость опции (cost)

Рисунок 14 — Платная тарифная опция типа integer с шагом изменения, значение увеличено на 1 шаг (billing.base = 25, billing.cost = 150, billing.unit.size = 100)



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ:

— 5000 +

Объем загружаемых сборок ⓘ:

— 225 +

Стандартное значение увеличено на 2 шага

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый» 2 000 ₽

Объем загружаемых сб... + 200 300 ₽

Спишется 2 150 ₽

2 шага Стоимость опции за 2 шага

Рисунок 15 — Платная тарифная опция типа integer с шагом изменения, значение увеличено на 2 шага (billing.base = 25, billing.cost = 150, billing.unit.size = 100)

- Заполните секцию `schema` таким же образом, как для бесплатной тарифной опции с пользовательским шагом изменения (подробнее — в подразделе [Бесплатная тарифная опция с пользовательским шагом изменения](#)).

Если для тарифной опции значение по умолчанию не равно стандартному значению (`schema.default ≠ 0`), то, когда пользователь переходит в мастер конфигурации тарифного плана, для такой тарифной опции будет отображаться её стоимость (рисунок 16). Пользователь может уменьшить значение опции до стандартного, которое входит в стоимость тарифного плана.

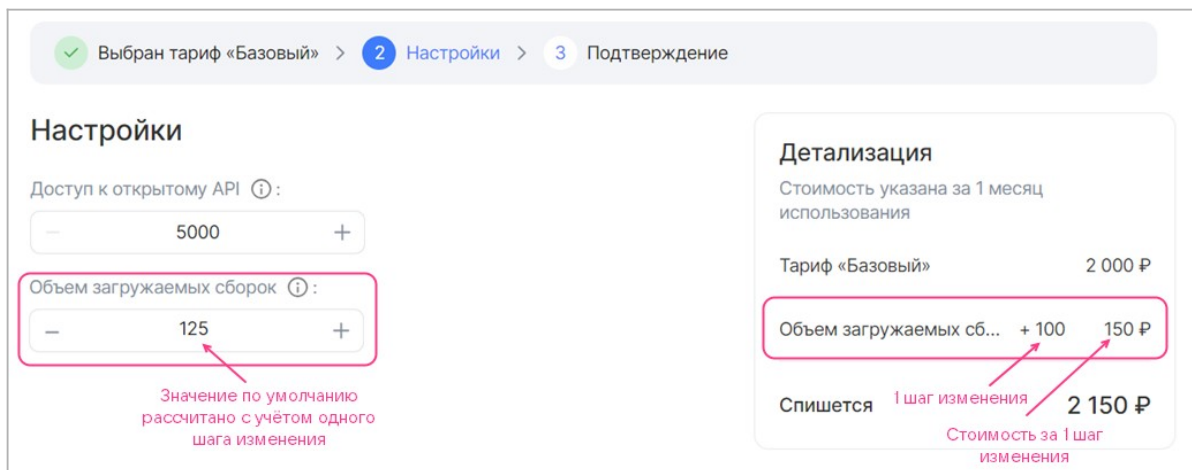



Рисунок 16 — Платная тарифная опция типа integer с шагом изменения ($\text{billing.base} = 25$, $\text{billing.cost} = 150$, $\text{billing.unit.size} = 100$, $\text{schema.default} = 1$)

 Подробное описание параметров секций **billing** и **schema** приведено в подразделах [Секция billing](#) и [Секция schema](#) соответственно.

2.4.3.4. Заполнение файла для тарифной опции-константы типа string

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр **actions**.
2. В секции **schema** задайте следующие параметры:
 - **description** — имя тарифной опции.
 - **hint** — описание тарифной опции (опционально).
 - **type** — тип тарифной опции. Укажите **string**.
 - **const** — значение тарифной опции.

Пример описания опции-константы типа **string**, формат **YAML**

```
actions:
- create
- update

schema:
  description: Логин администратора
  type: string
  const: admin@example.ru
```

На рисунке 17 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.



Рисунок 17 — Тарифная опция-константа типа string

2.4.3.5. Заполнение файла для тарифной опции типа string с вводом значения

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `string`.
 - `default` — значение по умолчанию.
 - Дополнительные параметры, приведённые в подразделе [Секция schema для тарифной опции типа string](#) (опционально).

Пример описания опции типа `string` с вводом значения, формат `YAML`

```
actions:
- create
- update

schema:
  description: Email администратора
  hint: Email для выпуска SSL-сертификата
  type: string
```

На рисунке 18 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

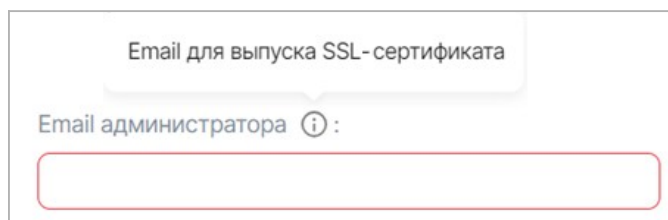


Рисунок 18 — Тарифная опция типа string с вводом значения

2.4.3.6. Заполнение файла для тарифной опции типа string с выбором значения из списка

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `string`.
 - `enum` — возможные значения тарифной опции.
 - `default` — значение по умолчанию.

Пример описания опции типа `string` с выбором значения из списка, формат `YAML`

```
actions:
- create
- update

schema:
  description: OS тип
  hint: Операционная система
  type: string
  enum: ["Ubuntu 20.4", "Windows 8.1", "Windows 10"]
  default: Windows 8.1
```

На рисунке 19 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

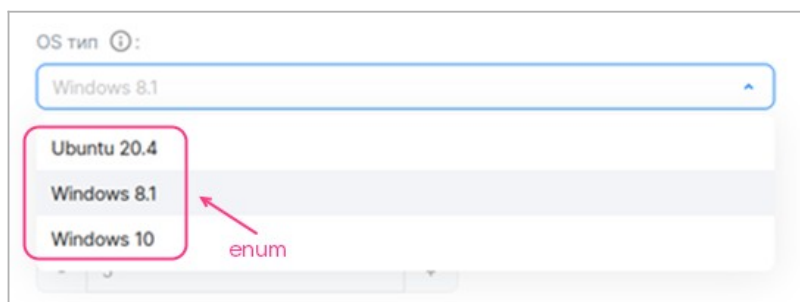


Рисунок 19 — Тарифная опция типа string с выбором значения из списка

2.4.3.7. Заполнение файла для тарифной опции-константы типа boolean

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.

2. В секции `schema` задайте следующие параметры:

- `description` — имя тарифной опции.
- `hint` — описание тарифной опции (опционально).
- `type` — тип тарифной опции. Укажите `boolean`.
- `const` — значение тарифной опции.

Пример описания опции-константы типа `boolean`, формат `YAML`

```
actions:
- create
- update

schema:
  description: Premium поддержка
  hint: Техническая поддержка 24/7
  type: boolean
  const: false
```

На рисунке 20 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

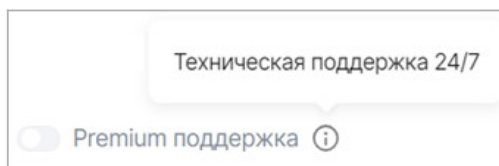


Рисунок 20 — Тарифная опция-константа типа `boolean` (`const = false`)

2.4.3.8. Заполнение файла для тарифной опции-переключателя `boolean`

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:

- `description` — имя тарифной опции.
- `hint` — описание тарифной опции (опционально).
- `type` — тип тарифной опции. Укажите `boolean`.
- `default` — значение по умолчанию (опционально).

Если параметр `default` не задан, то значение по умолчанию будет равно `false`.

Пример описания опции-переключателя `boolean`, формат `YAML`

```
actions:
- create
- update

schema:
  description: Уведомления об обновлениях
  hint: Получать ли на почту уведомления о новых версиях сервиса.
  type: boolean
  default: true
```

На рисунке 21 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

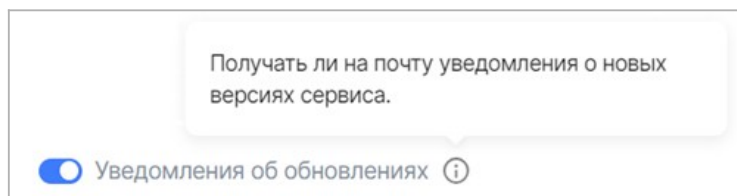


Рисунок 21 — Тарифная опция-переключатель `boolean`

2.4.3.9. Заполнение файла для тарифной опции типа `datasource` (тип `BM`)

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `string`.
 - `datasource.type` — тип сущности облачной платформы. Укажите `flavor`.
 - `datasource.filter` — фильтры (опционально). Возможные фильтры приведены в подразделе [Секция `schema` для тарифной опции типа `datasource`](#).

Пример описания опции `datasource` для типа `BM`, формат `YAML`

```
actions:
- create
- update

schema:
  description: Тип виртуальной машины
  type: string
```

```
datasource:
  type: flavor
```

На рисунке 22 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

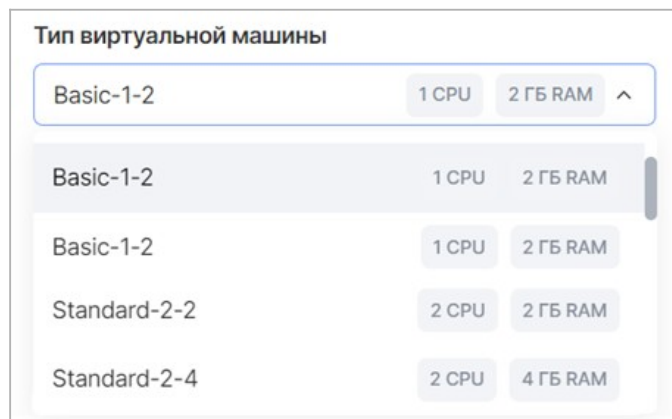


Рисунок 22 — Тарифная опция datasource (тип VM)

2.4.3.10. Заполнение файла для тарифной опции типа datasource (зона доступности)

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `string`.
 - `default` — значение по умолчанию (опционально). Возможные значения приведены в подразделе [Секция schema для тарифной опции типа datasource](#).
 - `datasource.type` — тип сущности облачной платформы. Укажите `az`.

Пример описания опции `datasource` для зоны доступности, формат `YAML`

```
actions:
- create
- update

schema:
  description: Зона доступности
  type: string
  default: gz1
```

```
datasource:
  type: az
```

На рисунке 23 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

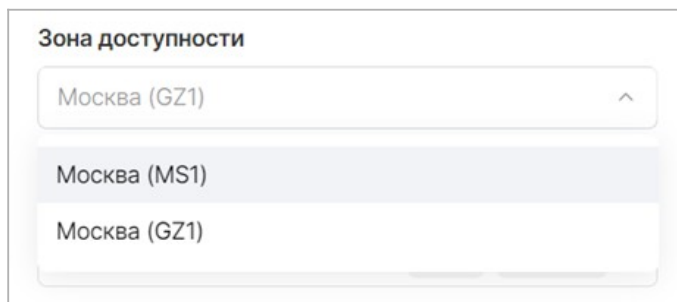


Рисунок 23 — Тарифная опция datasource (зона доступности)

2.4.3.11. Заполнение файла для тарифной опции типа datasource (виртуальная сеть)

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `string`.
 - `datasource.type` — тип сущности облачной платформы. Укажите `subnet`.

Пример описания опции `datasource` для виртуальной сети, формат `YAML`

```
actions:
- create
- update

schema:
  description: Сеть
  type: string
  datasource:
    type: subnet
```

На рисунке 24 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

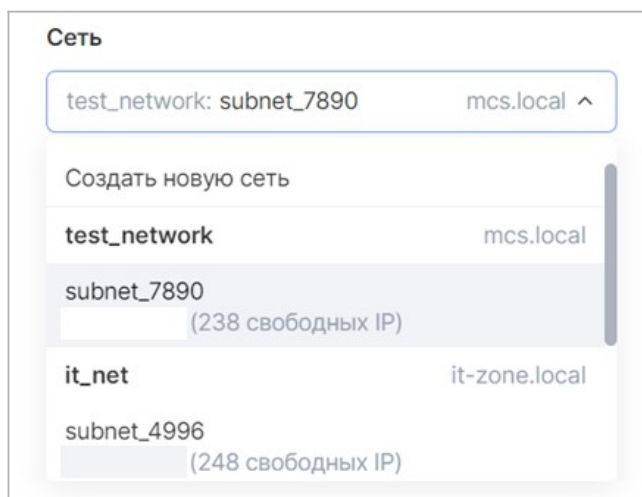


Рисунок 24 — Тарифная опция datasource (виртуальная сеть)

2.4.3.12. Заполнение файла для тарифной опции типа datasource (SSH-ключ)

Заполните файл `parameters/<OPTION_NAME>.yaml` следующим образом:

1. Укажите параметр `actions`.
2. В секции `schema` задайте следующие параметры:
 - `description` — имя тарифной опции.
 - `hint` — описание тарифной опции (опционально).
 - `type` — тип тарифной опции. Укажите `string`.
 - `datasource.type` — тип сущности облачной платформы. Укажите `keypair`.

Пример описания опции `datasource` для SSH-ключа

```
actions:
- create
- update

schema:
  description: Ключ виртуальной машины
  type: string
  datasource:
    type: keypair
```

На рисунках 25, 26 приведено, как вышеописанная опция будет отображаться в мастере конфигурации тарифного плана.

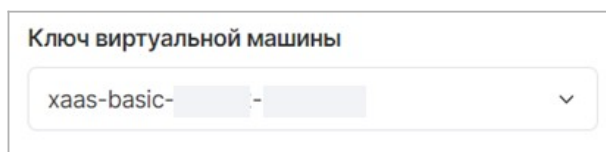


Рисунок 25 — Тарифная опция datasource (SSH-ключ)

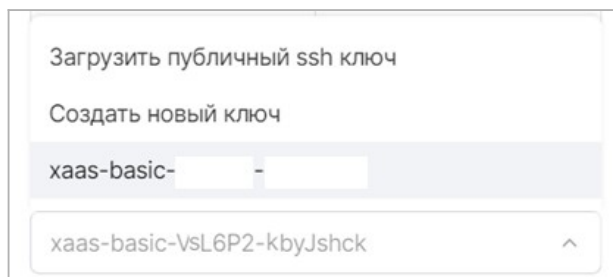


Рисунок 26 — Тарифная опция datasource — SSH-ключ (раскрытый вид)

2.4.3.13. Описание диска с помощью файлов тарифных опций

Диск описывается двумя тарифными опциями (двумя отдельными YAML-файлами):

- Тип диска — с помощью тарифной опции типа `datasource`.
- Размер диска — с помощью тарифной опции типа `integer` с шагом изменения 1.

Чтобы описать диск:

1. Опишите тарифную опцию типа `datasource`, получающую данные облачной платформы о типах дисков, в файле `parameters/<OPTION_NAME>.yaml`:

а. Укажите параметр `actions`.

б. В секции `schema` задайте следующие параметры:

- `description` — имя тарифной опции.
- `hint` — описание тарифной опции (опционально).
- `type` — тип тарифной опции. Укажите `string`.
- `default` — значение по умолчанию (опционально). Возможные значения приведены в подразделе [Секция schema для тарифной опции типа datasource](#).
- `tag` — тег. Тег связывает опцию, описывающую тип диска, с опцией, описывающей размер диска.
- `datasource.type` — тип сущности облачной платформы. Укажите `volume_type`.

- `datasource.filter` — фильтры (опционально). Возможные фильтры приведены в подразделе [Секция schema для тарифной опции типа datasource](#).

Если фильтры не указаны, то будут отображаться все типы дисков, поддерживаемые облачной платформой.

Пример описания опции `datasource` для типа диска, формат `YAML`

```
actions:
- create
- update


schema:
  description: Тип диска
  type: string
  default: ceph-ssd
  tag: disk1
  datasource:
    type: volume_type
    filter:
      disk_class:
        enum: ["ssd", "hdd"]
```

2. В отдельном файле `parameters/<OPTION_NAME>.yaml` опишите размер диска с помощью тарифной опции типа `integer` с шагом изменения 1:

а. Укажите параметр `actions`.

б. В секции `schema` задайте следующие параметры:

- `description` — имя тарифной опции.
- `hint` — описание тарифной опции (опционально).
- `type` — тип тарифной опции. Укажите `integer`.
- `default` — значение по умолчанию (опционально).
- `maximum` и `minimum` — максимальное и минимальное значения (опционально).
- `tag` — тег. Значение должно быть такое же, как в файле, описывающем тип диска.

 Размер диска измеряется в ГБ.

Пример описания размера диска через тарифную опцию типа `integer` с шагом изменения 1, формат `YAML`

```
actions:
- create
- update

schema:
  description: Размер диска
  type: integer
  default: 100
  minimum: 100
  tag: disk1
```



Стоимость диска определяется тарифами облачной платформы, поэтому её нельзя задать в описании тарифной опции.

На рисунке 27 приведено, как вышеописанные опции (тип и размер диска) будут отображаться в мастере конфигурации тарифного плана.

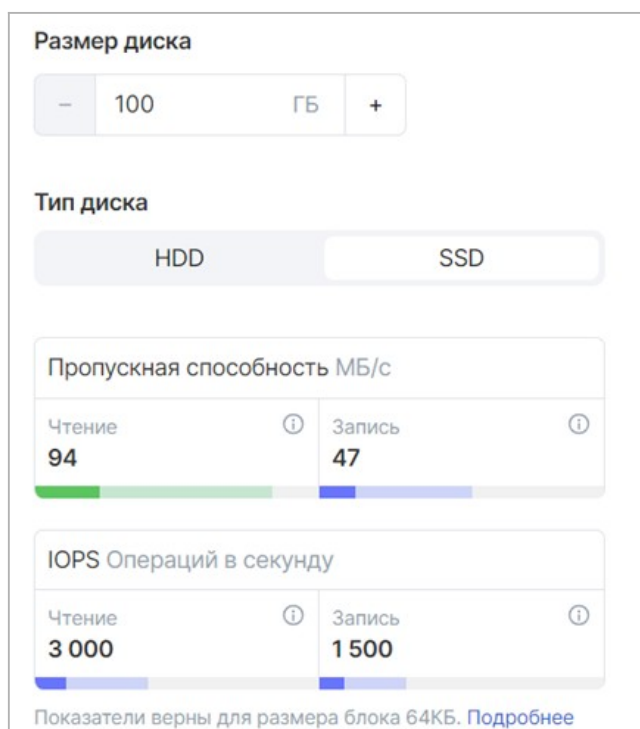



Рисунок 27 — Тарифные опции типа `integer` и `datasource`, позволяющие настроить размер и тип диска

2.4.4. YAML-файл тарифной опции

Тарифная опция описывается параметрами и секциями, приведёнными в таблице 5.

Таблица 5 — Параметры и секции для описания тарифной опции

Имя	Описание	Обязательный
actions	<p>Параметр, определяющий действия, при которых тарифная опция будет активна. Возможные значения параметра:</p> <ul style="list-style-type: none"> <code>create</code> — опция активна при подключении сервиса. <code>update</code> — опция активна при обновлении тарифного плана сервиса. <p>Если действие указано, то тарифная опция будет отображаться пользователю в активном виде.</p> <p>Если действие не указано — в неактивном виде. В неактивном виде пользователь не сможет изменить значение тарифной опции</p>	Да
schema	<p>Секция определяет:</p> <ul style="list-style-type: none"> Тип тарифной опции. Имя и описание тарифной опции (рисунок 28). Настройки значения тарифной опции. <p>Параметры секции приведены в подразделе Секция schema</p>	Да
billing	<p>Секция определяет стоимость тарифной опции, позволяет задать пользовательский шаг изменения для тарифной опции типа <code>integer</code>.</p> <p>Параметры секции приведены в подразделе Секция billing.</p> <div style="background-color: #fff9e6; padding: 10px; border: 1px solid #ccc;"> <p> В текущей версии Магазина поддерживаются платные тарифные опции только типа <code>integer</code> с шагом изменения.</p> </div>	Нет

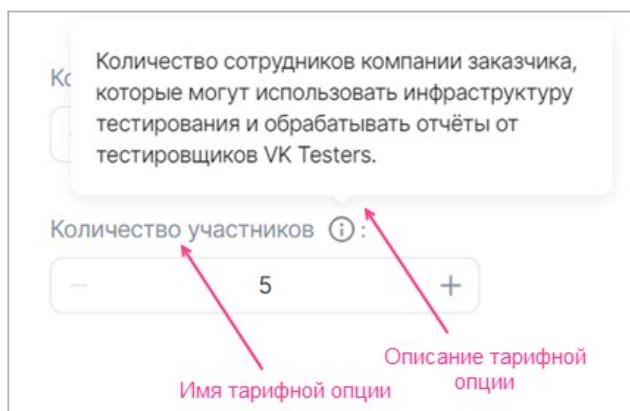


Рисунок 28 — Тарифная опция

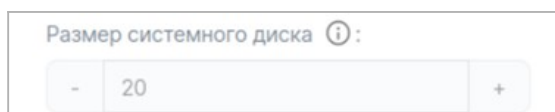
Примеры описания разных типов тарифных опций приведены в подразделе [Заполнение YAML-файлов тарифных опций](#).

2.4.4.1. Секция schema

2.4.4.1.1. Секция schema для тарифной опции типа integer

Подтипы тарифной опции **integer**:

- Константа (рисунок 29).

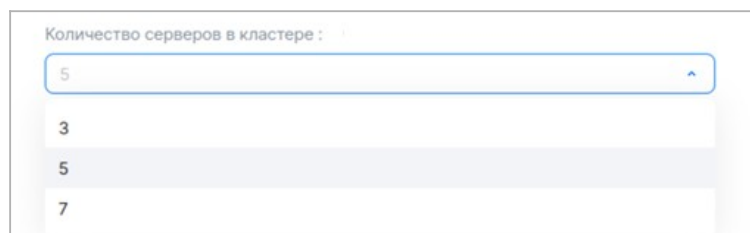


Размер системного диска ⓘ :

- 20 +

Рисунок 29 — Тарифная опция-константа типа integer

- С выбором значения из списка (рисунок 30).



Количество серверов в кластере :

5

3

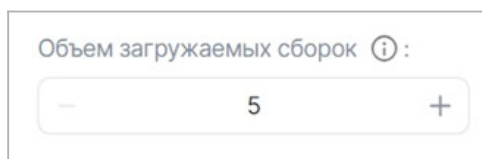
5

7

Рисунок 30 — Тарифная опция типа integer с выбором значения из списка

- С шагом изменения (рисунки 31, 32).

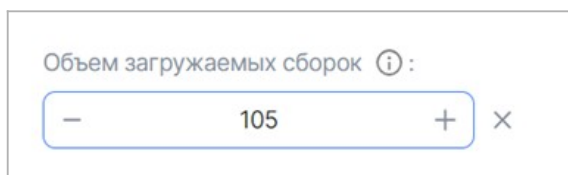
По умолчанию шаг изменения опции равен 1. Изменить размер шага, а также сделать его платным можно в секции [billing](#).



Объем загружаемых сборок ⓘ :

- 5 +

Рисунок 31 — Тарифная опция типа integer с шагом изменения



Объем загружаемых сборок ⓘ :

- 105 + x

Рисунок 32 — Тарифная опция типа integer с шагом изменения, значение увеличено на шаг изменения

Параметры для тарифной опции типа **integer** приведены в таблице 6.

Таблица 6 — Секция schema для тарифной опции типа integer

Имя	Описание	Формат	Обязательный
Основные параметры тарифной опции, одинаковые для всех подтипов			

Имя	Описание	Формат	Обязательный
description	Имя тарифной опции	string, до 255 символов	Да
hint	Подсказка с описанием тарифной опции	string, до 255 символов	Нет
type	Определяет тип тарифной опции. Укажите значение <code>integer</code>	-	Да
Параметры, чтобы настроить подтип тарифной опции			
Параметр для опции-константы			
const	Определяет значение тарифной опции-константы	integer	Да
Параметры для опции с выбором значения			
enum	Определяет список значений, среди которых пользователь сможет выбрать одно	Список, внутри списка — integer	Да
default	Определяет значение тарифной опции по умолчанию	integer	Да
Параметры для опции с шагом изменения 1			
default	Определяет значение тарифной опции по умолчанию. Если параметр не задан, то значение по умолчанию равно <code>0</code>	integer, ≥ 0 или <code>minimum</code> , \leq <code>maximum</code>	Нет
minimum	Определяет минимальное значение тарифной опции	integer, ≥ 0 и \leq <code>maximum</code>	Нет
maximum	Определяет максимальное значение тарифной опции	integer, > 0 и \geq <code>minimum</code>	Нет
tag	Тег. Позволяет связать несколько опций между собой. Используется, чтобы описать диск (подробнее — в подразделе Описание диска с помощью файлов тарифных опций)	string	Нет
Параметры для опции с пользовательским шагом изменения. Шаг изменения настраивается в секции billing			
default	Определяет значение тарифной опции по умолчанию, рассчитанное относительно стандартного значения (стандартное значение задаётся в параметре <code>billing.base</code>): <ul style="list-style-type: none">Если задано значение <code>0</code>, то значение по умолчанию тарифной опции равно стандартному значению (задано в параметре <code>billing.base</code>).Если задано значение <code>n > 0</code>, то значение по умолчанию тарифной опции	integer, ≥ 0 или <code>minimum</code> , \leq <code>maximum</code>	Нет

Имя	Описание	Формат	Обязательный
	<p>складывается из стандартного значения и шага изменения, кратного <code>n</code>:</p> <pre>billing.base + n * billing.unit.size</pre>		
minimum	<p>Определяет минимальное значение тарифной опции, рассчитанное относительно стандартного значения (стандартное значение задаётся в параметре <code>billing.base</code>):</p> <ul style="list-style-type: none"> Если задано значение <code>0</code>, то минимальное значение тарифной опции равно стандартному значению. Если задано значение <code>n > 0</code>, то минимальное значение тарифной опции складывается из стандартного значения и шага изменения, кратного <code>n</code>: <pre>billing.base + n * billing.unit.size</pre>	integer, ≥ 0 и \leq <code>maximum</code>	Нет
maximum	<p>Определяет максимальное значение тарифной опции, рассчитанное относительно стандартного значения (стандартное значение задаётся в параметре <code>billing.base</code>):</p> <ul style="list-style-type: none"> Если задано значение <code>0</code>, то максимальное значение тарифной опции равно стандартному значению. Если задано значение <code>n > 0</code>, то максимальное значение тарифной опции складывается из стандартного значения и шага изменения, кратного <code>n</code>: <pre>billing.base + n * billing.unit.size</pre>	integer, > 0 и \geq <code>minimum</code>	Нет

2.4.4.1.2. Секция schema для тарифной опции типа string

Подтипы тарифной опции `string`:

- Константа (рисунок 33).



Логин администратора :

admin@example.ru

Рисунок 33 — Тарифная опция-константа типа string

- С вводом значения (рисунок 34).

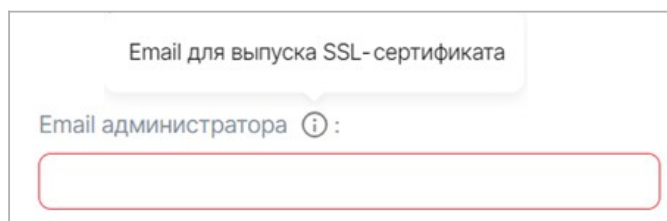


Рисунок 34 — Тарифная опция типа string с вводом значения

- С выбором значения из списка (рисунок 35).

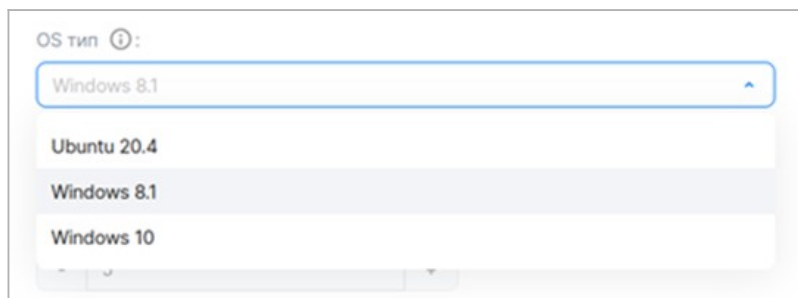


Рисунок 35 — Тарифная опция типа string с выбором значения из списка

Параметры для тарифной опции типа **string** приведены в таблице 7.

Таблица 7 — Секция schema для тарифной опции типа string

Имя	Описание	Формат	Обязательный
Основные параметры тарифной опции, одинаковые для всех подтипов			
description	Имя тарифной опции	string, до 255 символов	Да
hint	Подсказка с описанием тарифной опции	string, до 255 символов	Нет
type	Определяет тип тарифной опции. Укажите значение string	-	Да
Параметры, чтобы настроить подтип тарифной опции			
Параметр для опции-константы			
const	Определяет значение тарифной опции-константы	string	Да
Параметры для опции с вводом значения			
default	Определяет значение по умолчанию	string	Нет
pattern	Определяет шаблон, которому должно соответствовать значение тарифной опции	regex	Нет
minLength	Определяет минимальное количество символов для значения тарифной опции	integer, > 0 и ≤ maxLength	Нет
maxLength	Определяет максимальное количество	integer, > 0 и ≥	Нет

Имя	Описание	Формат	Обязательный
	символов для значения тарифной опции	<code>minLength</code>	
Параметры для опции с выбором значения из списка			
<code>enum</code>	Определяет список значений, среди которых пользователь сможет выбрать одно	Список, внутри списка — <code>string</code>	Да
<code>default</code>	Определяет значение по умолчанию	<code>string</code>	Да

2.4.4.1.3. Секция `schema` для тарифной опции типа `boolean`

Подтип тарифной опции `boolean`:

- Константа (рисунок 36).

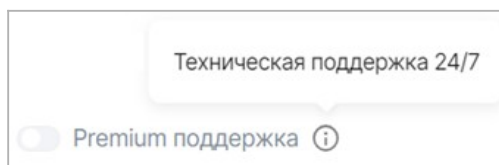


Рисунок 36 — Тарифная опция-константа типа `boolean`

- Переключатель (рисунок 37).

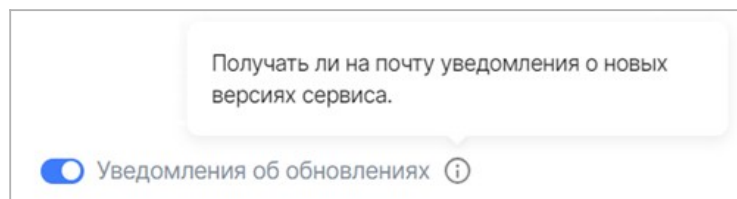



Рисунок 37 — Переключатель `boolean`

Параметры для тарифной опции типа `boolean` приведены в таблице 8.

Таблица 8 — Секция `schema` для тарифной опции типа `boolean`

Имя	Описание	Формат	Обязательный
Основные параметры тарифной опции, одинаковые для всех подтипов			
<code>description</code>	Имя тарифной опции	<code>string</code> , до 255 символов	Да
<code>hint</code>	Подсказка с описанием тарифной опции	<code>string</code> , до 255 символов	Нет
<code>type</code>	Определяет тип тарифной опции. Укажите значение <code>boolean</code>	-	Да
Параметры, чтобы настроить подтип тарифной опции			
Параметр для опции-константы			

Имя	Описание	Формат	Обязательный
const	Определяет значение тарифной опции-константы	boolean	Да
Параметр для опции-переключателя			
default	<p>Определяет значение тарифной опции по умолчанию.</p> <p> Если для переключателя <code>boolean</code> не задан параметр <code>default</code>, то значение по умолчанию будет равно <code>false</code>.</p>	boolean	Нет

2.4.4.1.4. Секция schema для тарифной опции типа datasource

Параметры для тарифной опции типа `datasource` приведены в таблице 9.

Таблица 9 — Секция schema для тарифной опции типа datasource

Имя	Описание	Формат	Обязательный
description	Имя тарифной опции	string, до 255 символов	Да
hint	Подсказка с описанием тарифной опции	string, до 255 символов	Нет
type	Совместно с секцией <code>datasource</code> определяет тарифную опцию как опцию, связанную с сущностями облачной платформы. Укажите значение <code>string</code>	-	Да
default	<p>Определяет значение по умолчанию.</p> <p>Может быть задан только для типов <code>datasource</code>:</p> <ul style="list-style-type: none"> <code>az</code>. <code>volume_type</code>. <p>Возможные значения для <code>az</code> (регион Москва):</p> <ul style="list-style-type: none"> <code>ms1</code>. <code>gz1</code>. <p>Возможные значения для <code>volume_type</code>:</p> <ul style="list-style-type: none"> <code>ceph-ssd</code> — диск типа SSD. <code>ceph-hdd</code> — диск типа HDD. <code>high-iops</code> — диск типа High-IOPS SSD (SSD с повышенной производительностью) 	-	Нет
tag	Тег. Позволяет связать несколько опций между	string	Нет

Имя	Описание	Формат	Обязательный
	с собой. Используется, чтобы описать диск (подробнее — в подразделе Описание диска с помощью файлов тарифных опций)		
datasource	Секция, которая совместно с параметром <code>type</code> определяет тарифную опцию как опцию, связанную с сущностями облачной платформы. Секция описывает конкретный тип сущности облачной платформы	-	Да
Параметры секции <code>datasource</code>			
datasource.type	Параметр, определяющий тип сущности облачной платформы. Возможные значения: <ul style="list-style-type: none"> <code>flavor</code> — типы VM, доступные в проекте пользователя облачной платформы. <code>az</code> — зоны доступности облачной платформы. <code>subnet</code> — виртуальные сети, доступные в проекте облачной платформы. <code>volume_type</code> — тип диска. <code>keypair</code> — SSH-ключ для доступа к VM. Пользователю будет отображаться список значений, соответствующий указанному типу с учётом фильтров (фильтры задаются в секции <code>datasource.filter</code>). Среди этих значений пользователю нужно будет выбрать одно	-	Да
datasource.filter	Секция может быть задана только для типов <code>datasource</code> : <ul style="list-style-type: none"> <code>flavor</code>. <code>volume_type</code>. Определяет фильтры для типа VM или диска	-	Нет
Фильтры для типа VM <code>datasource.filter</code>			
datasource.filter.vcpu	Определяет фильтры для CPU VM	Приведён в таблице Ограничения по CPU и RAM для VM	Нет
datasource.filter.ram	Определяет фильтры для RAM VM	Приведён в таблице Ограничения по CPU и RAM для VM	Нет

Имя	Описание	Формат	Обязательный
Фильтры для типа диска <code>datasource.filter</code>			
<code>datasource.filter.disk_class.enum</code>	Определяет фильтры по типу диска. Возможные значения: <ul style="list-style-type: none"> <code>ssd</code> — диск типа SSD. <code>hdd</code> — диск типа HDD. <code>high_iops_disk</code> — диск типа High-IOPS SSD (SSD с повышенной производительностью) 	Список	Нет
<code>datasource.filter.name.enum</code>	Определяет фильтры по имени диска. Возможные значения: <ul style="list-style-type: none"> <code>ceph-ssd</code> — диск типа SSD. <code>ceph-hdd</code> — диск типа HDD. <code>high-iops</code> — диск типа High-IOPS SSD (SSD с повышенной производительностью) 	Список	Нет

Ограничения по CPU и RAM для VM задаются с помощью параметров, приведённых в таблице 10.

Таблица 10 — Ограничения по CPU и RAM для VM

Имя	Описание	Формат	Обязательный
<code>minimum</code>	Определяет минимальное значение для CPU или RAM — в зависимости от того, в какой секции этот параметр задан. Для RAM значение указывается в МБ	integer, > 0 и \leq <code>maximum</code>	Нет
<code>maximum</code>	Определяет максимальное значение для CPU или RAM — в зависимости от того, в какой секции этот параметр задан	integer, > 0 и \geq <code>minimum</code>	Нет

2.4.4.2. Секция `billing`

Секция `billing` описывается только для тарифных опций типа `integer`, чтобы задать пользовательский шаг изменения или описать стоимость опции.



Поддерживается только предоплатный способ списания денежных средств для платных тарифных опций (подробнее — в подразделе [Стоимость тарифных планов и опций](#)).

В секции `billing` укажите параметры и дочерние секции, приведённые в таблице 11.

Таблица 11 — Параметры секции billing

Имя	Описание	Формат	Обязательный
base	<p>Определяет стандартное значение тарифной опции, входящее в стоимость тарифного плана.</p> <p>Стандартное значение — это минимальное значение, которое может задать пользователь.</p> <p>Если параметр не задан, то стандартное значение тарифной опции автоматически будет равно 0</p>	integer	Нет
cost	<p>Определяет стоимость шага, на который можно изменить значение тарифной опции.</p> <p>Если изменение опции бесплатно, то укажите 0. Параметры шага определяются в секции unit</p>	float64, >= 0	Да
unit	Определяет параметры шага изменения опции	-	Да
Параметры секции unit			
unit.size	<p>Определяет размер шага, на который можно изменить значение тарифной опции.</p> <p>Значение, указанное в этом параметре, тарифицируется в соответствии со стоимостью, заданной в параметре billing.cost</p>	integer, > 0	Да
unit.measurement	Определяет единицы измерения шага, заданного в параметре unit.size	string, до 255 символов	Нет

2.4.5. Файл plan.yaml


В файле `plans/<PLAN_NAME>/plan.yaml` описываются параметры конкретного тарифного плана. После загрузки сервисного пакета и публикации сервиса тарифный план будет доступен в открытых пространствах имён Магазина (`namespace_public`), указанных в сервисном ключе (подробнее — в подразделе [Загрузка image-based сервиса в Магазин](#)).

В файле `plans/<PLAN_NAME>/plan.yaml` укажите параметры и секции, приведённые в таблице 12.

Таблица 12 — Параметры файла plans/<PLAN_NAME>/plan.yaml

Имя	Описание	Формат	Обязательный
id	Идентификатор тарифного плана UUID4 (ID), сформированный с помощью генератора UUID4	string (UUID4)	Да
revision	Ревизия тарифного плана. Сочетание ревизии и ID тарифного плана определяет его	string, до 255 символов	Да

Имя	Описание	Формат	Обязательный
	уникальность в сервисе. Остальные параметры описывают характеристики конкретной ревизии тарифного плана		
name	Техническое название тарифного плана, которое не отображается в интерфейсе Магазина. Должно быть указано латинскими буквами с использованием знака нижнего подчеркивания вместо пробелов	string, до 255 символов	Да
description	Название тарифного плана, которое отображается в интерфейсе Магазина	string, до 255 символов	Да
free	Определяет, бесплатный этот тарифный план или нет	boolean	Да
billing	Определяет стоимость тарифного плана без учёта платных тарифных опций	-	Да
billing.cost	Определяет стоимость тарифного плана без учёта платных тарифных опций (подробнее — в подразделе Секция billing тарифного плана)	-	Да
parameters_patch	Позволяет переопределить параметры тарифных опций для конкретного плана (подробнее — в подразделе Секция parameters_patch)	-	Нет

 Сочетание ID и ревизии тарифного плана должно быть уникальным в рамках сервиса. Если план с такими же идентификатором и ревизией уже существует в этом сервисе, то тарифный план не будет обновлен.

Пример файла `plans/<PLAN_NAME>/plan.yaml` приведён в приложении [Пример файла plan.yaml](#).

2.4.5.1. Секция billing тарифного плана

Для стоимости тарифного плана поддерживается только предоплатный способ списания денежных средств (подробнее — в подразделе [Стоимость тарифных планов и опций](#)).

Чтобы описать стоимость тарифного плана (рисунок 38):

1. В файле `plans/<PLAN_NAME>/plan.yaml` укажите секцию `billing`.
2. Задайте стоимость плана за месяц `<MONTH_COST>` в параметре `cost`:

```
billing:
cost: <MONTH_COST>
```

Если план бесплатный, то укажите **0**. Стоимость сервиса задаётся в валюте страны, где развёрнут Магазин.

К стоимости плана можно добавить платные тарифные опции. Для этого опишите их стоимость в YAML-файлах опций (подробнее — в подразделе [Заполнение YAML-файлов тарифных опций](#)).

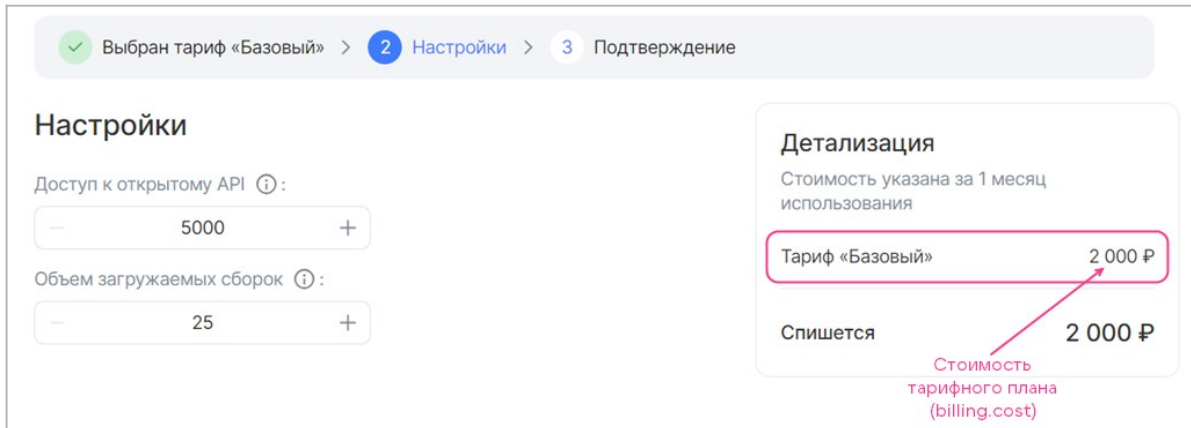


Рисунок 38 — Стоимость тарифного плана



Стоимость использования вычислительных ресурсов облачной платформы в стоимость тарифного плана сервиса не входит.

2.4.5.2. Секция `parameters_patch`

Чтобы для конкретного тарифного плана переопределить параметры тарифных опций или добавить новые, не заданные в YAML-файлах:

- В файле `plans/<PLAN_NAME>/plan.yaml` укажите:
 - Секцию `parameters_patch`.
 - Внутри `parameters_patch` — имена YAML-файлов тарифных опций.
- Задайте значения параметров тарифных опций. Имена параметров укажите с путём до корневой секции. Имя параметра и имена его родительских секций разделите между собой точкой.

Возможные параметры в зависимости от типа тарифной опции приведены в подразделе [YAML-файл тарифной опции](#)

Пример переопределения параметров тарифных опций

```
parameters_patch:
  users:
    schema.const: 5000
  volume_data_size:
```

```
schema.default: 550
schema.minimum: 550
```

Значения параметров, указанные в YAML-файле тарифной опции, не будут применяться в тарифном плане.

Чтобы переопределить секцию тарифной опции полностью:

1. В файле `plans/<PLAN_NAME>/plan.yaml` укажите:
 - Секцию `parameters_patch`.
 - Внутри `parameters_patch` — секцию тарифной опции. Например, `billing`.
2. Задайте параметры секции (подробнее — в подразделе [YAML-файл тарифной опции](#)).

Пример переопределения секции `billing` тарифной опции

```
parameters_patch:
  assemblies_size:
    billing:
      base: 25
      cost: 0
      unit:
        size: 100
```

Параметры, указанные для этой секции в YAML-файле тарифной опции, не будут применяться в тарифном плане.



Одновременное переопределение секции полностью и отдельного параметра из этой секции запрещено.



Если в тарифных планах используются опции, которые сильно отличаются друг от друга, то рекомендуется описать каждую опцию отдельным YAML-файлом в директории `parameters`. Не рекомендуется описывать одну опцию и переопределять большинство её параметров в рамках тарифных планов.

2.4.6. Файл `display.yaml`

Чтобы описать [Мастер конфигурации тарифного плана](#), в файле `plans/<PLAN_NAME>/display.yaml` укажите параметры, приведённые в таблице 13.

Таблица 13 — Параметры файла `plans/<PLAN_NAME>/plan.yaml`

Имя	Описание	Формат	Обязательный
pages	<p>Описывает все страницы мастера конфигурации тарифного плана, кроме первой и последней.</p> <p>Если параметр не указан, то мастер конфигурации будет состоять только из автоматически формируемых страниц</p>	Массив (подробнее — в подразделе Массив pages)	Нет
entities	<p>Описывает элементы инфраструктуры облачной платформы, чтобы в мастере конфигурации тарифного плана отображался расчёт их стоимости:</p> <ul style="list-style-type: none"> • VM. • Балансировщики нагрузки. • Внешние IP-адреса. <p>Стоимость рассчитывается автоматически в соответствии с тарифами облачной платформы</p>	Массив (подробнее — в подразделе Массив entities)	Да, если для инфраструктуры сервиса используются тарифные опции типа <code>datasource</code> для типа VM или диска

2.4.6.1. Массив `pages`

Чтобы описать страницы мастера конфигурации тарифного плана (рисунок 39):

- В файле `plans/<PLAN_NAME>/display.yaml` укажите массив `pages`.
 - В `pages` задайте:
 - Параметр `name` — имя страницы. Не должно превышать 32 символа.
 - Массив `groups`.
 - В `groups` опишите группы тарифных опций для конкретной страницы мастера конфигурации тарифного плана. Для каждой группы задайте:
 - Параметр `name` — имя группы тарифных опций. Не должно превышать 255 символов.
 - Массив `parameters` — тарифные опции, входящие в группу.
 - В `parameters` для каждой тарифной опции укажите параметр `name` — имя её YAML-файла.
- В интерфейсе Магазина тарифные опции будут отображаться с именами, заданными в секции `schema` в их YAML-файлах.
- Опишите последующие страницы мастера конфигурации тарифного плана таким же образом. Максимальное количество страниц — 5.

Рисунок 39 — Мастер конфигурации тарифного плана

```
pages:
- groups:
  - name: Сервер # Имя группы тарифных опций
    parameters:
  - name: network # Имя YAML-файла тарифной опции
  - name: vm
  - name: az

  - name: Системный диск
    parameters:
  - name: volume_size
  - name: volume_type
name: Настройки сервиса # Имя страницы
```

Все тарифные опции плана должны быть указаны в файле `plans/<PLAN_NAME>/display.yaml`.

2.4.6.2. Массив `entities`

Чтобы в мастере конфигурации тарифного плана рассчитывалась стоимость VM (рисунок 40):

1. В файле `plans/<PLAN_NAME>/display.yaml` укажите массив `entities`.
2. В `entities` задайте следующие параметры:

- `entity` — тип элемента инфраструктуры. Укажите `vm`.
- `description` — описание VM (опционально).
- `count.const` или `count.param` — количество VM.

Чтобы задать константу, укажите параметр `count.const` и его значение.

Чтобы количество VM определялось значением тарифной опции, укажите параметр `count.param` и имя соответствующего YAML-файла.

- `flavor.const` или `flavor.param` — тип VM.

Чтобы задать константу, укажите параметр `flavor.const` и ID типа VM.

Чтобы тип VM определялся значением тарифной опции, укажите параметр `flavor.param` и имя YAML-файла, описывающего тип VM (`datasource.type = flavor`).

3. Опишите диски VM с помощью массива `disks` (опционально, если в конфигурации инфраструктуры нет тарифной опции типа `datasource.type = volume_type`).

а. Для каждого диска задайте следующие параметры:

- `type.const` или `type.param` — тип диска.

Чтобы тип диска определялся значением тарифной опции, укажите параметр `type.param` и имя YAML-файла тарифной опции, описывающего тип диска (`datasource.type = volume_type`).

Чтобы задать константу, укажите параметр `type.const` и одно из значений:

- ☐ `ceph-ssd` — диск типа SSD.
- ☐ `ceph-hdd` — диск типа HDD.
- ☐ `high-iops` — диск типа High-IOPS SSD (SSD с повышенной производительностью).
- `size.const` или `size.param` — размер диска.

Чтобы размер диска определялся значением тарифной опции, укажите параметр `size.param` и имя соответствующего YAML-файла тарифной опции.

Выбран тариф «ПРО 1» > 2 Настройки сервиса > 3 Подтверждение

Настройки сервиса

Сервер

Сеть ⓘ

network_2530: subnet_4847 mcs.local ▾

Тип виртуальной машины

Standard-2-8 2 CPU 8 ГБ RAM ▾

Зона доступности

Москва (GZ1) ▾

Системный диск

Размер диска ⓘ

– 100 +

ГБ

Тип диска

SSD

Пропускная способность МБ/с

Чтение ⓘ

94

Запись ⓘ

47

IOPS Операций в секунду

Чтение ⓘ

3 000

Запись ⓘ

1 500

Показатели верны для размера блока 64КБ. Подробнее

Назад

Следующий шаг

Детализация

Стоимость указана за 1 месяц использования

Выбранный тариф

Тариф «ПРО 1» 0 ₽

Спишется

0 ₽

Стоимость инфраструктуры

Инфраструктура тарифицируется отдельно, по мере использования.

2 CPU	1 530 ₽
8 ГБ RAM	1 600 ₽
100 ГБ SSD	1 200 ₽
Итого ⓘ	4 330 ₽

Рисунок 40 — Мастер конфигурации тарифного плана. Информация о стоимости инфраструктуры

Пример описания VM для мастера конфигурации тарифного плана

```
entities:
  - entity: vm
    description: Виртуальная машина
    count:
      const: 1
    flavor:
      param: ds-flavor # Имя YAML-файла тарифной опции
    disks:
      - type:
          param: root_type
          size:
            param: root_size
      - type:
          param: data_type
```



```
size:
  param: data_size
```



Если в конфигурации инфраструктуры сервиса используется тарифная опция типа `datasource`, описывающая тип диска, то при описании VM массив `disks` обязательно должен быть заполнен.

Чтобы в мастере конфигурации тарифного плана рассчитывалась стоимость балансировщика нагрузки:

1. В файле `plans/<PLAN_NAME>/display.yaml` укажите массив `entities`.
2. В `entities` задайте следующие параметры:
 - `entity` — тип элемента инфраструктуры. Укажите `load_balancing`.
 - `count.const` или `count.param` — количество балансировщиков нагрузки.

Чтобы задать константу, укажите параметр `count.const` и его значение.

Чтобы количество балансировщиков нагрузки определялось значением тарифной опции, укажите параметр `count.param` и имя соответствующего YAML-файла.

Пример описания балансировщика нагрузки для мастера конфигурации тарифного плана

```
entities:
- entity: load_balancing
  count:
    param: number_balancing # Имя YAML-файла тарифной опции
```

Чтобы в мастере конфигурации тарифного плана рассчитывалась стоимость внешнего IP-адреса:

1. В файле `plans/<PLAN_NAME>/display.yaml` укажите массив `entities`.
2. В `entities` задайте следующие параметры:
 - `entity` — тип элемента инфраструктуры. Укажите `floating_ip`.
 - `count.const` или `count.param` — количество внешних IP-адресов.

Чтобы задать константу, укажите параметр `count.const` и его значение.

Чтобы количество внешних IP-адресов определялось значением тарифной опции, укажите параметр `count.param` и имя соответствующего YAML-файла.

Пример описания внешнего IP-адреса для мастера конфигурации тарифного плана

```
entities:
  - entity: floating_ip
    count:
      const: 1
```

2.5. Конфигурация инфраструктуры image-based сервиса

Конфигурация инфраструктуры image-based сервиса описывается с помощью манифеста Terraform `plans/<PLAN_NAME>/deployment/deploy.tf` на языке [HashiCorp Configuration Language \(HCL\)](#) (синтаксис — на [официальном сайте](#)).

В манифесте `plans/<PLAN_NAME>/deployment/deploy.tf` описывается инфраструктура для разворачивания инстанса сервиса. Для этого используются ресурсы и источники данных провайдеров из таблицы 14.

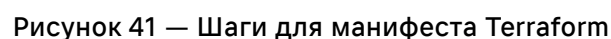
Таблица 14 — Провайдеры Terraform

Имя	Описание	Обозначение в манифесте
VK CS	Содержит ресурсы и источники данных для описания инфраструктуры сервиса. Ресурсы и источники данных приведены в официальной документации провайдера . Примеры содержимого для манифестов, описывающих создание ресурсов, приведены в сценариях в официальной документации VK Cloud	<code>vkcs</code>
VK CS Infra (iVK CS)	Содержит ресурсы и источники данных, расширяющие возможности провайдера VK CS. Например, позволяющие отслеживать состояние инстанса сервиса, запускать скрипты и использовать их результаты в процессе разворачивания. Для скриптов поддерживаются языки Bash и Python. Ресурсы и источники данных приведены в приложении Провайдер iVK CS	<code>ivkcs</code>
null	Ресурсы провайдера используются для настройки получения статуса агента на VM	<code>null</code>

В манифесте используются основные элементы Terraform:

- Входные переменные `variables` для конфигурации ресурсов.
- Ресурсы `resources`, позволяющие управлять объектами инфраструктуры. Например, создать VM. При описании ресурсов можно указывать зависимости с помощью блока `depends on`. Ресурс не будет создан, если не выполняется хотя бы одна зависимость.

- Основные и опциональные шаги, описываемые в манифесте Terraform, приведены на рисунке 41.



Основные шаги для разворачивания сервиса на VM в облачной платформе:


- 59

2. Создание VM из загрузочного образа сервиса.

3. Описание выходных параметров.


Дополнительные возможности:

- **Мониторинг и автовосстановление VM с инстансом сервиса** (отслеживание состояния VM и повторное разворачивание VM, если она вышла из строя).
- Выполнение скриптов в процессе разворачивания или переустановки сервиса (подробнее — в подразделе [Описание скриптов](#)).
- Использование результатов выполнения скриптов другими ресурсами манифеста (подробнее — в подразделе [Описание скриптов](#)).
- Использование DNS облачной платформы (подробнее — в приложении [Провайдер iVK CS](#)).

 Управление созданными ресурсными DNS-записями (например, удаление) в ЛК облачной платформы будет недоступно.

- Создание бакетов S3 (подробнее — в приложении [Провайдер iVK CS](#)). Созданные бакеты будут отображаться в ЛК облачной платформы.
- Перезагрузка VM в процессе разворачивания сервиса (подробнее — в приложении [Провайдер iVK CS](#)).

Дополнительные возможности обеспечиваются ресурсами провайдера iVK CS.

 Идентификатор разворачивания сервиса для ресурсов провайдера iVK CS можно получить с помощью специальной переменной `instance_uuid` (подробнее — в подразделе [Входные переменные для конфигурации ресурсов](#)).

Мониторинг, автовосстановление VM и использование скриптов обеспечиваются специальными сервисами (подробнее — в подразделе [Загрузка сервисного пакета в Магазин](#)), с которыми взаимодействует провайдер iVK CS:

- Сервис управления конфигурациями.
- Агент.

Подготовленный манифест загружается в систему развёртывания (`deployment system`) в составе сервисного пакета. Загрузка сервисного пакета описана в разделе [Загрузка image-based сервиса в Магазин](#).

Пример манифеста для разворачивания сервиса Kafka приведен в приложении [Пример манифеста для сервиса Kafka](#).

2.5.1. Входные переменные для конфигурации ресурсов

Опишите входные переменные для конфигурации ресурсов. Переменные можно условно поделить на группы, приведённые в таблице 15.

Таблица 15 — Переменные Terraform

Группа	Описание	Возможные типы
Специальные	<p>Переменные, позволяющие использовать данные, полученные от брокера.</p> <p>Поддерживаются следующие переменные:</p> <ul style="list-style-type: none"> <code>instance_uuid</code> — идентификатор разворачивания сервиса. Используется для описании большинства ресурсов провайдера iVK CS. <code>email</code> — email пользователя, подключившего сервис. <p>При объявлении таких переменных в манифесте, их значение будет получено Terraform автоматически</p>	string
Внешние	<p>Переменные, позволяющие задать значения тарифных опций, которые настраиваются в мастере конфигурации тарифного плана.</p> <p>Имя переменной должно соответствовать имени YAML-файла тарифной опции.</p> <p>Кроме объявления такой переменной в манифесте, она должна быть:</p> <ul style="list-style-type: none"> Описана как тарифная опция в своем YAML-файле. Указана в тарифном плане (файл <code>plans/<PLAN_NAME>/plan.yaml</code>). Указана в мастере конфигурации тарифного плана (файл <code>plans/<PLAN_NAME>/display.yaml</code>). <p>Для внешней переменной можно задать значение по умолчанию. Оно будет применено, если не удалось получить внешнее значение</p>	number, string, boolean
Локальные	<p>Переменные, использующиеся только в манифесте. Задайте значение в аргументе <code>default</code> при объявлении переменной</p>	Типы, поддерживаемые Terraform

Специальные переменные объявляются следующим образом:

```
# Идентификатор разворачивания сервиса
variable "instance_uuid" {
  type    = string
}
```

```
# Email пользователя
variable "email" {
  type    = string
}
```

Примеры объявления внешних и локальных переменных:

```
# Имя виртуальной сети, где будет создана VM
variable "sub_network" {
  type    = string
  default = "kafka-net"
}

# Количество CPU для VM
variable "flavor_vcpus" {
  type    = number
  default = 2
}
```

При описании входных переменных можно использовать блоки `locals`. С помощью них можно задать переменные с использованием вычислений, в том числе вычислений на основе других объявленных переменных.

Примеры переменных, вычисляемых в блоке `locals`, приведены в приложении [Пример манифеста для сервиса Kafka](#).

2.5.2. Создание VM из загрузочного образа сервиса

Чтобы создать виртуальную машину из загрузочного образа сервиса:

1. Создайте виртуальную сеть для VM. Используйте ресурсы:

- `vkcs_networking_network` — сеть для VM.
- `vkcs_networking_subnet` — подсеть из сети.

Пример создания виртуальной сети

```
resource "vkcs_networking_network" "subnet" {
  # Имя сети, заданное в переменной sub_network
  name = var.sub_network
}

resource "vkcs_networking_subnet" "compute" {
  name          = "kafka_subnet_1"
  # Идентификатор сети, где будет создана подсеть
  network_id    = vkcs_networking_network.subnet.id
  # Диапазон IP-адресов подсети
  cidr          = "192.168.199.0/24"
}
```

2. При необходимости создайте группу безопасности. Используйте ресурсы:

- `vkcs_networking_secgroup` — группа безопасности, в которую будут включены правила доступа.
- `vkcs_networking_secgroup_rule` — правила доступа для группы безопасности.

Пример создания группы безопасности

```
resource "vkcs_networking_secgroup" "secgroup" {
  name = "security_group"
  description = "terraform security group"
}

resource "vkcs_networking_secgroup_rule" "rules" {
  # Определение направления применения правил — для входящих (ingress) или исходящих (egress) соединений
  direction = "ingress"
  # Диапазон портов доступа
  port_range_max = 22
  port_range_min = 22
  # Протокол доступа
  protocol = "tcp"
  # Идентификатор группы безопасности, для которой создано правило
  security_group_id = vkcs_networking_secgroup.secgroup.id
  description = "secgroup_rule"
}
```

3. Создайте отдельный том диска для хранения данных сервиса. Используйте ресурс `vkcs_blockstorage_volume`.

Рекомендуется создавать отдельный том диска для хранения данных, чтобы не потерять их при переустановке сервиса.

Пример создания отдельного тома диска

```
resource "vkcs_blockstorage_volume" "image_data" {
  name = "volume"
  # Размер тома диска, ГБ
  size = 1
  # Зона доступности
  availability_zone = "GZ1"
  # Тип тома диска
  volume_type = "ceph-ssd"
}
```

4. Создайте VM из загрузочного образа сервиса. Используйте:

- Источник данных `vkcs_compute_flavor` — тип VM, от которого зависит количество CPU и RAM.

- Ресурс `vkcs_compute_instance` — инстанс VM. Для ID образа сервиса укажите значение созданного образа (подробнее — в подразделе [Создание образа сервиса](#)), которое отображается в ЛК облачной платформы.

Пример создания VM из загрузочного образа сервиса

```
# Получение доступных типов VM
data "vkcs_compute_flavor" "compute" {
  # Значения CPU и RAM (МБ), фильтрующие доступные типы VM
  vcpus = 2
  ram   = 2048
}

resource "vkcs_compute_instance" "compute" {
  name = "compute-instance"
  # Идентификатор типа VM
  flavor_id = data.vkcs_compute_flavor.compute.id
  # Перечень имён групп безопасности для VM
  security_groups = [vkcs_networking_secgroup.secgroup.name]
  # Зона доступности VM
  availability_zone = "GZ1"
  # Описание конфигурации виртуального root-диска VM
  block_device {
    # ID образа сервиса
    uuid = 45a1f1a3-ad21-4fa2-9364-91101edf54a0
    # Источник загрузки root-диска — образ сервиса (image)
    source_type = "image"
    # Тип диска. Укажите volume (постоянный)
    destination_type = "volume"
    # Тип тома диска
    volume_type = "ceph-ssd"
    # Размер тома диска, ГБ
    volume_size = 10
    # Место диска в порядке загрузки
    boot_index = 0
    # Если указано значение true, диск будет удален при удалении VM
    delete_on_termination = true
  }

  # Подсеть, где будет размещаться VM
  network {
    # Идентификатор сети VM
    uuid = vkcs_networking_network.subnet.id
  }

  # Зависимости. VM будет создана только после создания ресурсов, указанных в
  # зависимостях
  depends_on = [
    # Сеть VM
    vkcs_networking_network.subnet,
    # Подсеть VM
    vkcs_networking_subnet.compute,
    # Диск для хранения данных сервиса
    vkcs_blockstorage_volume.image_data,
    # Правила группы безопасности
    vkcs_networking_secgroup_rule.rules,
  ]
}
```




```

]

# Попытка остановить VM перед удалением
stop_before_destroy = true

}

```

 Перед публикацией сервиса ID образа сервиса будет заменён на ID публичного образа сервиса (подробнее — в подразделе [Публикация образа сервиса](#)).

5. Присоедините созданный отдельный том диска к VM. Используйте ресурс `vkcs_compute_volume_attach`.

Пример присоединения тома диска к VM

```

resource "vkcs_compute_volume_attach" "attached" {
  # Идентификатор созданной VM
  instance_id = vkcs_compute_instance.compute.id
  # Идентификатор присоединяемого тома диска
  volume_id   = vkcs_blockstorage_volume.kafka_data.id


  # Зависимости. Диск будет присоединен к VM только после создания ресурсов,
  # указанных в зависимостях
  depends_on = [
    # VM
    vkcs_compute_instance.compute,
  ]
}

```

2.5.3. Мониторинг и автовосстановление VM с инстансом сервиса

Мониторинг позволяет отслеживать состояние VM и инстанса сервиса.

Автовосстановление позволяет на основании данных мониторинга без дополнительных действий со стороны пользователя выполнить повторное разворачивание виртуальных машин с инстансом сервиса, если они вышли из строя.

 Чтобы исключить потерю данных при переустановке инстанса сервиса, рекомендуется создавать отдельный том диска для VM (подробнее — в подразделе [Создание VM из загрузочного образа сервиса](#)).


Мониторинг осуществляется по следующим параметрам:

- Статус агента, который передается в сервис управления конфигурациями.

- Доступность инстанса сервиса (**health check**). Агент проверяет доступность инстанса сервиса и передает результат в сервис управления конфигурациями.

Автовосстановление будет запущено в следующих случаях:

- Сервер управления конфигурациями получил статус агента **failed**.
- Сервер управления конфигурациями не получил статус агента в течение тайм-аута. По умолчанию тайм-аут равен 5 мин.
- Агент не получил **health check** в течение тайм-аута. По умолчанию тайм-аут равен 5 мин.

 Значение тайм-аута для получения статуса агента и **health check** не настраивается в манифесте Terraform. Если требуется изменить это значение для корректной работы сервиса, отправьте электронное письмо на marketplace@mcs.mail.ru.

Автовосстановление запускает процесс переустановки сервиса, при котором пересоздаются только те виртуальные машины, на которых агент или инстанс сервиса вышли из строя. Процесс переустановки сервиса может выполняться до 1,5 ч.

Чтобы настроить мониторинг и автовосстановление:

1. Получите данные для инициализации агента на VM. Используйте ресурс **ivkcs_agent_init**.

Пример получения данных для агента на VM

```
# Идентификатор разворачивания сервиса
variable "instance_uuid" {
  type    = string
}

resource "ivkcs_agent_init" "init" {
  # Идентификатор разворачивания сервиса
  uuid = var.instance_uuid
  # Имя хоста, где будет установлен агент
  hosts = ["HOST"]
}
```


2. Инициализируйте агент на VM с помощью переменной **user_data**:

```
user_data = ivkcs_agent_init.init.agent[element("<HOST_NAME>")]
```

Внутри переменной укажите имя хоста, где будет инициализирован агент.

Переменную укажите в ресурсе `vkcs_compute_instance` при создании VM (подробнее — в подразделе [Создание VM из загрузочного образа сервиса](#)).


3. Настройте получение статуса агента. Используйте ресурс `ivkcs_agent_status`.

 Автовосстановление применяется только к хосту, указанному в ресурсе `ivkcs_agent_status`.

Пример настройки получения статуса агента на VM

```
resource "ivkcs_agent_status" "status_host" {
  # Имя хоста
  host = "HOST"
  # Идентификатор разворачивания сервиса
  uuid = ivkcs_agent_init.init.uuid
}

# Проброс статуса хоста, полученного в ресурсе ivkcs_agent_status, через транзитный ресурс
resource "null_resource" "replacement_trigger" {
  triggers = {
    status = ivkcs_agent_status.status_host.id
  }
}
```

 Из-за особенностей функционирования Terraform, чтобы сервис управления конфигурациями получил корректный статус агента, пропустите его через транзитный ресурс `null_resource`.

4. Настройте способы мониторинга (типы `health check`) инстанса сервиса (опционально). Используйте ресурс `ivkcs_agent_check`.

Типы `health check` приведены в подразделе [Ресурс ivkcs_agent_check](#) приложения [Провайдер iVK CS](#).

Пример настройки `health check`

```
resource "ivkcs_agent_check" "health" {
  hosts = ["HOST"]
  uuid = var.instance_uuid

  # Мониторинг сервиса по порту
  port_health {
    # IP-адрес
    host = "127.0.0.1"
    # Порт
    port = 9092
  }
}
```

```
# Периодичность мониторинга
period = "1m"
}
```

С указанной периодичностью агент на VM будет передавать результаты мониторинга инстанса сервиса (**health check**) в сервис управления конфигурациями.

- Настройте пересоздание VM с помощью блока конфигурации **lifecycle**. Этот блок укажите в ресурсе **vkcs_compute_instance** при создании VM (подробнее — в подразделе [Создание VM из загрузочного образа сервиса](#)).

Пример блока **lifecycle**

```
lifecycle {
  replace_triggered_by = [
    # Ресурс, содержащий статус агента
    null_resource.replacement_trigger
  ]
}
```

2.5.4. Описание скриптов

Скрипты позволяют выполнить различные операции в процессе разворачивания сервиса (например, синхронизировать узлы при объединении в кластер).

В процессе разворачивания сервиса агент передает хеш выполненных скриптов в сервис управления конфигурациями. Это позволяет оценить текущее состояние VM с инстансом сервиса (какие скрипты выполнены на текущий момент) и привести его к состоянию, описанному в манифесте Terraform.

Агент передает сервису управления конфигурациями статус скриптов. Возможные статусы:

- ok** — скрипт выполнен успешно.
- running** — скрипт выполняется.
- failed** — скрипт завершен с ошибкой. Если получен статус **failed**, то система развёртывания будет пробовать перезапустить разворачивание сервиса. Процесс повторных запусков может занимать до 1,5 ч.

Результаты выполнения скриптов можно передать в сервис управления конфигурациями в форматированном виде и использовать при выполнении других скриптов.

Чтобы использовать скрипты при разворачивании сервиса:

1. Получите данные для инициализации агента на VM. Используйте ресурс `ivkcs_agent_init`.

Пример использования ресурса `ivkcs_agent_init`

```
# Идентификатор разворачивания сервиса
variable "instance_uuid" {
  type    = string
}

resource "ivkcs_agent_init" "init" {
  # Идентификатор разворачивания сервиса
  uuid = var.instance_uuid
  # Имя хоста, где будет установлен агент
  hosts = ["HOST"]
}
```


2. Инициализируйте агент на VM с помощью переменной `user_data`:

```
user_data = ivkcs_agent_init.init.agent[element("<HOST_NAME>")]
```

Внутри переменной укажите имя хоста, где будет инициализирован агент.

Переменную укажите в ресурсе `vkcs_compute_instance` при создании VM (подробнее — в подразделе [Создание VM из загрузочного образа сервиса](#)).

3. Во входных переменных опишите содержимое скриптов. Поддерживаются языки Bash или Python.

 Содержимое скриптов можно описать на следующем шаге в ресурсе `ivkcs_agent_exec` без использования входных переменных.

4. Опишите порядок и настройки выполнения скриптов. Используйте ресурс `ivkcs_agent_exec`.

Для каждого скрипта можно определить опции, описанные в подразделе [Ресурс `ivkcs_agent_exec`](#) приложения [Провайдер iVK CS](#) (например, тайм-аут выполнения, количество попыток выполнения скрипта до определения его статуса как `failed`).

5. Если требуется получить форматированный результат выполнения скрипта для переиспользования в других ресурсах манифеста, то опишите его получение с помощью ресурса `ivkcs_agent_script_result`.

В приложении [Пример манифеста для сервиса Kafka](#) приведен пример манифеста для разворачивания сервиса Kafka с применением скриптов:

- Для получения SSL-сертификата.

- Для синхронизации узлов при объединении в кластер.

2.5.5. Описание выходных параметров

Опишите выходные параметры для передачи пользователю по email после выполнения манифеста и разворачивания сервиса. Например, IP-адрес созданной VM с инстансом сервиса.

Если выходной параметр содержит чувствительные данные, то задайте атрибут **sensitive**. В этом случае данные этого выходного параметра будут переданы только пользователю и не будут сохранены в логи Terraform.

Пример выходного параметра

```
# Вывод IP-адреса VM
output "address" {
  description = "The host IP-address"
  value = vkcs_compute_instance.compute.access_ip_v4
}
```

2.6. Загрузка image-based сервиса в Магазин

2.6.1. Подготовительные операции

У пользователя, загружающего сервисный пакет в Магазин, должна быть учётная запись (личный кабинет) в облачной платформе. Инструкция по регистрации приведена в [официальной документации облачной платформы](#).


Отправьте электронное письмо на marketplace@mcs.mail.ru, в котором укажите следующую информацию:

- Название компании.
- OpenStack PID облачного проекта, в котором будет выполняться тестирование манифестов Terraform.

Значение отображается в ЛК облачной платформы на странице настроек проекта на вкладке с Terraform.

- Email пользователя, которому необходим доступ в тестовые пространства имён Магазина.

В ответном письме будет выслан сервисный ключ. Он содержит тестовые и открытые пространства имён Магазина, в которых будет доступен сервис.

 OpenStack PID проекта должен быть в списке поставщиков, чтобы тестировать манифесты Terraform локально и с системой развёртывания.

2.6.2. Этапы загрузки сервиса

Этапы загрузки image-based сервиса (рисунок 42):

1. [Локальное тестирование манифеста Terraform](#).
2. [Тестирование манифестов с системой развёртывания](#).
3. [Загрузка сервисного пакета в Магазин](#).
4. [Публикация образа сервиса](#).
5. [Публикация сервиса](#).

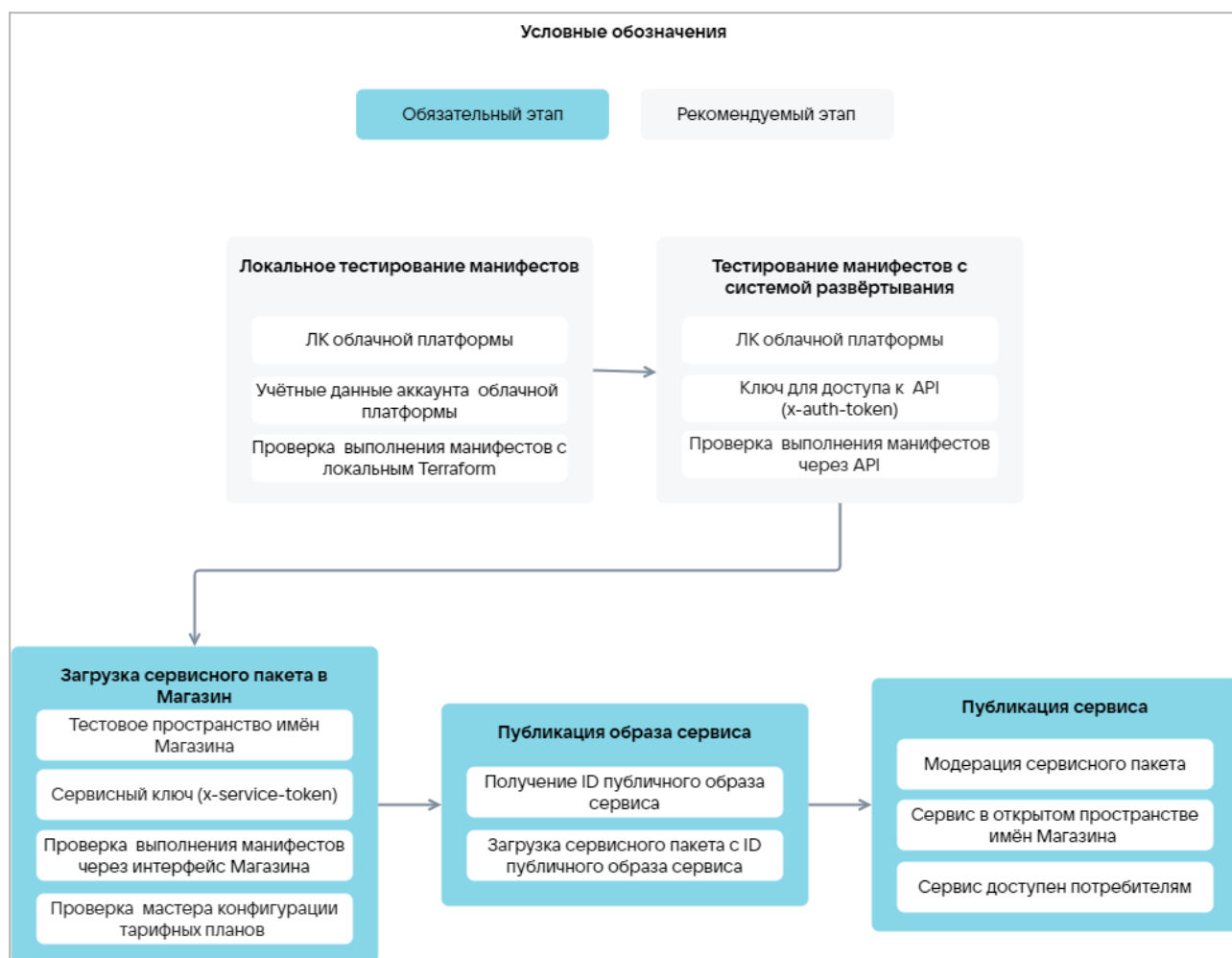



Рисунок 42 — Этапы загрузки image-based сервиса в Магазин

Перед тестированием манифестов Terraform проверьте, что параметры уже существующих ресурсов облачной платформы (например, тип ВМ, образ сервиса) указаны в манифестах верно. Используйте [источники данных провайдера VK CS](#). Установка и настройка Terraform приведена в подразделе [Локальное тестирование манифеста Terraform](#).

Пример проверки того, что указанный тип VM существует в облачной платформе

```
data "vkcs_compute_flavor" "compute" {
  flavor_id = "4e115a9b-0ac2-440d-a120-95cf130d63c7"
}
```

 Проверка параметров существующих ресурсов облачной платформы сократит время тестирования манифестов и исключит попытки создания ресурсов с некорректной конфигурацией.

2.6.3. Локальное тестирование манифеста Terraform

Перед локальным тестированием убедитесь, что OpenStack PID облачного проекта внесён в список поставщиков (подробнее — в подразделе [Подготовительные операции](#)).

Чтобы локально протестировать манифест Terraform `plans/<PLAN_NAME>/deployment/deploy.tf`:

1. Установите Terraform локально:
 - а. Скачайте Terraform с [официального зеркала VK Cloud](#).
 - б. Распакуйте архив и в переменной среды `Path` укажите путь к распакованному файлу.
 - в. Выполните команду `terraform -help`, чтобы убедиться в успешной установке Terraform.
2. В ЛК облачной платформы включите двухфакторную аутентификацию (2FA) и доступ по API (подробнее — в [официальной документации облачной платформы](#)).
3. Отредактируйте файл с конфигурацией CLI для Terraform: для Windows — `terraform.rc`, для других ОС — `.terraformrc` (подробнее — в [официальной документации Terraform](#)):
 - а. Чтобы Terraform одновременно работал с зеркалами и локальными провайдерами, добавьте следующую информацию:

```
provider_installation {
  filesystem_mirror {
    # Полный путь до директории plugins
    path      = "/home/user_name/.terraform.d/plugins"
    include = ["vk-cs.local/*/*", "ivk-cs.local/*/*"]
  }

  network_mirror {
    url = "https://terraform-mirror.mcs.mail.ru"
```



```

    include = ["registry.terraform.io/*/*"]
  }

  direct {
    exclude = ["registry.terraform.io/*/*", "vk-cs.local/*/*", "ivk-
cs.local/*/*"]
  }
}

```


б. Если не требуется, чтобы Terraform одновременно работал с зеркалами и локальными провайдерами, то очистите файл.

в. Если на локальном компьютере такого файла нет, то пропустите этот шаг.

4. Скачайте плагин провайдера VK CS с [GitHub](#).

5. Отправьте электронное письмо на marketplace@mcs.mail.ru, чтобы получить плагин провайдера iVK CS. В письме укажите ОС компьютера, на котором будете локально тестировать манифест, и её архитектуру (например, Linux на архитектуре amd64).

6. Если файла с конфигурацией CLI для Terraform нет на локальном компьютере или он был очищен, то скачайте плагин провайдера null с [официального сайта Terraform](#).

 Если в файле с конфигурацией CLI для Terraform настроена одновременная работа с зеркалами и локальными провайдерами, то Terraform скачает плагин провайдера null автоматически.

7. В директории `terraform.d` создайте директорию `plugins`. Директория `terraform.d` создаётся на компьютере автоматически после установки Terraform, её путь зависит от ОС (подробнее — в [официальной документации Terraform](#)).

8. В директории `plugins` создайте следующую структуру директорий:

- `ivk-cs.local`.
 - `ivk-cs`.
 - `ivkcs`.
 - `<PROVIDER_VERSION>`.
 - `<OPERATING_SYSTEM>`.
- `registry.terraform.io`.
- `hashicorp`.

- `null.`
 - `<PROVIDER_VERSION>.`
 - `<OPERATING_SYSTEM>.`
- `vk-cs.local.`
 - `vk-cs.`
 - `vkcs.`
 - `<PROVIDER_VERSION>.`
 - `<OPERATING_SYSTEM>.`

где:

- `<PROVIDER_VERSION>` — версия плагина провайдера. Значение определяется из имени файла плагина. Например, для плагина `terraform-provider-vkcs_0.4.2` версия равна `0.4.2`, для плагина `terraform-provider-null_v3.2.1_x5` — `3.2.1`.
- `<OPERATING_SYSTEM>` — ОС компьютера. Чтобы определить значение, выполните команду `terraform --version`. В выводе команды отображается текущая версия Terraform и ОС с архитектурой.

Пример ответа

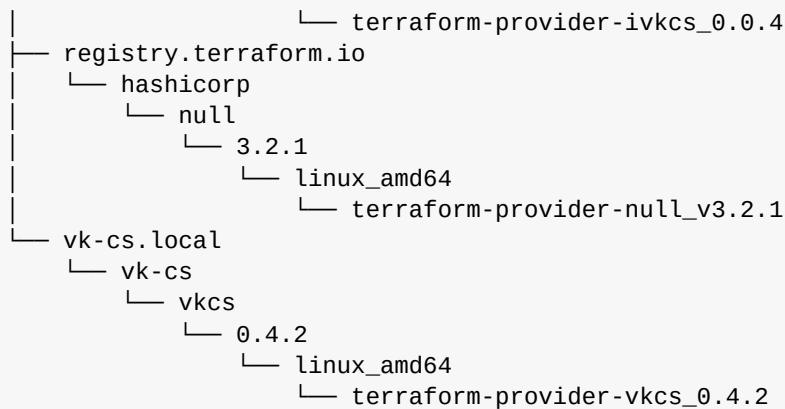
```
Terraform v1.5.4
on linux_amd64
```

В приведённом выше примере значение `<OPERATING_SYSTEM>` равно `linux_amd64`.

9. Поместите плагины скачанных провайдеров в соответствующие директории `<OPERATING_SYSTEM>`.

Пример структуры директории `/home/user_name/.terraform.d/plugins` для ОС Linux на архитектуре amd64

```
/home/user_name/.terraform.d
├── plugins
│   ├── ivk-cs.local
│   │   ├── ivk-cs
│   │   │   ├── ivkcs
│   │   │   │   ├── 0.0.4
│   │   │   │   └── linux_amd64
│   │   └──
│   └──
└──
```



10. Создайте рабочую директорию для тестирования манифеста.
11. В рабочей директории создайте файл `auth.tf`.
12. В файле `auth.tf` опишите используемые провайдеры.
 - а. Укажите провайдеры VK CS и iVK CS. Если файла с конфигурацией CLI для Terraform нет на локальном компьютере или он был очищен, то дополнительно укажите провайдер null.

```

terraform {
  required_providers {
    vkcs = {
      # Путь локального плагина провайдера. Указывается часть пути после
      terraform.d/plugins
      source = "vk-cs.local/vk-cs/vkcs"
      # Версия провайдера
      version = "0.2.2"
    }
    ivkcs = {
      source = "ivk-cs.local/ivk-cs/ivkcs"
      version = "0.1.15"
    }
    null = {
      source = "registry.terraform.io/hashicorp/null"
      version = "3.2.1"
    }
  }
}

```

- б. Для провайдеров VK CS и iVK CS укажите информацию об аккаунте и проекте облачной платформы, из-под которых будете тестировать манифест:

```

provider "vkcs" {
  # User Domain ID
  user_domain_name = "users"
  username = "USER_NAME"
  password = "PASSWORD"
}

```

```
project_id = "PROJECT_ID"
# Region Name
region = "RegionOne"
}

provider "ivkcs" {
  # URL компонента Keystone облачной платформы
  auth_url = "https://mcs.mail.ru/infra/identity/v3/"
  user_domain_name = "users"
  username = "USER_NAME"
  password = "PASSWORD"
  project_id = "OPENSTACK_PROJECT_ID"
  region = "RegionOne"
  # URL сервиса Infra API
  infra_url = "https://mcs.mail.ru/marketplace/api/infra-api/api/v1/"
}
```

Информация об аккаунте и проекте отображается в ЛК облачной платформы.

13. Скопируйте файл `plans/<PLAN_NAME>/deployment/deploy.tf` в рабочую директорию.
14. Если для локального тестирования требуется создать дополнительные сущности, которые не описаны в манифесте `deploy.tf` (например, ВМ с внешним IP), то в рабочей директории создайте файл `extended.tf`.
15. В файле `extended.tf` опишите дополнительные сущности с использованием ресурсов и источников данных провайдеров.
16. В рабочей директории выполните команду `terraform init`, чтобы инициализировать провайдеры.
17. В рабочей директории выполните команду `terraform apply`, чтобы запустить создание ресурсов, описанных в файлах `deploy.tf` и `extended.tf`.
18. Введите `yes`, чтобы подтвердить создание ресурсов.
19. После завершения выполнения манифестов зайдите в ЛК облачной платформы и убедитесь, что ресурсы созданы.

2.6.4. Тестирование манифестов с системой развёртывания

После локального тестирования манифестов Terraform и перед загрузкой сервисного пакета в Магазин рекомендуется протестировать манифесты `plans/<PLAN_NAME>/deployment/deploy.tf` с системой развёртывания. Это позволит убедиться в том, что все описанные ресурсы могут быть созданы системой развёртывания.

Перед тестированием с системой развёртывания убедитесь, что OpenStack PID облачного проекта внесён в список поставщиков (подробнее — в подразделе [Подготовительные операции](#)).

В процессе тестирования потребуются:

- Ключ для доступа к API. Ключ отображается в ЛК облачной платформы на странице настроек проекта (вкладка с информацией о доступе по API).
- Доменное имя облачной платформы — `https://mcs.mail.ru`.

Чтобы протестировать манифест Terraform с системой развёртывания:

1. В ЛК облачной платформы включите двухфакторную аутентификацию (2FA) и доступ по API (подробнее — в [официальной документации облачной платформы](#)).
2. Загрузите манифест в систему развёртывания.
3. Проверьте текущую конфигурацию загруженного манифеста.
4. Создайте инстанс сервиса.
5. Удалите инстанс сервиса.
6. В процессе создания, после создания и после удаления инстанса сервиса проверьте его состояние.

Параметры API-запросов описаны в следующих подразделах.

2.6.4.1. Загрузка манифеста Terraform в систему развёртывания

Чтобы загрузить манифест `plans/<PLAN_NAME>/deployment/deploy.tf` в систему развёртывания, выполните запрос с параметрами, приведёнными в таблице 16.

Таблица 16 — Параметры запроса на загрузку манифеста

Параметр	Значение
Метод запроса	<code>POST</code>
Путь запроса	<code>https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/hoe/config/<MANIFEST_NAME></code> где: <ul style="list-style-type: none"> • <code><CLOUD_HOST></code> — доменное имя облачной платформы. • <code><MANIFEST_NAME></code> — имя манифеста
Тело запроса	Содержимое манифеста <code>plans/<PLAN_NAME>/deployment/deploy.tf</code> . Если для тестирования требуются дополнительные ресурсы провайдеров, то добавьте их в тело запроса
<code>x-auth-token</code>	Ключ для доступа к API

HTTP-коды ответа приведены в таблице 17.

Таблица 17 — HTTP-коды ответа

Код	Описание
201	Манифест загружен или обновлен
400, 500	Ошибка выполнения запроса
401	Ошибка авторизации

При отправке запроса с именем манифеста, уже существующим в системе развёртывания, конфигурация этого манифеста будет обновлена.

Манифесты загружаются в рамках одного аккаунта пользователя.

Все ранее загруженные конфигурации манифеста сохраняются в системе развёртывания, чтобы обеспечить корректную работу Terraform. Инстанс сервиса разворачивается с текущей конфигурацией манифеста, которую можно просмотреть с помощью GET-запроса `/hoe/config/<MANIFEST_NAME>`.

Пример запроса на загрузку манифеста

```
$ curl -v -X POST
https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/hoe/config/test_1.0 \
-H 'x-auth-token: <AUTH_TOKEN>' \
--data-binary "@deploy.tf"
```

где: `<AUTH_TOKEN>` — ключ для доступа к API.

2.6.4.2. Проверка текущей конфигурации манифеста Terraform

Чтобы проверить текущую конфигурацию манифеста Terraform, находящуюся в системе развёртывания, выполните запрос с параметрами, приведёнными в таблице 18.

Таблица 18 — Параметры запроса на проверку текущей конфигурации манифеста

Параметр	Значение
Метод запроса	<code>GET</code>
Путь запроса	<code>https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/hoe/config/<MANIFEST_NAME></code> где: <ul style="list-style-type: none"> <code><CLOUD_HOST></code> — доменное имя облачной платформы. <code><MANIFEST_NAME></code> — имя манифеста
<code>x-auth-token</code>	Ключ для доступа к API

HTTP-коды ответа приведены в таблице 19.

Таблица 19 — HTTP-коды ответа

Код	Описание
200	Запрос выполнен
401	Ошибка авторизации
404	Манифест не найден
500	Ошибка выполнения запроса

В ответе на запрос передается содержимое текущей конфигурации указанного манифеста Terraform.

Пример запроса на проверку текущей конфигурации манифеста

```
$ curl
https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/hoec/config/test_1.0 \
-H 'x-auth-token: <AUTH_TOKEN>'
```

где: **<AUTH_TOKEN>** — ключ для доступа к API.

2.6.4.3. Создание инстанса сервиса

Чтобы создать инстанс сервиса, выполните запрос с параметрами, приведёнными в таблице 20. Будут созданы ресурсы текущей конфигурации манифеста Terraform.

Таблица 20 — Параметры запроса на создание инстанса сервиса

Параметр	Значение
Метод запроса	POST
Путь запроса	https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/hoec/object где: <CLOUD_HOST> — доменное имя облачной платформы
Тело запроса	Укажите следующие параметры: <ul style="list-style-type: none"> uuid — идентификатор инстанса сервиса, сформированный с помощью генератора UUID4. config — имя манифеста. vars — внешние входные переменные манифеста. Задание значений для таких переменных в теле запроса имитирует действия пользователя в мастере конфигурации тарифного плана. <i>Пример тела запроса в JSON-формате</i> <pre>{ "uuid": "675f6f08-2344-4cf4-a7f4-f02311f795d7", "config": "<MANIFEST_NAME>", "vars": { "sub_network": "my-net",</pre>

Параметр	Значение
	<pre>"image_uuid": "163ff752-1390-4b72-a23c-b0001e3e65d3" }</pre>
<code>x-auth-token</code>	Ключ для доступа к API

HTTP-коды ответа приведены в таблице 21.

Таблица 21 — HTTP-коды ответа

Код	Описание
201	Инстанс сервиса создан
400, 500	Ошибка выполнения запроса
401	Ошибка авторизации

Пример запроса на создание инстанса сервиса

```
$ curl -v -X POST
https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/hoе/object \
-H "Content-Type: application/json" \
-H 'x-auth-token: <AUTH_TOKEN>' \
--data "{
  \"uuid\": \"675f6f08-2344-4cf4-a7f4-f02311f795d7\",
  \"config\": \"test_1.0\",
  \"vars\": {
    \"sub_network\": \"my-net\",
    \"image_uuid\": \"163ff752-1390-4b72-a23c-b0001e3e65d3\"
  }
}"
```

где: `<AUTH_TOKEN>` — ключ для доступа к API.

2.6.4.4. Удаление инстанса сервиса

Чтобы удалить инстанс сервиса, созданный в результате выполнения манифеста Terraform, выполните запрос с параметрами, приведёнными в таблице 22.

Таблица 22 — Параметры запроса на удаление инстанса сервиса

Параметр	Значение
Метод запроса	<code>DELETE</code>
Путь запроса	<p><code>https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/hoе/object/<OBJECT_UUID></code></p> <p>где:</p> <ul style="list-style-type: none"> <code><CLOUD_HOST></code> — доменное имя облачной платформы.

Параметр	Значение
	<ul style="list-style-type: none"> <OBJECT_UUID> — идентификатор разворачивания инстанса сервиса. Значение соответствует параметру uuid в запросе на разворачивание инстанса сервиса
x-auth-token	Ключ для доступа к API

HTTP-коды ответа приведены в таблице 23.

Таблица 23 — HTTP-коды ответа

Код	Описание
201	Инстанс сервиса удален
400, 500	Ошибка выполнения запроса
401	Ошибка авторизации
404	Манифест не найден

Пример запроса на удаление инстанса сервиса

```
$ curl -v -X DELETE
https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/hoе/object/675f6f08-2344-4cf4-a7f4-f02311f795d7 \
-H 'x-auth-token: <AUTH_TOKEN>'
```

где: **<AUTH_TOKEN>** — ключ для доступа к API.

2.6.4.5. Проверка состояния инстанса сервиса

Чтобы проверить состояние инстанса сервиса, выполните запрос с параметрами, приведёнными в таблице 24.

Таблица 24 — Параметры запроса на проверку состояния инстанса сервиса

Параметр	Значение
Метод запроса	GET
Путь запроса	<p>https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/hoе/object/<OBJECT_UUID></p> <p>где:</p> <ul style="list-style-type: none"> <CLOUD_HOST> — доменное имя облачной платформы. <OBJECT_UUID> — идентификатор разворачивания инстанса сервиса. Значение соответствует параметру uuid в запросе на разворачивание инстанса сервиса
x-auth-token	Ключ для доступа к API

HTTP-коды ответа приведены в таблице 25.

Таблица 25 — HTTP-коды ответа

Код	Описание
200	Запрос выполнен
401	Ошибка авторизации
404	Идентификатор инстанса сервиса не найден
500	Ошибка выполнения запроса

В ответе на запрос передаётся текущий статус инстанса сервиса. Возможные статусы:

- **applying** — манифест выполняется, ресурсы в процессе создания.
- **running** — манифест выполнен, инстанс сервиса создан.
- **failed** — манифест завершен с ошибкой, инстанс сервиса не создан.
- **deleted** — инстанс сервиса удален.

Пример запроса на проверку состояния инстанса сервиса

```
$ curl
https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/hoe/object/675f6f08-2344-4cf4-a7f4-f02311f795d7 \
-H 'x-auth-token: <AUTH_TOKEN>'
```

где: **<AUTH_TOKEN>** — ключ для доступа к API.

Пример ответа на запрос, выполненный после удаления инстанса сервиса

```
{
  "uuid": "675f6f08-2344-4cf4-a7f4-f02311f795d7",
  "target_status": "deleted",
  "vars": {},
  "out": "{}",
  "status": "deleted",
  "conf_name": "user@vk.team",
  "conf_hash": "75587bae82f2492ea8a94b8b067c9898",
  "pid": "b66dde3d4d0e415aaf3412e17e53259c",
  "create_at": "2023-04-26T13:57:54.849565Z",
  "update_at": "2023-04-26T14:02:42.667399Z",
  "full_deployed": false,
  "attempts": 0,
  "max_attempts": 15
}
```

где:

- **uuid** — идентификатор разворачивания инстанса сервиса.

- `target_status` — целевой статус инстанса сервиса (`deleted` или `running`).
- `vars` — входные переменные манифеста.
- `out` — выходные параметры манифеста.
- `status` — текущий статус инстанса сервиса.
- `conf_name` — имя пользователя, развернувшего сервис.
- `conf_hash` — хеш конфигурации инстанса сервиса.
- `pid` — OpenStack PID проекта пользователя, развернувшего сервис.
- `create_at` — дата и время, когда инстанс сервиса был создан.
- `update_at` — дата и время последнего обновления инстанса сервиса системой развёртывания.
- `full_deployed` — успешно ли развернут инстанс сервиса (для статуса `deleted` значение равно `false`).
- `attempts` — количество выполненных повторных попыток при разворачивании или автовосстановлении инстанса сервиса.
- `max_attempts` — максимальное количество повторных попыток.

2.6.5. Загрузка сервисного пакета в Магазин

Загрузка сервисного пакета в Магазин осуществляется с помощью системы развёртывания через сервис Infra API. Схема загрузки приведена на рисунке 43.

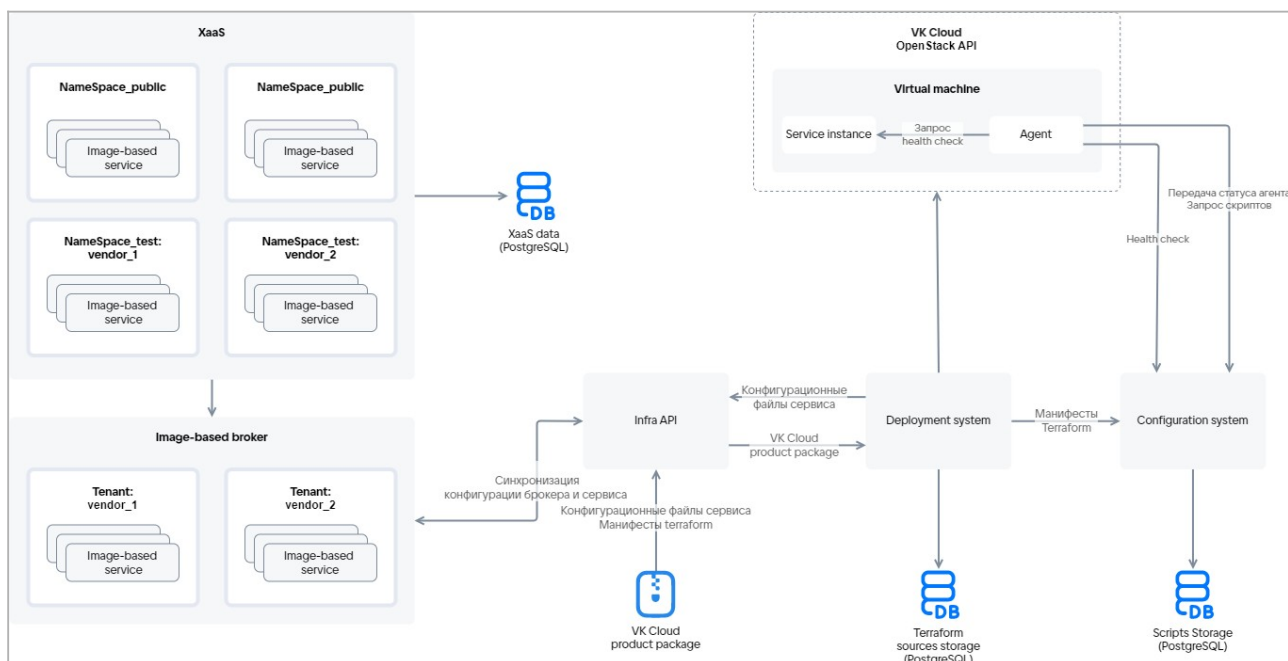


Рисунок 43 — Схема загрузки сервисного пакета

Описание основных элементов схемы приведено в таблице 26.

Таблица 26 — Описание основных элементов схемы

Имя	Описание
Infra API	Является точкой интеграции Магазина и облачной платформы для разворачивания image-based сервисов. Infra API проверяет данные пользователя, загружающего сервисный пакет или разворачивающего инстанс сервиса. После успешной проверки Infra API получает доступ к межсервисному взаимодействию для создания ресурсов, описанных в манифесте Terraform. Infra API взаимодействует с системой развёртывания и сервисом управления конфигурациями
Система развёртывания (Deployment system)	Обеспечивает разворачивание image-based сервиса на VM в облачной платформе. Управляет инфраструктурой и ПО сервиса
Image-based брокер	Управляет жизненным циклом инстанса сервиса, включает в себя tenants. Tenants объединяют image-based сервисы одного поставщика (vendor)
Сервис управления конфигурациями (Configuration system)	Сервер, на котором хранится: <ul style="list-style-type: none"> • Конфигурация, описанная в манифестах Terraform. • Результаты выполнения скриптов. • Текущая конфигурация инстанса сервиса. Предоставляет доступ к актуальной версии агента, устанавливаемой на VM
Агент (Agent)	Программное обеспечение, устанавливаемое на VM в процессе разворачивания сервиса. Выполняет скрипты и передает результаты их выполнения в сервис управления конфигурациями. Передает текущее состояние инстанса сервиса в сервис управления

Имя	Описание
	конфигурациями, чтобы обеспечить мониторинг и автовосстановление инстанса сервиса (подробнее — в подразделе Мониторинг и автовосстановление ВМ с инстансом сервиса)
Сервисный пакет (VK Cloud product package)	Структурированный набор YAML-файлов и манифестов Terraform

Тестовые (`Namespace_test`) и открытые (`Namespace_public`) пространства имён Магазина для каждого сервиса заданы в сервисном ключе.

Чтобы загрузить сервисный пакет в Магазин:

1. Выполните [подготовительные операции](#).
2. Протестируйте манифесты Terraform локально (подробнее — в подразделе [Локальное тестирование манифеста Terraform](#)).
3. Протестируйте манифесты Terraform с системой развёртывания (подробнее — в подразделе [Тестирование манифестов с системой развёртывания](#)).
4. Убедитесь, что файлы сервисного пакета заполнены и их структура соответствует используемой версии генератора JSON-файла (подробнее — в подразделе [Структура сервисного пакета](#)).
5. Убедитесь, что сочетания ID и ревизии каждого тарифного плана уникальны в рамках сервисного пакета.
6. Запакуйте сервисный пакет (директория `<SERVICE_NAME>` с конфигурационными файлами сервиса и манифестами Terraform) в zip-архив. Размер архива должен быть не более 30 МБ.
7. Чтобы загрузить архив в Магазин, выполните запрос к сервису Infra API со следующими параметрами:

- Метод запроса: `POST`.
- Путь запроса: `https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/product`

где:

- `<CLOUD_HOST>` — доменное имя облачной платформы (например, `https://mcs.mail.ru`).
- Тело запроса: zip-архив.
- `x-service-token`: сервисный ключ.

HTTP-коды ответа:

- 204 — сервисный пакет загружен.
- 400, 404, 500 — ошибка выполнения запроса.
- 401 — ошибка авторизации.

Конфигурационные файлы, описывающие тарифные планы и опции сервиса, передаются image-based брокеру, манифесты Terraform — системе развёртывания.


Сервисный пакет, загружаемый в Магазин, сначала попадает в тестовое пространство имён, после публикации — в открытое.

Пример запроса на загрузку сервисного пакета в Магазин

```
$ curl -v -X POST
https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/product \
-H 'x-service-token: <SERVICE_TOKEN>' \
-F "upload=@/home/VKservice.zip"
```

где: **<SERVICE_TOKEN>** — сервисный ключ.


- После окончания загрузки сервиса зайдите в ЛК облачной платформы и убедитесь, что сервис отображается в Магазине. Загруженный сервис будет доступен только пользователям тестовых пространств имён Магазина, указанных в сервисном ключе.

 Если после загрузки сервис не отображается в Магазине, то выйдите из ЛК и войдите заново.

- Протестируйте сервис в тестовом пространстве имён Магазина:


- Зайдите в ЛК облачной платформы.
- Проверьте, что в ЛК мастер конфигурации каждого тарифного плана отображается корректно.
- Проверьте, что в ЛК сервис можно установить. Убедитесь, что разворачивание инстанса сервиса выполнено успешно.

Если при разворачивании инстанса сервиса не удалось создать ресурс, описанный в манифесте, то система развёртывания запустит процесс разворачивания сервиса повторно. Процесс разворачивания инстанса сервиса может занимать до 1,5 ч.

 При каждой новой попытке системы развёртывания установить сервис все существующие ресурсы удаляются и создаются заново.

г. Проверьте, что в ЛК установленную конфигурацию инстанса сервиса можно обновить.

Изменение параметров ресурсов, описанных в манифесте, запускает процесс переустановки сервиса, при котором обновляются только измененные ресурсы. Если система развёртывания не смогла обновить ресурсы, то она будет пробовать обновить их повторно. Процесс обновления конфигурации инстанса сервиса с учетом повторных попыток может занимать до 1,5 ч.

 При обновлении конфигурации инстанса сервиса система развёртывания обновляет только измененные ресурсы.

д. Проверьте, что в ЛК установленный сервис можно удалить.

е. При необходимости внесите изменения в конфигурацию сервиса (подробнее — в разделе [Обновление image-based сервиса](#)).


2.6.6. Публикация образа сервиса

Чтобы образ сервиса сделать публичным:

1. Отправьте электронное письмо на marketplace@mcs.mail.ru.

В письме укажите ID образа сервиса.

Указанный образ будет сделан публичным. В ответном письме будет выслан ID публичного образа сервиса.

 Публичный образ сервиса будет доступен всем пользователям облачной платформы.

2. В манифестах `plans/<PLAN_NAME>/deployment/deploy.tf` поменяйте ID образа сервиса на ID публичного образа сервиса.

3. В файле `service.yaml` укажите новую ревизию сервиса.

4. Чтобы загрузить в Магазин сервисный пакет с ID публичного образа сервиса, выполните запрос к сервису Infra API, приведённый в подразделе [Загрузка сервисного пакета в Магазин](#).

2.6.7. Публикация сервиса

Потребителям сервис будет доступен только после публикации.

Чтобы опубликовать image-based сервис в Магазине, отправьте электронное письмо на marketplace@mcs.mail.ru. В письме укажите ID сервиса и его ревизию.

Будет проведено модерирование сервисного пакета. После этого сервис будет опубликован в Магазине.

Опубликованная ревизия сервиса будет доступна в открытых пространствах имён Магазина, указанных в сервисном ключе. Сервис будет доступен потребителям.




Потребителям доступна только последняя опубликованная ревизия сервиса.

3. Обновление image-based сервиса

Обновление конфигурации image-based сервиса может включать:


- Изменение параметров тарифных планов и опций.
- Добавление тарифных планов и опций.
- Удаление тарифных планов и опций.
- Изменение параметров ресурсов, описанных в манифестах Terraform.
- Добавление ресурсов в манифесты Terraform.

 Текущая версия Магазина не поддерживает удаление ресурсов из конфигурации манифестов Terraform. Функциональность будет реализована в следующей версии.

Все изменения в манифестах должны поддерживать обратную совместимость.


Чтобы обновить image-based сервис в Магазине:

1. Отредактируйте файлы сервисного пакета. Параметры файлов приведены в разделе [Добавление image-based сервиса в Магазин](#).
2. В файле `service.yaml` укажите новую ревизию сервиса.

 Сервис не будет обновлён, если указанная ревизия сервиса уже загружена в Магазин.

3. Если были изменены файлы тарифных опций (директория `parameters`), то для каждого тарифного плана, использующего хотя бы одну измененную опцию, укажите новую ревизию.

Если были изменены файлы, описывающие тарифный план, то в файле `plans/<PLAN_NAME>/plan.yaml` укажите новую ревизию.

 Тарифные планы и опции не будут обновлены, если указанные ревизии планов уже загружены в Магазин.

4. Загрузите обновлённый сервисный пакет и опубликуйте новую ревизию сервиса в открытом пространстве имён Магазина (подробнее — в подразделе [Загрузка image-based сервиса в Магазин](#)).

Предыдущая опубликованная ревизия переходит в тестовое пространство имён Магазина. В тестовых пространствах имён сохраняются все ревизии сервиса.

Публикация новой ревизии не влияет на уже существующие инстансы сервиса. Ревизия этих инстансов сервиса не изменяется.

4. Перенос image-based сервиса в архив

В архив переносится конкретная ревизия image-based сервиса.

Сервис, перенесённый в архив, не будет доступен в Магазине (ни в тестовых, ни в открытых пространствах имён). Сервис нельзя подключить.

Инстансы сервиса, созданные до переноса сервиса в архив, остаются доступными пользователю, но их нельзя обновить (изменить тарифный план или опции).

Чтобы перенести сервис в архив, выполните запрос с параметрами, приведёнными в таблице 27.

Таблица 27 — Параметры запроса на архивацию сервиса

Параметр	Значение
Метод запроса	POST
Путь запроса	<code>https://<CLOUD_HOST>/marketplace/api/infra-api/api/v1-public/product/archive</code> где: <CLOUD_HOST> — доменное имя облачной платформы (например, <code>https://mcs.mail.ru</code>)
Тело запроса	<p>В теле запроса передаётся список сервисов.</p> <p>Для каждого сервиса укажите следующие параметры:</p> <ul style="list-style-type: none"> <code>id</code> — ID сервиса в формате UUID4. <code>revision</code> — ревизия сервиса. <p>Пример тела запроса в JSON-формате</p> <pre>{ "services": [{ "id": "b27ee400-b045-43eb-8a7d-db48c280ba16", "revision": "v.1.1" }] }</pre>
<code>x-service-token</code>	Сервисный ключ

HTTP-коды ответа приведены в таблице 28.

Таблица 28 — HTTP-коды ответа

Код	Описание
204	Сервис перенесён в архив
400, 500	Ошибка выполнения запроса

Код	Описание
401	Ошибка авторизации
404	Сервис не найден

Пример запроса на архивацию сервиса

```
$ curl -v -X POST https://mcs.mail.ru/marketplace/api/infra-api/api/v1-public/product/archive \
-H "Content-Type: application/json" \
-H "x-service-token: $token" \
--data "{
  \"services\": [
    {
      \"id\": \"b27ee400-b045-43eb-8a7d-db48c280ba16\",
      \"revision\": \"v.1.1\"
    }
  ]
}"
```

5. Добавление SaaS-сервиса в Магазин

5.1. Порядок действий

Чтобы добавить SaaS-сервис в Магазин:

1. Разработайте брокер для сервиса.
2. Опишите конфигурацию сервиса (тарифные планы, опции).
3. Загрузите конфигурацию сервиса в брокер.
4. Загрузите и опубликуйте сервис в Магазине.

5.2. Брокер для SaaS-сервиса

Магазин взаимодействует с сервисом с помощью брокера.

Разработайте брокер для SaaS-сервиса по протоколу VK OSB. Чтобы получить протокол VK OSB, отправьте электронное письмо на marketplace@mcs.mail.ru. Брокер должен реализовывать методы, описывающие жизненный цикл инстансов сервиса (рисунок 44).

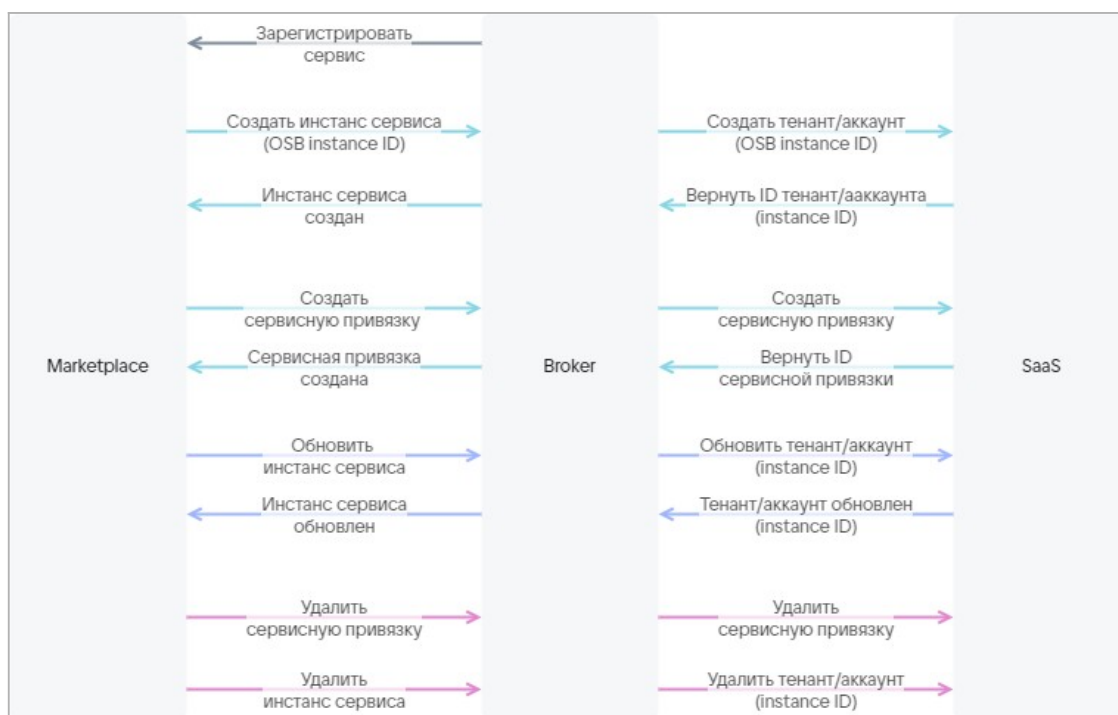


Рисунок 44 — Взаимодействие брокера с Магазином и SaaS-сервисом

Брокер можно разработать на основе шаблона, в котором уже реализовано взаимодействие между Магазином и брокером. Шаблон необходимо доработать описанием взаимодействия брокера с конкретным SaaS-сервисом.

Чтобы разработать брокер на основе шаблона:

1. Отправьте электронное письмо на marketplace@mcs.mail.ru, чтобы получить шаблон брокера. Шаблон разработан на языке Python на основе фреймворка [FastAPI](#).
2. Разработайте клиент для взаимодействия с API сервиса.
3. Разработайте сервисный менеджер, описывающий взаимодействие брокера и сервиса:
 - а. Реализуйте метод для создания тенанта/аккаунта, принимающий на вход:
 - OSB ID инстанса сервиса, который формируется Магазином при подключении сервиса. OSB ID передается в SaaS-сервис, чтобы SaaS-сервис сформировал ID инстанса сервиса. ID инстанса сервиса используется в методах, описанных ниже.
 - ID тарифного плана.
 - Тарифные опции плана, описанные в секции `plans.schemas.service_instance.create` в JSON-файле.
 - Контекст (опционально). Контекст содержит информацию о пользователе (email, OpenStack PID облачного проекта пользователя).
 - б. Реализуйте метод для обновления тенанта/аккаунта, принимающий на вход:
 - ID инстанса сервиса.
 - ID тарифного плана.
 - Тарифные опции плана, описанные в секции `plans.schemas.service_instance.update` в JSON-файле.
 - в. Реализуйте методы для получения и удаления тенанта/аккаунта, принимающие на вход ID инстанса сервиса.
 - г. Реализуйте метод для создания сервисных привязок, принимающий на вход:
 - ID инстанса сервиса.
 - Параметры сервисных привязок, описанные в секции `plans.schemas.service_binding.create` в JSON-файле.
 - Контекст (опционально). Контекст содержит информацию о пользователе (email, OpenStack PID облачного проекта пользователя).
 - д. Реализуйте методы для получения и удаления сервисных привязок, принимающие на вход:
 - ID инстанса сервиса.

- ID сервисной привязки.

4. Интегрируйте клиент в сервисный менеджер.

Если в SaaS-сервисе есть постоплатные тарифные опции (подробнее — в подразделе [Стоимость тарифных планов и опций](#)), то выполните дополнительные действия:

1. В сервисном менеджере реализуйте метод для получения отчёта по фактически использованным ресурсам SaaS-сервиса. В ответе на запрос брокер должен передать Магазину отчёт.

Пример отчёта о фактически использованных ресурсах SaaS-сервиса

```
{
  "batch_id": 35,
  "data": [
    {
      "kind": "vms",
      "type": "cb",
      "unit": "month",
      "price": 600.0,
      "value": 1.0,
      "plan_uuid": "2f070fe3-3e31-4482-bad4-a4d0c36bab31",
      "instance_uuid": "d1dff97c-df6c-45a7-95e1-b539a8d77721"
    },
    {
      "kind": "storage",
      "type": "cb",
      "unit": "GB-month",
      "price": 3.3,
      "value": 2.4474525451660156,
      "plan_uuid": "2f070fe3-3e31-4482-bad4-a4d0c36bab31",
      "instance_uuid": "d1dff97c-df6c-45a7-95e1-b539a8d77721"
    },
    {
      "kind": "vms",
      "type": "cb",
      "unit": "month",
      "price": 600.0,
      "value": 0.0,
      "plan_uuid": "354df2fa-5ec3-45e1-b99b-7d45840cf3df",
      "instance_uuid": "3910e993-63f0-4a90-95d7-5eea35785c29"
    },
    {
      "kind": "storage",
      "type": "cb",
      "unit": "GB-month",
      "price": 3.3,
      "value": 0.0,
      "plan_uuid": "354df2fa-5ec3-45e1-b99b-7d45840cf3df",
      "instance_uuid": "3910e993-63f0-4a90-95d7-5eea35785c29"
    },
    {
      "kind": "storage",
      "type": "cb",
      "unit": "GB-month",
```

```

    "price": 7.0,
    "value": 0.0,
    "plan_uuid": "d719f348-3497-4341-8465-a438bfcc2d96",
    "instance_uuid": "f10e67be-dfd7-4dcd-917c-4bfd3cd64ce2"
  }
]
}

```

где:

- **batch_id** — ID отчета, сформированный брокером.
- **kind** — имя тарифной опции. Должно соответствовать имени, используемому в файле с конфигурацией сервиса.
- **type** — имя сервиса. Должно соответствовать **SERVICE_NAME**, указанному в имени файла с конфигурацией сервиса (**catalog_<SERVICE_NAME>.json**).
- **unit** — единица измерения опции.
- **price** — стоимость единицы опции в месяц. Должно соответствовать значению **plans.billing.options.cost** конкретной опции, указанному в файле с конфигурацией сервиса.
- **value** — фактическое использованное количество опции за месяц, сформированное SaaS-сервисом.
- **plan_uuid** — ID тарифного плана.
- **instance_uuid** — ID инстанса (тенанта/аккаунта) сервиса.

2. В сервисном менеджере реализуйте метод для сообщения брокеру о том, что отчёт обработан (списание денежных средств по отчёту произошло или запланировано).

5.3. Конфигурация SaaS-сервиса

Опишите конфигурацию SaaS-сервиса для брокера. Если SaaS-брокер создан по шаблону, то выполните действия:

1. Создайте файл **catalog_<SERVICE_NAME>.json**.
2. В JSON-файле опишите конфигурацию сервиса по следующей структуре:

```

{
  "services": [
    {
      <SERVICE_PARAMETERS>,
      "preview": {

```



```

    "parameters": [
    ],
  },
  "plans": [
  ]
}
]
}

```

где:

- **<SERVICE_PARAMETERS>** — параметры сервиса (подробнее — в подразделе [Параметры сервиса](#)).
- Секция **preview** — описывает тарифные опции для матрицы тарифных планов (подробнее — в подразделе [Секция preview](#)).
- Секция **plans** — описывает тарифные планы и их опции (подробнее — в подразделе [Секция plans](#)).

Пример JSON-файла приведён в приложении [Пример JSON-файла для SaaS-сервиса](#).

5.3.1. Параметры сервиса

В JSON-файле укажите параметры сервиса, приведённые в таблице 29.

Таблица 29 — Параметры сервиса

Имя	Описание	Формат	Обязательный
id	Идентификатор сервиса UUID4 (ID), сформированный с помощью генератора UUID4	string (UUID4)	Да
revision	Ревизия сервиса. Сочетание ревизии и ID сервиса определяет его уникальность в Магазине. Остальные параметры описывают характеристики конкретной ревизии сервиса	string, до 255 символов	Да
name	Название сервиса	string, до 255 символов	Да
short_description	Краткое описание сервиса, которое будет отображаться в его карточке в Магазине	string, до 255 символов	Да
full_description	Полное описание сервиса, которое будет отображаться при просмотре информации о нём	string. Поддерживает формат Markdown	Нет
singleton	Определяет, есть ли ограничение в один инстанс сервиса на один проект облачной платформы	boolean	Нет
auto_bind	Определяет, нужно ли после	boolean	Нет

Имя	Описание	Формат	Обязательный
	разворачивания сервиса автоматически создавать сервисную привязку		
icon	URL иконки сервиса	string, до 512 символов	Нет
help	URL документации сервиса	string, до 512 символов	Нет
bindable	Определяет, можно ли создавать сервисные привязки для этого сервиса	boolean	Да
plan_updateable	Определяет, может ли пользователь переходить с одного тарифного плана на другой без удаления сервиса. Значение параметра применяется для всех планов сервиса. Значение можно переопределить для конкретного плана (подробнее в подразделе — Параметры тарифного плана)	boolean	Да
deactivatable	Определяет, можно ли временно приостановить использование сервиса	boolean	Да
bindings_retrievable	Определяет, нужно ли повторять попытку создания сервисной привязки в течение определенного времени, если предыдущая попытка не удалась	boolean	Да
instances_retrievable	Определяет, нужно ли повторять попытку создания инстанса сервиса в течение определенного времени, если предыдущая попытка не удалась	boolean	Да



Сочетание ID и ревизии сервиса должно быть уникальным в рамках Магазина. Если сервис с такими же идентификатором и ревизией уже существует в Магазине, то конфигурация сервиса не будет обновлена.

5.3.2. Секция preview

Секция **preview** определяет, какие тарифные опции будут отображаться в матрице тарифных планов (подробнее — в подразделе [Матрица тарифных планов](#)).


В матрице будут отображаться все тарифные планы, указанные в секции **plans**, и тарифные опции, указанные в секции **preview**.

В секции **preview** перечислите тарифные опции по следующей структуре:

```
"preview": {
  "parameters": [
    {
      "name": "<OPTION>"
    },
    ...
  ]
}
```

где:

- Секция **parameters** — определяет тарифные опции для матрицы тарифных планов. Может быть пустой.
- **<OPTION>** — имя тарифной опции в JSON-файле.

 В матрице тарифные опции будут отображаться с именами, заданными в параметре **description** этих опций.

Пример заполнения секции **preview**

```
"preview": {
  "parameters": [
    {
      "name": "vms"
    },
    {
      "name": "servers"
    },
    {
      "name": "storage"
    }
  ]
}
```

5.3.3. Секция plans

В секции **plans** опишите тарифные планы сервиса по следующей структуре:

```
"plans": [
  {
    <PLAN_PARAMETERS>,
    "display": {
    },
    "billing": {
    },
    "schemas": {
    }
  },
  ...
]
```

```
    ...  
  ]
```

где:


- `<PLAN_PARAMETERS>` — параметры плана (подробнее — в подразделе [Параметры тарифного плана](#)).
- Секция `display` — описывает мастер конфигурации конкретного тарифного плана (подробнее — в подразделе [Секция display](#)).
- Секция `billing` — описывает стоимость плана и его опций (подробнее — в подразделе [Секция billing](#)).
- Секция `schemas` — описывает тарифные опции плана (подробнее — в подразделе [Секция schemas](#)).


5.3.3.1. Параметры тарифного плана

Для тарифного плана укажите параметры, приведённые в таблице 30.

Таблица 30 — Параметры тарифного плана

Имя	Описание	Формат	Обязательный
id	Идентификатор тарифного плана UUID4 (ID), сформированный с помощью генератора UUID4	string (UUID4)	Да
revision	Ревизия тарифного плана. Сочетание ревизии и ID тарифного плана определяет его уникальность в сервисе. Остальные параметры описывают характеристики конкретной ревизии тарифного плана	string, до 255 символов	Да
name	Техническое название тарифного плана, которое не отображается в интерфейсе Магазина. Должно быть указано латинскими буквами с использованием знака нижнего подчеркивания вместо пробелов	string, до 255 символов	Да
description	Название тарифного плана, которое отображается в интерфейсе Магазина	string, до 255 символов	Да
free	Определяет, бесплатный тарифный план или нет	boolean	Да
plan_updateable	Определяет, может ли пользователь переходить с одного тарифного плана на другой без удаления сервиса. Переопределяет значение, заданное в одноимённом параметре сервиса	boolean	Нет
metadata	Определяет тестовые и открытые пространства имён Магазина, в которых тарифный план	map, ключи — string	Нет

Имя	Описание	Формат	Обязательный
	<p>будет доступен.</p> <p>Тестовые пространства имён задаются в ключе <code>test_ns</code>.</p> <p>Открытые пространства имён задаются в ключе <code>prod_ns</code>.</p> <p>Если пространства имён не заданы, то будут использованы значения по умолчанию:</p> <ul style="list-style-type: none"> • Тестовое — <code>test</code>. • Открытое — <code>vkcs_ru</code>. <div>  Если в ключе <code>test_ns</code> указано тестовое пространство имён, которого еще нет в Магазине, то оно будет создано автоматически. Чтобы получить доступ к этому пространству имён, отправьте электронное письмо на marketplace@mcs.mail.ru. В письме укажите тестовое пространство имён и email пользователя. </div>		

 Сочетание ID и ревизии тарифного плана должно быть уникальным в рамках сервиса. Если план с такими же идентификатором и ревизией уже существует в этом сервисе, то тарифный план не будет обновлен.

5.3.3.2. Секция `display`

В секции `display` опишите мастер конфигурации тарифного плана (подробнее — в подразделе [Мастер конфигурации тарифного плана](#)) по следующей структуре:

```
"display": {
  "pages": [
    {
      <PAGE_PARAMETERS>,
      "groups": [
        {
          <GROUP_PARAMETERS>,
          "parameters": [
            {
              <OPTION_PARAMETERS>
            },
            ...
          ]
        },
        ...
      ]
    },
    ...
  ],
  ...
}
```

```
]
}
```


где:

- Секция `pages` — описывает страницы мастера конфигурации тарифного плана. Может быть пустой.
- `<PAGE_PARAMETERS>` — параметры одной страницы.
- Секция `groups` — описывает группы тарифных опций в рамках одной страницы.
- `<GROUP_PARAMETERS>` — параметры группы тарифных опций.
- Секция `parameters` — определяет тарифные опции в рамках одной группы.
- `<OPTION_PARAMETERS>` — параметры тарифных опций.

В секции `display` описываются все страницы мастера конфигурации тарифного плана, кроме первой и последней. Максимальное количество страниц — 5.

Параметры страниц, групп и тарифных опций в группах одинаковые и приведены в таблице 31.

Таблица 31 — Параметры страниц, групп и тарифных опций для мастера конфигурации тарифного плана

Имя	Описание	Формат	Обязательный
name	Имя страницы, группы или тарифной опции в JSON-файле. <div>  В интерфейсе Магазина тарифные опции будут отображаться с именами, заданными в параметре <code>description</code> этих опций (секция <code>plans.schemas</code>). </div>	string. Имя страницы — до 32 символов. Имя группы — до 255 символов	Да
index	Порядковый номер страницы, группы на странице или тарифной опции в группе	integer	Нет

Мастер конфигурации тарифного плана, приведённый на рисунке 45, соответствует следующему содержимому секции `display`:

```
"display": {
  "pages": [
    {
      "name": "Настройки", // Имя страницы
      "index": 0,
```

```
"groups": [
  {
    "name": "", // Имя группы
    "index": 0,
    "parameters": [
      {
        "name": "api_requests_daily_limit" // Имя тарифной опции в JSON-файле
        "index": 0,
      },
      {
        "name": "groups"
        "index": 1,
      },
      {
        "name": "products"
        "index": 2,
      },
      {
        "name": "repots"
        "index": 3
      }
    ]
  }
]
```

✓ Выбран тариф «Расширенный»

2 Настройки

3 Подтверждение

Настройки

Доступ к открытому API ⓘ:

Количество групп пользователей ⓘ:

Количество продуктов ⓘ:

☒ Выгрузка отчётов в CSV ⓘ

[Назад](#) [Следующий шаг](#)

Детализация

Стоимость указана за 1 месяц использования

Тариф «Расширенный»	10 000 ₽
Количество групп пользователей + 5	200 ₽
Спишется	10 200 ₽

Рисунок 45 — Мастер конфигурации тарифного плана

5.3.3.3. Секция billing

Секция **billing** описывает:

- Стоимость конкретного тарифного плана.

Для стоимости тарифного плана поддерживается только предоплатный способ списания денежных средств (подробнее — в подразделе [Стоимость тарифных планов и опций](#)).

- Стоимость тарифных опций плана.

На текущий момент платными тарифными опциями могут быть только числовые опции.

Для стоимости тарифных опций поддерживаются постоплатный и предоплатный способы списания денежных средств. Стоимость описывается одними и теми же параметрами независимо от способа. Применяемый способ оплаты определяется местом описания тарифных опций в секции `plans.schemas`:

- Если опции описаны в секциях `service_instance.create` и `service_instance.update`, то применяется предоплатный способ списания.
- Если опции описаны в секции `service_instance.resource_usages`, то применяется постоплатный способ списания.



В рамках одного тарифного плана могут быть опции только с одним способом списания денежных средств.

- Пользовательский шаг изменения для тарифной опции типа `integer`.

Опишите секцию `billing` по следующей структуре:

```
"billing": {
  "cost": <MONTH_COST>,
  "options": {
    "<OPTION>": {
      <OPTION_BILLING>
    },
    ...
  }
}
```

где:

- Параметр `cost` — определяет стоимость плана за месяц `<MONTH_COST>` без учёта платных тарифных опций. Задаётся в валюте страны, где развёрнут Магазин. Если план бесплатный, укажите `0`.
- Секция `options` (опциональная) — описывает стоимость платных тарифных опций.
- `<OPTION>` — имя тарифной опции в JSON-файле.

- `<OPTION_BILLING>` — параметры стоимости и шага изменения тарифной опции. Сама опция (тип, настройки значения) описывается в секции `schemas`.

5.3.3.3.1. Секция `billing` с бесплатной тарифной опцией типа `integer` с пользовательским шагом изменения

Шаг изменения тарифной опции типа `integer` в `<OPTION_BILLING>` описывается такими же параметрами, как и для `image-based` сервиса (подробнее — в подразделе [Секция `billing`](#)).

Чтобы опция была бесплатной, укажите `0` в параметре `billing.options.<OPTION>.cost`.

Пример описания секции `billing` для плана с бесплатной тарифной опцией типа `integer` с шагом изменения

```
"billing": {
  "cost": 2000,
  "options": {
    "quantity": {
      "base": 25,
      "cost": 0,
      "unit": {
        "size": 100
      }
    }
  }
}
```

На рисунках 46, 47, 48 приведено, как будет отображаться в мастере конфигурации тарифного плана стоимость тарифного плана и опция, описанные выше.

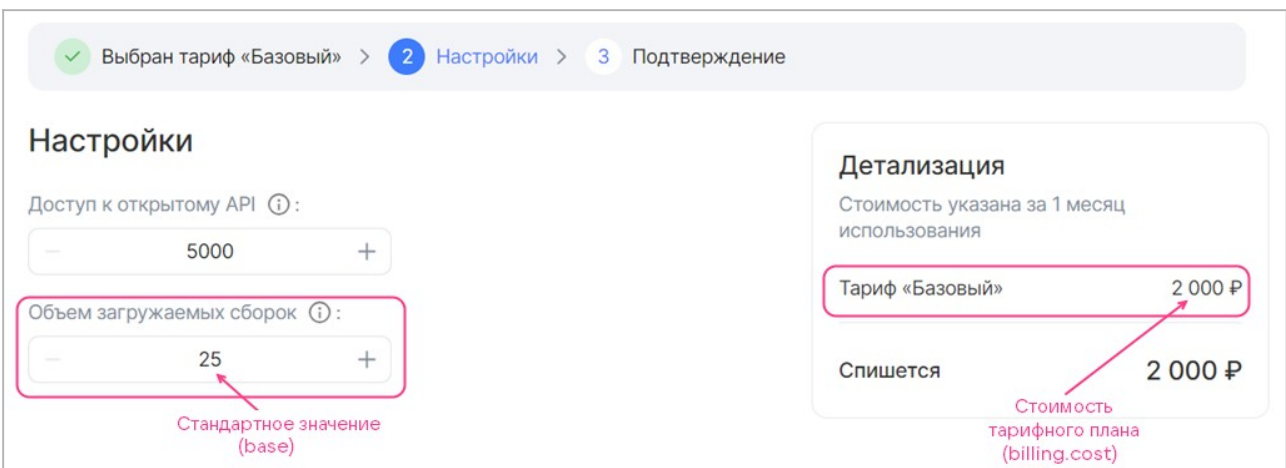


Рисунок 46 — Тарифный план с бесплатной опцией типа `integer` с шагом изменения (`base = 25`, `size = 100`)

✓

Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ:

—

5000

+

Объем загружаемых сборок ⓘ:

—

125

+

Стандартное значение увеличено на 1 шаг

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый»	2 000 ₽
Спишется	2 000 ₽

Рисунок 47 — Тарифный план с бесплатной опцией типа `integer` с шагом изменения, значение опции увеличено на 1 шаг (base = 25, size = 100)

✓

Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ:

—

5000

+

Объем загружаемых сборок ⓘ:

—

225

+

Стандартное значение увеличено на 2 шага

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый»	2 000 ₽
Спишется	2 000 ₽

Рисунок 48 — Тарифный план с бесплатной опцией типа `integer` с шагом изменения, значение опции увеличено на 2 шага (base = 25, size = 100)

5.3.3.3.2. Секция `billing` с предоплатной тарифной опцией типа `integer` с шагом изменения

Стоимость и шаг изменения предоплатной тарифной опции типа `integer` в `<OPTION_BILLING>` описывается такими же параметрами, как и для `image-based` сервиса (подробнее — в подразделе [Секция `billing`](#)).

Чтобы опция была предоплатной, укажите стоимость за 1 шаг изменения в параметре `billing.options.<OPTION>.cost`.

Пример описания секции `billing` для плана с предоплатной тарифной опцией типа `integer` с шагом изменения

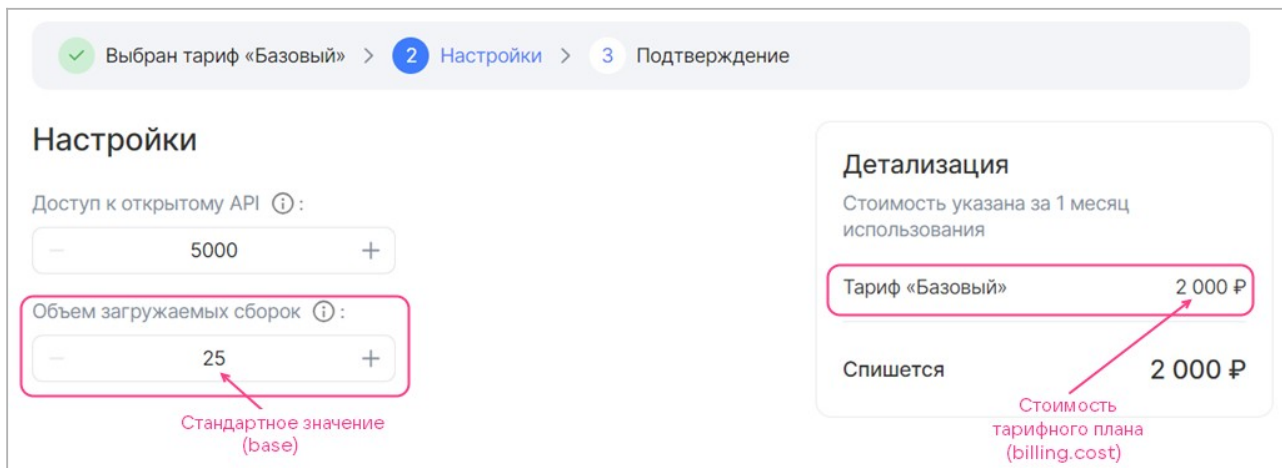
```
"billing": {
  "cost": 2000,
  "options": {
    "quantity": {
      "base": 25,
```

```

    "cost": 150,
    "unit": {
      "size": 100
    }
  }
}
}

```

На рисунках 49, 50, 51 приведено, как будет отображаться в мастере конфигурации тарифного плана стоимость тарифного плана и опция, описанные выше.



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ: 5000

Объем загружаемых сборок ⓘ: 25

Стандартное значение (base)

Детализация

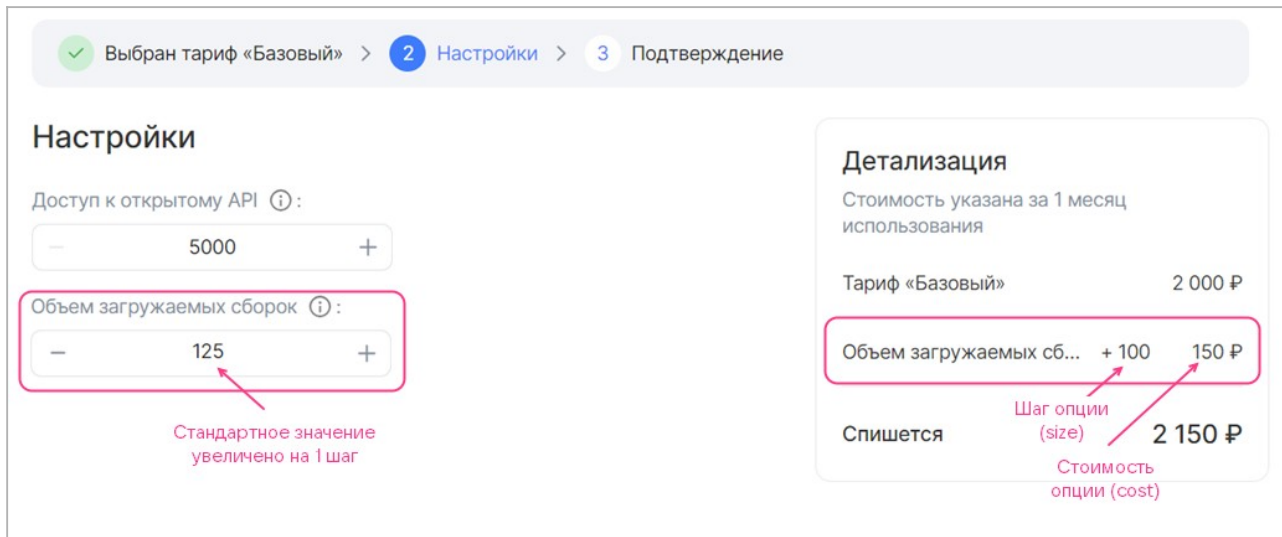
Стоимость указана за 1 месяц использования

Тариф «Базовый» 2 000 ₽

Спишется 2 000 ₽

Стоимость тарифного плана (billing.cost)

Рисунок 49 — Тарифный план с предоплатной опцией типа **integer** с шагом изменения (base = 25, size = 100)



Выбран тариф «Базовый» > 2 Настройки > 3 Подтверждение

Настройки

Доступ к открытому API ⓘ: 5000

Объем загружаемых сборок ⓘ: 125

Стандартное значение увеличено на 1 шаг

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый» 2 000 ₽

Объем загружаемых сб... + 100 150 ₽

Шаг опции (size)

Спишется 2 150 ₽

Стоимость опции (cost)

Рисунок 50 — Тарифный план с предоплатной опцией типа **integer** с шагом изменения, значение опции увеличено на 1 шаг (base = 25, size = 100)

✓

Выбран тариф «Базовый» >

2

Настройки >

3

Подтверждение

Настройки

Доступ к открытому API ⓘ :

–

5000

+

Объем загружаемых сборок ⓘ :

–

225

+

Стандартное значение увеличено на 2 шага

Детализация

Стоимость указана за 1 месяц использования

Тариф «Базовый» 2 000 ₽

Объем загружаемых сб...

+ 200

300 ₽

Спишется 2 шага 2 150 ₽


Стоимость опции за 2 шага


Рисунок 51 — Тарифный план с предоплатной опцией типа `integer` с шагом изменения, значение опции увеличено на 2 шага (base = 25, size = 100)

5.3.3.3. Секция `billing` с постоплатной числовой тарифной опцией

Стоимость постоплатной опции типа `integer` или `number` в `<OPTION_BILLING>` описывается параметрами, приведёнными в таблице 32.

Таблица 32 — Параметры числовой постоплатной тарифной опции

Имя	Описание	Формат	Обязательный
cost	<p>Определяет стоимость единицы опции.</p> <p> Стоимость должна соответствовать значению <code>price</code>, указанному в методе брокера для получения отчёта по фактически использованным ресурсам SaaS-сервиса (подробнее — в подразделе Брокер для SaaS-сервиса).</p>	float64, >= 0	Да
unit	Определяет единицы измерения опции	—	Да
unit.size	Шаг тарификации опции. Значение должно быть <code>1</code>	integer, > 0	Да
unit.measurement	Определяет единицы измерения опции	string, до 255 символов	Нет

 Постоплатные тарифные опции могут быть только в бесплатном тарифном плане.

Пример описания секции `billing` для плана с постоплатной тарифной опцией `storage`

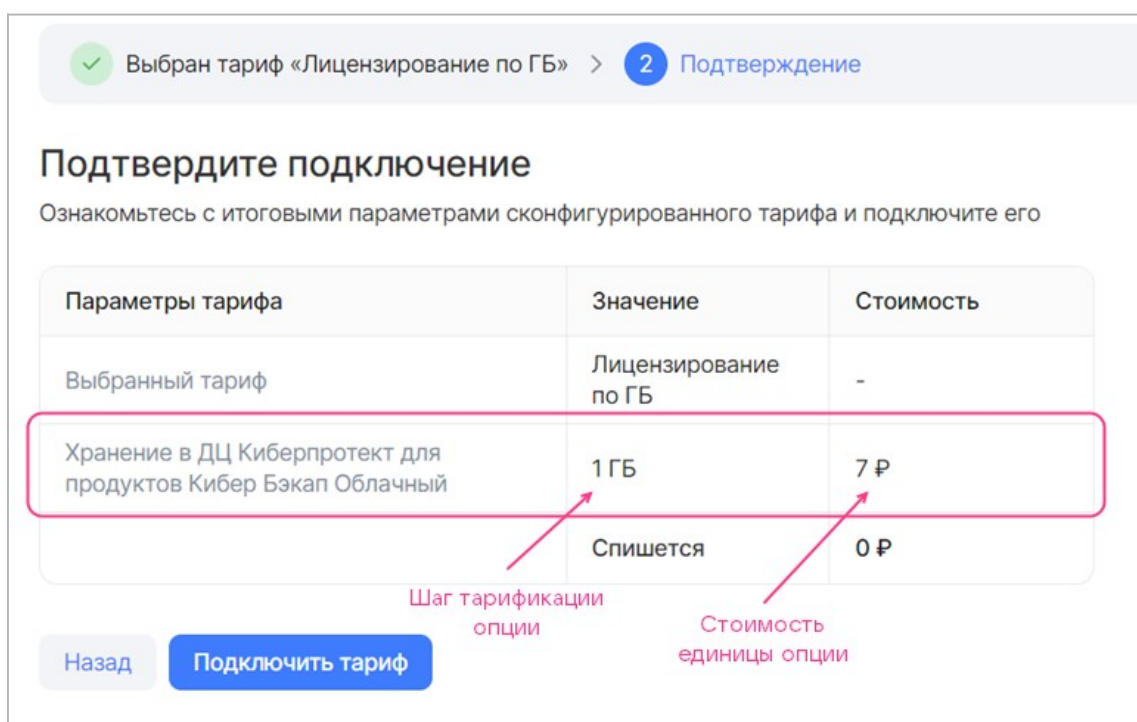
```
"billing": {
  "cost": 0,
```

```

"options": {
  "storage": {
    "cost": 7,
    "unit": {
      "size": 1,
      "measurement": "ГБ"
    }
  }
}
}
}

```

В примере выше тарифный план бесплатный, единица тарифной опции **storage** стоит 7 денежных единиц (рисунок 52).



Параметры тарифа	Значение	Стоимость
Выбранный тариф	Лицензирование по ГБ	-
Хранение в ДЦ Киберпротект для продуктов Кибер Бэкап Облачный	1 ГБ	7 Р
	Спишется	0 Р

Рисунок 52 — Тарифный план с постоплатной опцией (billing.cost = 7, billing.unit.size = 1)

5.3.3.4. Секция schemas

В секции **schemas** опишите тарифные опции конкретного плана (подробнее — в подразделе [Типы тарифных опций](#)) по следующей структуре:

```

"schemas": {
  "service_instance": {
    "create": {
      "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
        }
      }
    }
  }
}

```

```

    },
    "update": {
      "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
        }
      }
    },
    "resource_usages": {
      "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
        }
      }
    }
  },
  "service_binding": {
    "create": {
      "parameters": {
        "type": "object",
        "properties": {
        }
      }
    }
  }
}

```

где:

- Секция `service_instance` — описывает тарифные опции плана и определяет способ списания денежных средств для платных опций.
 - Секция `service_instance.create` — описывает бесплатные и предоплатные тарифные опции, которые будут активными в мастере конфигурации тарифного плана при подключении сервиса.
 - Секция `service_instance.update` — описывает бесплатные и предоплатные тарифные опции, которые будут активными в мастере конфигурации тарифного плана при обновлении тарифного плана сервиса.
 - Секция `service_instance.resource_usages` — описывает постоплатные тарифные опции.



Все опции с постоплатным способом списания денежных средств также должны быть описаны в брокере (подробнее — в подразделе [Брокер для SaaS-сервиса](#)).

- Секция `service_binding` — описывает создание сервисных привязок.



В рамках одного тарифного плана могут быть опции только с одним способом списания денежных средств. Могут быть описаны:

- Только секции `service_instance.create` и `service_instance.update` (обе или только одна).
- Или только секция `service_instance.resource_usages`.

Все секции внутри `schemas` являются обязательными для объявления в JSON-файле. Секции могут быть пустыми.

Параметры тарифных опций описываются JSON-схемами. Стоимость платных опций, а также шаг изменения для опции типа `integer` описывается в секции `plans.billing.options` (подробнее — в подразделе [Секция billing](#)).

5.3.3.4.1. Бесплатные и предоплатные тарифные опции

Бесплатные и предоплатные тарифные опции описываются в секциях `properties` такими же параметрами, как для image-based сервиса (подробнее — в подразделе [Секция schema](#)). Стоимость опции описывается в секции `billing`.

Для SaaS-сервиса поддерживаются следующие типы тарифных опций:

- `integer`.
- `string`.
- `boolean`.

Заполнение секции `schema` с примерами отображения разных типов опций в интерфейсе Магазина описано в подразделе [Заполнение YAML-файлов тарифных опций](#).

Пример описания разных типов опций в формате `JSON` приведён ниже.

Пример описания секции `schemas` для разных типов тарифных опций

```
"schemas": {
  "service_instance": {
    "create": {
      "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
          "int_const": { // Тарифная опция-константа типа integer
            "type": "integer",
            "description": "Размер системного диска",
            "hint": "В ГБ",
```


```

        "const": 20
    },
    "int_enum": { // Тарифная опция типа integer с выбором значения из
списка
        "type": "integer",
        "description": "Количество серверов в кластере",
        "enum": [3, 5, 7],
        "default": 5
    },
    "int_step_1": { // Тарифная опция типа integer с шагом изменения 1
        "type": "integer",
        "description": "Количество участников",
        "hint": "Количество сотрудников компании заказчика, которые могут
использовать инфраструктуру тестирования и обрабатывать отчёты от тестировщиков VK
Testers.",
        "default": 20,
        "minimum": 20
    },
    "int_step_user": { // Тарифная опция типа integer с
пользовательским шагом изменения. Параметры шага описываются в секции billing
        "type": "integer",
        "description": "Объем загружаемых сборок",
        "hint": "На платформу можно загружать тестовые сборки приложений
для раздачи сотрудникам заказчика и тестировщикам VK Testers. Чем больше хранилище, тем
больше версий ваших продуктов можно сохранять на платформе тестирования. Поддерживаемые
платформы: iOS, Android, Windows, MacOS, Linux.",
        "default": 0
    },
    "string_const": { // Тарифная опция-константа типа string
        "type": "string",
        "description": "Логин администратора",
        "const": "admin@example.ru"
    },
    "string_input": { // Тарифная опция типа string с вводом значения
        "type": "string",
        "description": "Email администратора",
        "hint": "Email для выпуска SSL-сертификата"
    },
    "string_enum": { // Тарифная опция типа string с выбором значения
из списка
        "type": "string",
        "description": "OS тип",
        "hint": "Операционная система",
        "enum": ["Ubuntu 20.4", "Windows 8.1", "Windows 10"],
        "default": "Windows 8.1"
    },
    "boolean_const": { // Тарифная опция-константа типа boolean
        "type": "boolean",
        "description": "Premium поддержка",
        "hint": "Техническая поддержка 24/7",
        "const": false
    },
    "boolean": { // Тарифная опция-переключатель типа boolean
        "type": "boolean",
        "description": "Уведомления об обновлениях",
        "hint": "Получать ли на почту уведомления о новых версиях

```



```
сервиса.",
    "default": true
  }
},
"update": {
  "parameters": {
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
    }
  }
},
"resource_usages": {
  "parameters": {
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
    }
  }
},
"service_binding": {
  "create": {
    "parameters": {
      "type": "object",
      "properties": {
      }
    }
  }
}
}
```

 Чтобы сделать тарифную опцию платной, опишите её стоимость в секции `plans.billing.options`.


5.3.3.4.2. Постоплатные тарифные опции

Постоплатные тарифные опции описываются в секции `properties` параметрами, приведёнными в таблице 33. Стоимость опции описывается в секции `billing`.

Таблица 33 — Параметры для постоплатных опций

Имя	Описание	Формат	Обязательный
description	Имя тарифной опции	string, до 255 символов	Да
hint	Подсказка с описанием тарифной опции	string, до 255 символов	Нет
type	Определяет тип тарифной опции. Укажите	-	Да

Имя	Описание	Формат	Обязательный
	значение <code>integer</code> или <code>number</code>		

 Имя опции в JSON-файле должно соответствовать значению `kind`, указанному в методе брокера, реализующем передачу отчёта в Магазин (подробнее — в подразделе [Брокер для SaaS-сервиса](#)).

В примере ниже в секции `schemas` описан тарифный план с опцией типа `number` (рисунок 53). Для опции задан постоплатный способ списания денежных средств.

Пример описания секции `schemas` для тарифного плана с постоплатной опцией

```
"schemas": {
  "service_instance": {
    "create": {
      "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
        }
      }
    },
    "update": {
      "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
        }
      }
    }
  },
  "resource_usages": {
    "parameters": {
      "$schema": "http://json-schema.org/draft-04/schema#",
      "type": "object",
      "properties": {
        "storage": {
          "description": "Хранение в ДЦ Киберпротект для продуктов Бэкап
Облачный",
          "type": "number"
        }
      }
    }
  }
},
"service_binding": {
  "create": {
    "parameters": {
      "type": "object",
      "properties": {
      }
    }
  }
}
```

```
}
}
}
```

✓ Выбран тариф «Лицензирование по ГБ» > 2 Подтверждение

Подтвердите подключение

Ознакомьтесь с итоговыми параметрами сконфигурированного тарифа и подключите его

Параметры тарифа	Значение	Стоимость
Выбранный тариф	Лицензирование по ГБ	-
Хранение в ДЦ Киберпротект для продуктов Кибер Бэкап Облачный	1 ГБ	7 ₽
	Спишется	0 ₽

Имя тарифной опции (description)

Назад
Подключить тариф

Рисунок 53 — Постоплатная тарифная опция

5.4. Загрузка конфигурации сервиса в брокер

Загрузите конфигурацию сервиса в брокер. Если SaaS-брокер создан по шаблону, то выполните действия:

1. Перейдите в директорию с брокером.
2. Поместите файл с конфигурацией сервиса `catalog_<SERVICE_NAME>.json` в директорию `resources`.
3. В файле `resources/plan_mapping.json` укажите:
 - `<SERVICE_NAME>` — имя сервиса.
 - `<PLAN_ID>` — ID тарифных планов, указанных в файле `catalog_<SERVICE_NAME>.json`.
 - `<SAAS_PLAN_ID>` (формат `string`) — ID тарифных планов на стороне сервиса, соответствующих ID тарифных планов `<PLAN_ID>`.

```
{
  "<SERVICE_NAME>": {
    "<PLAN_ID>": <SAAS_PLAN_ID>,
    "<PLAN_ID>": <SAAS_PLAN_ID>
  }
}
```

Пример содержимого файла `plan_mapping.json`

```
{
  "VKT": {
    "0dc54b75-8590-402b-a4b0-89d5a47eef71": 1,
    "312b628e-3089-4079-be1d-080ab4ca3acb": 2,
    "e2ae1588-9a35-4c2d-8559-3f5081a799a9": 3,
    "b54e7247-c9a3-49ff-88cd-7a9dccc22e7b": 4
  }
}
```

4. В директории с брокером переименуйте файл `.env.example` в `.env`.
5. В файле `.env` укажите имя сервиса в `BROKER_MODE`:

```
BROKER_MODE=<SERVICE_NAME>
```

5.5. Загрузка SaaS-сервиса в Магазин

5.5.1. Этапы загрузки сервиса

У пользователя, загружающего сервис в Магазин, должна быть учётная запись (личный кабинет) в облачной платформе. Инструкция по регистрации приведена в [официальной документации облачной платформы](#).

Этапы загрузки SaaS-сервиса в Магазин (рисунок 54):

1. Локальное тестирование брокера.
2. Разворачивание брокера на окружении.
3. Регистрация брокера.
4. Тестирование сервиса в Магазине.
5. Публикация сервиса.



Рисунок 54 — Загрузка SaaS-сервиса

5.5.2. Локальное тестирование брокера

Протестируйте брокер локально.

Чтобы локально протестировать брокер, разработанный на основе шаблона, выполните следующие действия:

1. В файле `.env` в директории брокера опишите переменные окружения:

- `BROKER_PROVIDER_CLIENT_ID` — имя брокера для доступа к SaaS-сервису.
- `BROKER_PROVIDER_SECRET` — пароль для доступа брокера к SaaS-сервису.
- `BROKER_PROVIDER_URL` — URL SaaS-сервиса.
- `BROKER_USERNAME` — имя Магазина для межсервисного взаимодействия с брокером.
- `BROKER_PASSWORD` — пароль Магазина для межсервисного взаимодействия с брокером.

⚠ Значение переменной `BROKER_USERNAME` должно совпадать со значением `username`, значение `BROKER_PASSWORD` — с `password`, указанными в заявке о регистрации брокера.

2. В файле `.env` в директории брокера укажите значения переменных для доступа к БД брокера.
3. Чтобы установить python-библиотеки, выполните команду:

```
$ pip install -r requirements/prod.txt
```

4. Чтобы запустить брокер, выполните команду:

```
$ gunicorn app.main:app --log-file - --workers ${UVICORN_WORKERS:-1} --worker-class
uvicorn.workers.UvicornWorker --bind 0.0.0.0:${BROKER_PORT:-8000} --timeout $
{WORKER_TIMEOUT:-90}
```

Пример ответа на команду запуска брокера

```
INFO:      Started server process [34934]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

После установки брокера по адресу `http://0.0.0.0:8000/docs` будет доступен API в редакторе swagger UI (рисунки 55, 56).

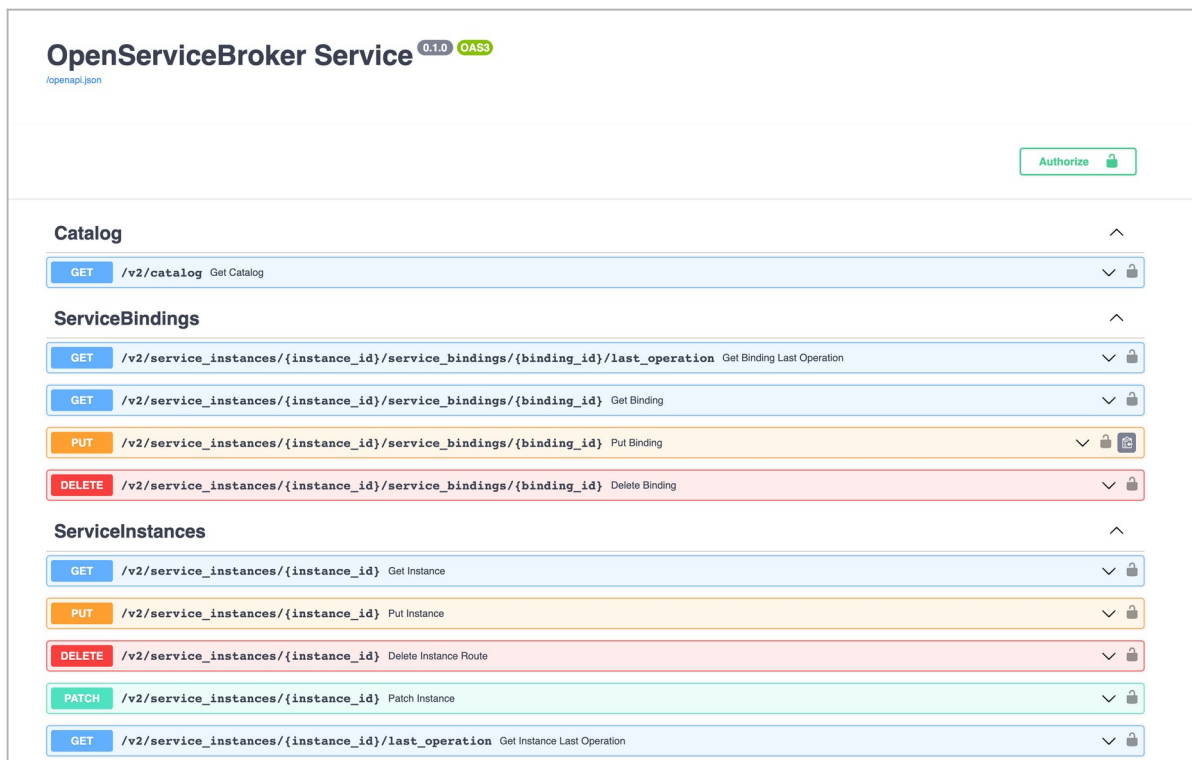


Рисунок 55 — Пример API swagger UI

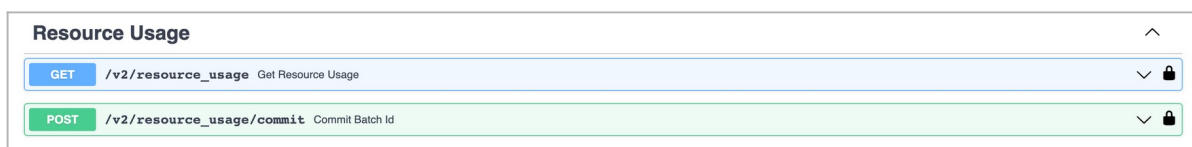


Рисунок 56 — Пример API swagger UI, содержащий запросы для постоплатных тарифных опций

5. Выполните запросы, описанные в API swagger UI.

В запросах используйте:

- Значение версии протокола VK OSB `x-broker-api-version` (например, `"0.1"`).
- Значение `BROKER_USERNAME` из файла `.env`.
- Значение `BROKER_PASSWORD` из файла `.env`.

5.5.3. Разворачивание брокера на окружении

Разверните образ с брокером на окружении.

Чтобы Магазин мог взаимодействовать с брокером и SaaS-сервисом, брокер должен быть доступен круглосуточно.

❗ Чтобы собрать образ, воспользуйтесь файлом `Dockerfile` в директории шаблона брокера.

5.5.4. Регистрация брокера

Чтобы зарегистрировать брокер, отправьте электронное письмо на marketplace@mcs.mail.ru. В письме укажите:

- Информацию о компании:
 - Название компании.
 - Контактное лицо.
 - Телефон.
- Параметры брокера, приведённые в таблице 34.

Таблица 34 — Параметры брокера для регистрации в Магазине

Имя	Описание	Формат	Обязательный
name	Имя брокера	string	Да
url	URL, на который будут отправляться запросы от Магазина	string	Да
description	Описание брокера	string	Нет
osb_version	Версия протокола VK OSB	string	Да
username	Имя Магазина для межсервисного взаимодействия с брокером. Должно совпадать со значением переменной окружения в файле <code>.env</code>	string	Нет
password	Пароль Магазина для межсервисного взаимодействия с брокером. Должно совпадать со значением переменной окружения в файле <code>.env</code>	string	Нет

После регистрации брокера SaaS-сервис будет доступен в тестовом пространстве имён Магазина. В открытом пространстве имён Магазина сервис будет доступен только после публикации (подробнее — в подразделе [Публикация сервиса](#)).

5.5.5. Тестирование сервиса в Магазине

До публикации сервис доступен в тестовых пространствах имён Магазина, указанных в параметре `metadata` в конфигурации сервиса.

Чтобы протестировать сервис в тестовом пространстве имён Магазина:

1. Чтобы получить доступ к тестовым пространствам имён Магазина, отправьте электронное письмо на marketplace@mcs.mail.ru. В письме укажите:

- Имена тестовых пространств имён, указанных в конфигурации сервиса.
- Email пользователя, которому необходим доступ в тестовые пространства имён Магазина.

Если доступ к тестовым пространствам имён, указанным в конфигурации сервиса, есть, то пропустите этот шаг.

2. Зайдите в ЛК облачной платформы.

3. Проверьте, что мастер конфигурации каждого тарифного плана отображается корректно.

4. Проверьте, что сервис можно установить.

5. Проверьте, что установленную конфигурацию инстанса сервиса можно обновить.

6. Проверьте, что установленный сервис можно удалить.

7. При необходимости внесите изменения в конфигурацию сервиса (подробнее — в разделе [Обновление SaaS-сервиса](#)).

5.5.6. Публикация сервиса

Потребителям сервис будет доступен только после публикации.

Чтобы опубликовать SaaS-сервис в Магазине, отправьте электронное письмо на marketplace@mcs.mail.ru. В письме укажите ID сервиса и его ревизию.

Будет проведено модерирование конфигурации сервиса. После этого сервис будет опубликован в Магазине.

Опубликованная ревизия сервиса будет доступна в открытых пространствах имён Магазина, указанных в параметре `metadata` в конфигурации сервиса. Сервис будет доступен потребителям.



Потребителям доступна только последняя опубликованная ревизия сервиса.

6. Обновление SaaS-сервиса

Обновление конфигурации SaaS-сервиса может включать:

- Изменение параметров тарифных планов и опций.
- Добавление тарифных планов и опций.
- Удаление тарифных планов и опций.

Чтобы обновить SaaS-сервис в Магазине:


1. Отредактируйте файл с конфигурацией сервиса.

а. Чтобы сохранить текущую опубликованную ревизию сервиса доступной в тестовых пространствах имён Магазина, в JSON-файл вставьте новую секцию для описания новой конфигурации сервиса:


```
{
  "services": [
    {
      <SERVICE_PARAMETERS>,
      "preview": {
        "parameters": [
        ]
      },
      "plans": [
      ]
    },
    {
      <SERVICE_PARAMETERS>,
      "preview": {
        "parameters": [
        ]
      },
      "plans": [
      ]
    }
  ]
}
```

Если текущую ревизию сервиса не требуется сохранить доступной в тестовом пространстве имён Магазина, то пропустите этот шаг.


- б. Опишите новую конфигурацию сервиса (подробнее — в подразделе [Конфигурация SaaS-сервиса](#)).
- в. В новой конфигурации сервиса укажите новую ревизию сервиса.

 Сервис не будет обновлён, если указанная ревизия сервиса уже загружена в Магазин.

г. Если были изменены параметры тарифных планов, то в новой конфигурации сервиса укажите новые ревизии планов.

 Тарифные планы не будут обновлены, если указанные ревизии планов уже загружены в Магазин.

2. Если были добавлены или удалены тарифные планы, то отредактируйте файл `resources/plan_mapping.json` в директории брокера (подробнее — в подразделе [Загрузка конфигурации сервиса в брокер](#)).
3. Магазин периодически запрашивает у брокера текущую конфигурацию сервиса, поэтому новая конфигурация автоматически попадет в тестовое пространство имён Магазина.

 Чтобы Магазин мог получать текущую конфигурацию сервиса, брокер должен быть доступен круглосуточно.

4. Протестируйте новую ревизию сервиса в тестовом пространстве имён Магазина (подробнее — в подразделе [Тестирование сервиса в Магазине](#)).
5. Опубликуйте новую ревизию сервиса в открытом пространстве имён Магазина (подробнее — в подразделе [Публикация сервиса](#)).

Предыдущая опубликованная ревизия переходит в тестовое пространство имён Магазина, если она сохранена в файле с конфигурацией сервиса. В тестовых пространствах имён сохраняются все ревизии, описанные в конфигурации сервиса.

Ревизии сервиса, удаленные из конфигурации сервиса, не будут доступны ни в тестовых, ни в открытых пространствах имён Магазина.

Публикация новой ревизии не влияет на уже существующие инстансы (тенанты/аккаунты) сервиса. Ревизия этих инстансов сервиса не изменяется, даже если она удалена из конфигурации сервиса и недоступна в Магазине.

А. Примеры YAML-файлов для image-based сервиса

А.1. Пример файла service.yaml

```
id: 72b70199-1823-40c8-aa7e-f43a23ddf380
revision: v. 1.0
name: VK Testers
short_description: Программа крауд-тестирования с многотысячным коммьюнити бета-
тестировщиков и собственной платформой для работы с данными
singleton: false
auto_bind: true
icon: icon.png
help: http://vk.cc/vktesters_po_faq
bindable: true
plan_updateable: false
deactivatable: false
bindings_retrievable: true
instances_retrievable: true

plans:
- name: free
- name: basic

preview:
  parameters:
    - name: api_requests_daily_limit
    - name: groups
    - name: products
```

А.2. Пример файла plan.yaml

```
id: b2b42648-5449-462f-b03e-b4cddb010b2
revision: v. 1.0
name: basic
description: Базовый
free: false

billing:
  cost: 2000

parameters_patch:
  users:
    schema.const: 5000
  volume_data_size:
    schema.default: 550
    schema.minimum: 550
```

В. Провайдер iVK CS

В.1. Ресурсы

Ресурсы провайдера iVK CS приведены в таблице 35.

Таблица 35 — Ресурсы провайдера iVK CS

Имя	Описание
ivkcs_agent_init	Инициализация агента на хостах и сопряжение агента с сервисом управления конфигурациями
ivkcs_agent_exec	Запуск скриптов в процессе разворачивания сервиса
ivkcs_agent_check	Мониторинг состояния инстанса сервиса на хостах
ivkcs_agent_status	Мониторинг статуса агентов на хостах
ivkcs_s3	Создание бакетов S3
ivkcs_dns	Создание А-записей в DNS облачной платформы (домены 5-ого и последующих уровней)
ivkcs_compute_instance_reboot	Перезагрузка хоста в процессе разворачивания сервиса

В.1.1. Ресурс ivkcs_agent_init

Аргументы ресурса `ivkcs_agent_init` приведены в таблице 36.

Таблица 36 — Аргументы ресурса ivkcs_agent_init

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
uuid	Идентификатор разворачивания сервиса	string	Да	Нет
hosts	Список имён хостов для инициализации агента	list, элементы списка — string	Да	Да
options	Опции агента	set, аргументы списка — в таблице 37	Нет	Нет

Таблица 37 — Опции инициализации агента

Имя	Описание	Формат	Значение по умолчанию
memory_limit	Ограничение оперативной памяти,	integer	256

Имя	Описание	Формат	Значение по умолчанию
	используемой для агента, МБ		

Ресурс `ivkcs_agent_init` вычисляет аргумент `agent` (формат map). Ключами аргумента являются имена хостов, значением — bash-скрипт. Функции bash-скрипта:

- Запрашивает агент у сервиса управления конфигурациями.
- Скачивает агент и устанавливает его на хосты.
- Выдает хостам ключи доступа в сервис управления конфигурациями.

Пример ресурса `ivkcs_agent_init`

```
resource "ivkcs_agent_init" "init" {
  uuid = "<UUID>"
  hosts = ["HOST"]
  options {
    memory_limit = 512
  }
}
```

В.1.2. Ресурс `ivkcs_agent_exec`

Аргументы ресурса `ivkcs_agent_exec` приведены в таблице 38.

Таблица 38 — Аргументы ресурса `ivkcs_agent_exec`

Имя	Описание	Формат	Обязательный
uuid	Идентификатор разворачивания сервиса	string	Да
name	Общее имя для скриптов, описанных в аргументе <code>step</code>	string	Да
hosts	Список имён хостов, где будет выполняться скрипт	list, элементы списка — string	Да
step	Описывает скрипты, которые будут запускаться агентами на хостах. Определяет порядок выполнения скриптов	list, структура одного элемента приведена в таблице ниже	Да

Аргументы одного элемента `step` приведены в таблице 39.




Таблица 39 — Аргументы одного элемента step

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
index	Индекс скрипта, определяет порядок выполнения	integer	Да	Нет
type	Язык скрипта. Возможные значения: <ul style="list-style-type: none"> • <code>bash</code>. • <code>python</code> 	string	Да	Нет
options	Параметры выполнения скрипта	set, аргументы списка — в таблице Опции скрипта	Нет	Да
content	Тело скрипта	string	Да	Да

Аргументы `options` для описания опций скрипта приведены в таблице 40.

Таблица 40 — Опции скрипта

Имя	Описание	Формат	Значение по умолчанию
environment_variables	Список переменных окружения, дополнительных к переменным окружения системы	list, элементы списка — string в формате <code><ENVIRONMENT_VARIABLE_NAME>=<VALUE></code>	—
exit_codes	Ожидаемые коды выхода скрипта. Если получен код выхода, означающий успешное выполнение скрипта, то скрипту присваивается статус <code>success</code> . Если получен код выхода, означающий завершение скрипта с ошибкой, то скрипт будет запущен повторно. Если в результате всех повторных попыток скрипт завершен с ошибкой, то статус скрипта — <code>failed</code> . Параметры повторного запуска задаются в опциях: <ul style="list-style-type: none"> • <code>attempts</code> • <code>attempt_pause</code> 	list, элементы списка — integer	0, -1. Код 0 — скрипт завершен успешно. Код -1 — с ошибкой
halt_codes	Коды выхода скрипта, при которых скрипт не будет запущен повторно	list, элементы списка — integer	125
stop_signal	Сигнал, отправляемый скрипту, если превышен тайм-аут его выполнения — <code>SIGTERM</code> .	integer	15

Имя	Описание	Формат	Значение по умолчанию
	Тайм-аут задается в опции <code>timeout</code>		
stop_timeout	<p>Тайм-аут, отсчитываемый с момента получения сигнала <code>SIGTERM</code>. По истечении тайм-аута процесс завершается принудительно — <code>SIGKILL</code>.</p> <p>Пример значений:</p> <ul style="list-style-type: none"> • <code>1m</code>. • <code>15s</code>. • <code>1h</code> 	string	5s
cwd	Директория для выполнения скрипта	string	<code>/tmp</code>
user	<p>Пользователь, от имени которого скрипт будет запущен.</p> <p> Если указанный пользователь не будет найден, то скрипт будет запущен от пользователя по умолчанию.</p>	string	root
group	<p>Группа пользователя, от имени которого скрипт будет запущен.</p> <p> Если указанная группа пользователей не будет найдена, то скрипт будет запущен от группы по умолчанию.</p>	string	root
timeout	<p>Тайм-аут выполнения скрипта. Если скрипт не завершился в течение тайм-аута, то ему отправляется сигнал <code>SIGTERM</code> (задан в опции <code>stop_signal</code>).</p> <p>Значение задается в таком же формате, как и опция <code>stop_timeout</code></p>	string	60s
attempts	<p>Количество попыток повторного запуска скрипта, если получены коды выхода о завершении скрипта с ошибкой (заданы в опции <code>exit_codes</code>).</p> <p> Коды из списка <code>halt_codes</code> не приводят к повторному запуску скрипта.</p>	integer	3
attempt_pause	<p>Длительность паузы между повторными запусками скрипта.</p> <p>Значение задается в таком же формате, как и опция <code>stop_timeout</code></p>	string	5s
pause_before	Длительность паузы перед запуском скрипта.	string	0

Имя	Описание	Формат	Значение по умолчанию
	Значение задается в таком же формате, как и опция <code>stop_timeout</code>		
pause_after	Длительность паузы после завершения скрипта. Значение задается в таком же формате, как и опция <code>stop_timeout</code>	string	0
store_result	Сохранять или нет результат выполнения скрипта на диск хоста	boolean	true
structured_result_file	Путь файла с форматированным результатом скрипта	string	—
once	Определяет, будет ли скрипт выполняться только один раз или будет выполняться при каждом повторном разворачивании сервиса. Если задано значение <code>true</code> , то скрипт будет выполнен только при первоначальном разворачивании сервиса	boolean	false

Для каждого скрипта, описанного в `step`, ресурс `ivkcs_agent_exec` вычисляет аргумент `hash` (формат string), который содержит хеш скрипта для передачи в сервис управления конфигурациями. По полученным значениям сервис управления конфигурациями сравнивает текущее и целевое (описано в манифесте Terraform) состояние хостов.

В.1.3. Ресурс `ivkcs_agent_check`

Аргументы ресурса `ivkcs_agent_check` приведены в таблице 41.

Таблица 41 — Аргументы ресурса `ivkcs_agent_check`

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
uuid	Идентификатор разворачивания сервиса	string	Да	Да
hosts	Список имён хостов	list, элементы списка — string	Да	Да

В ресурсе указывается один или несколько способов мониторинга состояния инстансов сервиса. Поддерживаются следующие способы мониторинга:

- Проверка доступности порта — описывается в блоке `port_health`.
- Проверка доступности адреса — описывается в блоке `http_health`.

- Проверка доступности инстансов сервиса с использованием скриптов — описывается в блоке `script_health`.

Ресурс `ivkcs_agent_check` вычисляет аргумент `hash` (формат string), который содержит хеш результата мониторинга для передачи в сервис управления конфигурациями.

Пример ресурса `ivkcs_agent_check`

```
resource "ivkcs_agent_check" "check1" {
  uuid = "<UUID>"
  hosts = ["HOST"]

  http_health {
    protocol = "http"
    method = "GET"
    host = "127.0.0.1"
    path = "status"
    period = "1m"
    port = 11012
    http_codes = [200]
  }
}
```

В.1.3.1. Блок `port_health`

Каждый порт описывается отдельным блоком `port_health`.

Аргументы для блока `port_health` приведены в таблице 42.

Таблица 42 — Аргументы блока `port_health`

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
host	IP-адрес или DNS-имя хоста	string	Да	Да
period	Периодичность проверки доступности порта. Пример значений: <ul style="list-style-type: none"> • <code>1m</code>. • <code>15s</code>. • <code>1h</code> 	string	Да	Да
port	Проверяемый порт	integer	Да	Да

В.1.3.2. Блок `http_health`

Каждая проверка адреса описывается отдельным блоком `http_health`.

Аргументы для блока `http_health` приведены в таблице 43.

Таблица 43 — Аргументы блока `http_health`

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
<code>protocol</code>	Протокол запроса. Возможные значения: <ul style="list-style-type: none"> <code>http</code>. <code>https</code> 	string	Да	Да
<code>method</code>	Метод запроса. Возможные значения: <ul style="list-style-type: none"> <code>GET</code>. <code>POST</code>. <code>PUT</code>. <code>DELETE</code>. <code>OPTIONS</code> 	string	Да	Да
<code>host</code>	IP-адрес или DNS-имя хоста	string	Да	Да
<code>path</code>	Путь запроса (конечная точка (endpoint) метода)	string	Нет	Да
<code>period</code>	Периодичность проверки доступности адреса. Пример значений: <ul style="list-style-type: none"> <code>1m</code>. <code>15s</code>. <code>1h</code> 	string	Да	Да
<code>port</code>	Проверяемый порт	integer	Нет	Да
<code>http_codes</code>	Ожидаемые коды ответа	list, элементы списка — integer	Да	Да
<code>url</code>	Полный адрес, включающий: <ul style="list-style-type: none"> IP-адрес или DNS-имя хоста. Порт. Путь запроса (конечная точка (endpoint) метода) 	string	Нет	Да



Если указан аргумент `url`, то его значение будет иметь приоритет над значениями в аргументах `host`, `path` и `port`.

В.1.3.3. Блок `script_health`

Каждый скрипт описывается отдельным блоком `script_health`.

Аргументы для блока `script_health` приведены в таблице 44.

Таблица 44 — Аргументы блока `script_health`

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
type	Язык скрипта. Возможные значения: <ul style="list-style-type: none"> <code>bash</code>. <code>python</code> 	string	Да	Да
options	Параметры выполнения скрипта	set, аргументы списка — в таблице Опции скрипта	Нет	Да
script	Тело скрипта	string	Да	Да

В.1.4. Ресурс `ivkcs_agent_status`

Аргументы ресурса `ivkcs_agent_status` приведены в таблице 45.

Таблица 45 — Аргументы ресурса `ivkcs_agent_status`

Имя	Описание	Формат	Обязательный
uuid	Идентификатор разворачивания сервиса	string	Да
host	Имя хоста	string	Да

Ресурс `ivkcs_agent_status` вычисляет аргумент `status` — статус агента на хосте, передаваемый в сервис управления конфигурациями. Возможные значения:

- `ok` — агент доступен.
- `running` — агент в процессе инициализации.
- `failed` — агент недоступен.


Пример ресурса `ivkcs_agent_status`

```
resource "ivkcs_agent_status" "status_host1" {
  uuid = "<UUID>"
  host = ["HOST1"]
  depends_on = [
    ivkcs_agent_init.init,
  ]
}
```

В.1.5. Ресурс ivkcs_s3

Аргумент ресурса `ivkcs_s3` приведен в таблице 46.

Таблица 46 — Аргументы ресурса ivkcs_s3

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
name	<p>Уникальное имя бакета S3.</p> <p> Чтобы обеспечить уникальность имени, используйте идентификатор разворачивания сервиса UUID как часть имени.</p>	string	Да	Да

Ресурс `ivkcs_s3` вычисляет аргументы, приведённые в таблице 47.

Таблица 47 — Вычисляемые аргументы ресурса ivkcs_s3

Имя	Описание	Формат
access	Открытый ключ для доступа к бакету	string
secret	Закрытый ключ для доступа к бакету	string


Пример ресурса `ivkcs_s3`

```
resource "ivkcs_s3" "test" {
  name = "<UUID>_bucket_test"
}
```

В.1.6. Ресурс ivkcs_dns

Аргументы ресурса `ivkcs_dns` приведены в таблице 48.

Таблица 48 — Аргументы ресурса ivkcs_dns

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
name	<p>Уникальное доменное имя 5-го или последующего уровня.</p> <p> Чтобы обеспечить уникальность имени, используйте идентификатор разворачивания сервиса UUID как часть имени.</p>	string	Да	Да
domain	Домен, поддерживаемый облачной платформой.	string	Да	Да

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
	Значение должно быть <code>xaas.msk.vkcs.cloud</code>			
ip	IP-адрес хоста	string	Да	Нет

Пример ресурса `ivkcs_dns`

```
resource "ivkcs_dns" "dns" {
  name = "<UUID>_testname"
  domain = "xaas.msk.vkcs.cloud"
  ip = "8.8.8.8"
}
```

В.1.7. Ресурс `ivkcs_compute_instance_reboot`

Аргумент ресурса `ivkcs_compute_instance_reboot` приведен в таблице 49.

Таблица 49 — Аргумент ресурса `ivkcs_compute_instance_reboot`

Имя	Описание	Формат	Обязательный	Пересоздание ресурса при изменении значения
instance	ID хоста, который будет перезагружен. ID вычисляется ресурсом, создающим хост (ресурс <code>vkcs_compute_instance</code> провайдера VK CS)	string	Да	Нет

Пример ресурса `ivkcs_dns`

```
resource "ivkcs_compute_instance_reboot" "init" {
  instance = vkcs_compute_instance.<RESOURCE_NAME>.id
}
```

В.2. Источники данных

Провайдер iVK CS предоставляет источник данных `ivkcs_agent_script_result` — получение форматированного результата скрипта, выполненного в агенте. Позволяет получить данные в процессе разворачивания сервиса, передать их в Terraform и использовать в дальнейших действиях для разворачивания сервиса.

Аргументы источника данных `ivkcs_agent_script_result` приведены в таблице 50.

Таблица 50 — Аргументы источника данных `ivkcs_agent_script_result`

Имя	Описание	Формат	Обязательный
uuid	Идентификатор разворачивания сервиса	string	Да
host	Имя хоста, где был выполнен скрипт. Соответствует имени, указанному в аргументе <code>hosts</code> ресурса <code>ivkcs_agent_exec</code>	string	Да
group	Имя группы скриптов. Соответствует значению аргумента <code>name</code> ресурса <code>ivkcs_agent_exec</code>	string	Да
index	Индекс скрипта. Соответствует значению аргумента <code>step.index</code> ресурса <code>ivkcs_agent_exec</code>	integer	Да

Источник данных `ivkcs_agent_script_result` вычисляет аргументы:

- `result` — результат выполнения скрипта.
- `structured_result` — форматированный результат выполнения скрипта.

С. Пример манифеста для сервиса Kafka

Ниже приведен манифест для разворачивания динамически собираемого кластера Kafka, с мониторингом и автовосстановлением инстанса сервиса.

Пример манифеста для сервиса Kafka

```
# -----variables-----
# Специальные переменные
# Идентификатор разворачивания сервиса
variable "instance_uuid" {
    type    = string
}

# Внешние переменные
# Имя виртуальной сети
variable "sub_network" {
    type      = string
    default   = "kafka-net"
    description = ""
}

# Идентификатор образа сервиса
variable "image_uuid" {
    type      = string
    default   = "8c7a6443-bb79-4f04-884a-14231f0ba6cb"
    description = "kafka image"
}

# Зона доступности
variable "availability_zone" {
    type      = string
    default   = "GZ1"
}

# Коэффициент для вычисления количества хостов в переменной count_vm
variable "coefficient_kafka" {
    type      = number
    default   = 1
    description = "2*n + 1"
}

# Порты доступа для группы безопасности
variable "ports" {
    type      = list(number)
    default   = [22, 2182, 2888, 3888, 9092, 9093, 9091]
    description = "ports for secgroup rule. ZooKeeper: [2182, 2888, 3888]. Kafka: [9092, 9093, 9091]"
}

# Тип отдельного тома диска для хранения данных
variable "volume_type" {
    type      = string
    default   = "ceph-ssd"
    description = "one of: high-iops, ceph-ssd, ceph-hdd"
```



```
}

# Размер отдельного тома диска для хранения данных
variable "volume_size" {
  type    = number
  default = 4
  description = "one of: high-iops ceph-ssd ceph-hdd"
}

# Количество CPU для VM
variable "flavor_vcpus" {
  type    = number
  default = 2
}

# Количество RAM для VM, МБ
variable "flavor_ram" {
  type    = number
  default = 2048
}

# Локальные переменные для ZooKeeper и Kafka
variable "ssl_ca_password" {
  type    = string
  default = "SSL_CA_PASSWORD"
  description = "Пароль к корневому самоподписанному сертификату"
}

# Максимальное количество топиков для Kafka
variable "kafka_max_topics" {
  type    = number
  default = 15
}

variable "zookeeper_truststore_storepass" {
  type    = string
  default = "ZOOKEEPER_TRUSTSTORE_STOREPASS"
  description = "Java-контейнер с сертификатами, файлом. Используется для коммуникации всех ZooKeeper между собой"
}

variable "zookeeper_keystore_storepass" {
  type    = string
  default = "ZOOKEEPER_KEYSTORE_STOREPASS"
  description = "Хранит клиентский ключ"
}

# Используется для коммуникации между инстансами Kafka
variable "kafka_broker_truststore_storepass" {
  type    = string
  default = "KAFKA_BROKER_TRUSTSTORE_STOREPASS"
}

# Хранит клиентский ключ
variable "kafka_broker_keystore_storepass" {
```

```

type    = string
default = "KAFKA_BROKER_KEYSTORE_STOREPASS"
}

# ---
locals {
  # Вычисление количества хостов через коэффициент, заданный в переменной
  coefficient_kafka
  count_vm = 2 * var.coefficient_kafka + 1
  # Генерация имён хостов
  hosts_name = [for i in range(local.count_vm): "${var.instance_uuid}-kafka-${i + 1}"]
}

# -----network-----
# Создание виртуальной сети
resource "vkcs_networking_network" "subnet" {
  # Имя сети, заданное в переменной sub_network
  name = var.sub_network
}

# Создание виртуальной подсети
resource "vkcs_networking_subnet" "compute" {
  name          = "kafka_subnet_1"
  # Идентификатор сети, где будет создана подсеть
  network_id = vkcs_networking_network.subnet.id
  # Диапазон IP-адресов подсети
  cidr         = "192.168.199.0/24"
}

# -----security group-----
# Создание группы безопасности
resource "vkcs_networking_secgroup" "secgroup" {
  # Имя группы безопасности
  name = "${var.instance_uuid}-security_group_for_kafka"
  description = "kafka security group"
  # Зависимости. Группа безопасности будет создана только после создания ресурсов,
указанных в зависимостях
  depends_on = [
    # Инициализация агентов
    ivkcs_agent_init.init,
  ]
}

# Правила для группы безопасности
resource "vkcs_networking_secgroup_rule" "rules" {
  count = length(var.ports)
  # Определение направления применения правил – для входящих (ingress) или исходящих
(egress) соединений
  direction = "ingress"
  # Список портов доступа, заданных в переменной ports
  port_range_max = element(var.ports, count.index)
  port_range_min = element(var.ports, count.index)
  # Протокол доступа
  protocol = "tcp"
  # Идентификатор группы безопасности, для которой созданы правила
  security_group_id = vkcs_networking_secgroup.secgroup.id
}

```

```
description = "secgroup_rule_${count.index + 1}"
}

# -----volume-----
# Создание отдельных томов дисков для хранения данных
resource "vkcs_blockstorage_volume" "kafka_data" {
  # Создание количества томов дисков, соответствующее количеству хостов
  count = local.count_vm
  name = "${var.instance_uuid}-kafka-${count.index + 1}-storage"
  # Размер диска, заданный в переменной volume_size
  size = var.volume_size
  # Зона доступности, заданная в переменной availability_zone
  availability_zone = var.availability_zone
  # Тип диска, заданный в переменной volume_type
  volume_type = var.volume_type
}

# -----conf system-init-----
# Получение данных для инициализации агентов на хостах
resource "ivkcs_agent_init" "init" {
  # Идентификатор разворачивания сервиса
  uuid = var.instance_uuid
  # Список имён хостов
  hosts = local.hosts_name
}

# Преобразование данных, чтобы их использовать в скрипте ivkcs_agent_exec.sync
locals {
  ansible = {"hosts" = [for i in range(local.count_vm): {
    hostname: "${element(local.hosts_name, i)}.novalocal",
    ip: vkcs_compute_instance.compute[i].access_ip_v4,
    id: i + 1
  }]}
}

# Получение статусов агентов на хостах
resource "ivkcs_agent_status" "status_host" {
  count = length(local.hosts_name)
  # Имя хоста
  host = local.hosts_name[count.index]
  # Идентификатор разворачивания сервиса
  uuid = ivkcs_agent_init.init.uuid
  # Зависимости. Статус будет создан только после инициализации агента
  depends_on = [
    ivkcs_agent_init.init,
  ]
}

# Проброс статусов хостов, полученных в ресурсе ivkcs_agent_status, через транзитный ресурс
resource "null_resource" "replacement_trigger" {
  count = length(local.hosts_name)
  triggers = {
    status = ivkcs_agent_status.status_host[count.index].id
  }
}
```

```

}

# -----image-----
# Получение доступных типов VM
data "vkcs_compute_flavor" "compute" {
  # Значения CPU и RAM для фильтрации типов VM, заданные в переменных flavor_vcpus и
  flavor_ram
  vcpus = var.flavor_vcpus
  ram   = var.flavor_ram
}

# Описание конфигурации виртуальных машин, создаваемых из загрузочного образа сервиса
resource "vkcs_compute_instance" "compute" {
  # Создание VM в количестве, вычисленном в переменной count_vm
  count = local.count_vm
  # Имя VM, сгенерированное в переменной hosts_name
  name = element(local.hosts_name, count.index)
  # Идентификатор типа VM
  flavor_id = data.vkcs_compute_flavor.compute.id
  # Перечень имён групп безопасности для VM
  security_groups = [vkcs_networking_secgroup.secgroup.name]
  # Зона доступности VM, заданная в переменной availability_zone
  availability_zone = var.availability_zone
  # Описание конфигурации виртуального root-диска VM
  block_device {
    # Идентификатор образа сервиса, заданный в переменной image_uuid
    uuid = var.image_uuid
    # Источник загрузки root-диска – образ сервиса (image)
    source_type = "image"
    # Тип диска
    destination_type = "volume"
    # Тип тома диска
    volume_type = "ceph-ssd"
    # Размер тома диска, ГБ
    volume_size = 10
    # Место диска в порядке загрузки boot
    boot_index = 0
    # Если указано значение true, диск будет удален при удалении VM
    delete_on_termination = true
  }
  # Инициализация агентов на виртуальных машинах
  user_data = ivkcs_agent_init.init.agent[element(local.hosts_name, count.index)]

  # Подсеть, где будет размещаться VM
  network {
    # Идентификатор сети VM
    uuid = vkcs_networking_network.subnet.id
  }

  # Зависимости. VM будут созданы только после создания ресурсов, указанных в
  зависимостях
  depends_on = [
    # Сеть VM
    vkcs_networking_network.subnet,
    # Подсеть VM
    vkcs_networking_subnet.compute,
  ]
}

```

```

# Диски для хранения данных сервиса
vkcs_blockstorage_volume.kafka_data,
# Инициализация агента
ivkcs_agent_init.init,
# Правила группы безопасности
vkcs_networking_secgroup_rule.rules,
]

# Попытка остановить ВМ перед удалением
stop_before_destroy = true

# Тайм-аут для создания ВМ
timeouts {
  create = "10m"
}

# Настройка пересоздания ВМ на основе статуса агента
lifecycle {
  replace_triggered_by = [
    null_resource.replacement_trigger[count.index]
  ]
}
}

# Присоединение томов дисков к созданным виртуальным машинам
resource "vkcs_compute_volume_attach" "attached" {
  count = local.count_vm
  # Идентификатор ВМ, к которой присоединяется том диска
  instance_id = element(vkcs_compute_instance.compute[*].id, count.index)
  # Идентификатор присоединяемого тома диска
  volume_id   = element(vkcs_blockstorage_volume.kafka_data[*].id, count.index)

  # Зависимости. Диски будут присоединены к ВМ только после создания ресурсов, указанных
  в зависимостях
  depends_on = [
    # ВМ
    vkcs_compute_instance.compute,
  ]
}

# -----conf system-run-----

# Скрипт, генерирующий SSL-сертификат на хосте с индексом 0
resource "ivkcs_agent_exec" "initssl" {
  hosts = [local.hosts_name[0]]
  name = "init ssl"
  uuid = var.instance_uuid
  step {
    index    = 1
    type     = "bash"
    content = <<-EOT
#!/bin/bash
ansible-playbook ssl_init.yml --extra-vars "ssl_ca_password=${var.ssl_ca_password}"
EOT
    options = {
      timeout = "5m"
      structured_result_file = "/tmp/cert_result.json"
    }
  }
}
```

```

        cwd = "/opt/playbooks"
        attempts = 1
        once = true
    }
}
Зависимости. Скрипт будет выполнен только после создания ресурсов, указанных в
зависимостях
depends_on = [
    # ВМ
    vkcs_compute_instance.compute,
    # Инициализация агента
    ivkcs_agent_init.init,
]
}

# Получение форматированного результата скрипта, генерирующего SSL-сертификат
data "ivkcs_agent_script_result" "ssl" {
    uuid = var.instance_uuid
    host = ivkcs_agent_exec.initssl.hosts[0]
    group = ivkcs_agent_exec.initssl.name
    index = 1
    depends_on = [ivkcs_agent_exec.initssl]
}

# Скрипт, синхронизирующий узлы в кластере с помощью ZooKeeper и распространяющий
сгенерированный SSL-сертификат на все узлы кластера
resource "ivkcs_agent_exec" "sync" {
    hosts = local.hosts_name
    name = "sync kafka"
    uuid = var.instance_uuid
    step {
        index = 1
        type = "bash"
        content = <<-EOT
#!/bin/bash
ansible-playbook start.yml \
--extra-vars "ssl_ca_password=${var.
}" \
--extra-vars "zookeeper_truststore_storepass=${var.zookeeper_truststore_storepass}" \
--extra-vars "zookeeper_keystore_storepass=${var.zookeeper_keystore_storepass}" \
--extra-vars "kafka_broker_truststore_storepass=${
var.kafka_broker_truststore_storepass}" \
--extra-vars "kafka_broker_keystore_storepass=${var.kafka_broker_keystore_storepass}" \
--extra-vars "current_host=$(hostname)" \
--extra-vars "max_expected_topics=${var.kafka_max_topics}" \
--extra-vars '${jsonencode(local.ansible)}' \
--extra-vars '${jsonencode(data.ivkcs_agent_script_result.ssl.structured_result)}'
EOT
        options = {
            timeout = "10m"
            cwd = "/opt/playbooks"
            attempts = 1
        }
    }
}
}

```

```
# -----health check-----

# Мониторинг инстансов сервиса
resource "ivkcs_agent_check" "health" {
  hosts = local.hosts_name
  uuid  = var.instance_uuid

  # Мониторинг Kafka по порту
  port_health {
    host = "127.0.0.1"
    port  = 9092
    period = "1m"
  }

  # Мониторинг ZooKeeper по порту
  port_health {
    host = "127.0.0.1"
    port  = 2182
    period = "1m"
  }

  # Зависимости. Мониторинг будет выполнен только после создания ресурсов, указанных в
  зависимостях
  depends_on = [
    # Синхронизация узлов в кластере
    ivkcs_agent_exec.sync,
  ]

  # Тайм-аут создания health check
  timeouts {
    create = "5m"
  }
}

# -----outputs-----

# Вывод IP-адресов виртуальных машин
output "address" {
  description = "The addresses for hosts"
  value = vkcs_compute_instance.compute[*].access_ip_v4
}

# Вывод SSL-сертификата
output "cert" {
  description = "ssl base64 CA cert, in tar.gz"
  value = data.ivkcs_agent_script_result.ssl.structured_result["cert_archive"]
  # Выходной параметр содержит чувствительные данные
  sensitive = true
}

# Вывод информации об образе сервиса
output "image" {
  description = "instance info"
  value = vkcs_compute_instance.compute
  # Выходной параметр содержит чувствительные данные
  sensitive = true
}
```

D. Пример JSON-файла для SaaS-сервиса

```
{
  "services": [
    {
      "id": "04527a41-5948-4e09-9a1a-57e1aecb3ebc",
      "revision": "v. 1.0",
      "name": "VK Testers",
      "short_description": "Программа крауд-тестирования с многотысячным комьюнити бета-тестировщиков и собственной платформой для работы с данными",
      "full_description": "Мы предлагаем удобный баг-трекер для организации процесса тестирования:\n  - Автоматизация процессов с помощью API\n  - Гибкая настройка проектов\n  - Интеграция с внешними таск-трекерами\n  - Дистрибуция тестовых сборок\n  - Desktopная и мобильная версии баг-трекера",
      "singleton": false,
      "auto_bind": true,
      "icon": "https://vk.com/images/bugs/vktesters_logo.svg",
      "help": "http://vk.cc/vktesters_po_faq",
      "bindable": true,
      "plan_updateable": true,
      "deactivatable": false,
      "bindings_retrievable": true,
      "instances_retrievable": true,
      "preview": {
        "parameters": [
          {
            "name": "api_requests_daily_limit"
          },
          {
            "name": "products"
          },
          {
            "name": "members"
          }
        ]
      },
    },
    {
      "plans": [
        {
          "id": "f6593bfb-c0b8-40a3-8b82-c05e07f6ae9a",
          "revision": "v. 2.0",
          "name": "free",
          "description": "Бесплатный",
          "free": true,
          "metadata": {
            "test_ns": ["test"],
            "prod_ns": ["vkcs_ru"]
          },
          "display": {
            "pages": [
              {
                "name": "Настройки",
                "index": 0,
                "groups": [

```



```

    "name": "",
    "index": 0,
    "parameters": [
      {
        "name": "api_requests_daily_limit",
        "index": 0
      },
      {
        "name": "groups",
        "index": 1
      },
      {
        "name": "products",
        "index": 2
      },
      {
        "name": "members",
        "index": 3
      }
    ]
  }
}

]
}
}
}
}

"billing": {
  "cost": 0,
  "options": {
    "products": {
      "base": 5,
      "cost": 100,
      "unit": {
        "size": 10
      }
    },
    "groups": {
      "base": 0,
      "cost": 200,
      "unit": {
        "size": 5
      }
    },
    "members": {
      "base": 5,
      "cost": 100,
      "unit": {
        "size": 1
      }
    },
    "checklists_per_product": {
      "base": 2,
      "cost": 50,
      "unit": {
        "size": 1
      }
    },
  },

```

```

    "api_requests_daily_limit": {
      "base": 1000,
      "cost": 50,
      "unit": {
        "size": 1000,
        "measurement": "Запросы в сутки"
      }
    }
  },
  "schemas": {
    "service_instance": {
      "create": {
        "parameters": {
          "$schema": "http://json-schema.org/draft-04/schema#",
          "type": "object",
          "properties": {
            "products": {
              "description": "Количество продуктов",
              "hint": "Предельное количество тестируемых на платформе проектов: к продуктам прикрепляются отчёты и чек-листы, а также загружаются сборки.",
              "type": "integer",
              "minimum": 0,
              "default": 0
            },
            "groups": {
              "type": "integer",
              "description": "Количество групп пользователей",
              "hint": "Возможность разделения сотрудников заказчика на отдельные группы с различными правами в выбранных продуктах. Инструмент позволяет структурировать доступы и тестируемые продукты, объединяя их в группы по обозначенным критериям, что повышает уровень контроля.",
              "minimum": 0,
              "default": 0
            },
            "members": {
              "type": "integer",
              "description": "Количество участников",
              "hint": "Количество сотрудников компании заказчика, которые могут использовать инфраструктуру тестирования и обрабатывать отчёты от тестировщиков VK Testers.",
              "minimum": 0,
              "default": 0
            },
            "checklists_per_product": {
              "type": "integer",
              "description": "Количество чек-листов на продукт",
              "hint": "Чек-лист — список действий, которые нужно выполнить при тестировании продукта. Большее количество чек-листов позволяет лучше структурировать проводимые участниками VK Testers проверки.",
              "minimum": 0,
              "default": 0
            }
          }
        },
        "api_requests_daily_limit": {
          "type": "integer",
          "description": "Доступ к открытому API",

```

```

        "hint": "Открытое API помогает встроить платформу тестирования в
работу существующих в вашей компании решений. Увеличенный суточный лимит на запросы
позволяет поддерживать больше интеграций одновременно.",
        "minimum": 0,
        "default": 0
    },
    "report_notifications": {
        "type": "boolean",
        "description": "Уведомления о новых отчетах",
        "hint": "Настройка мгновенных уведомлений для сотрудников компании
о появлении новых отчётов.",
        "const": false
    }
}
},
"update": {
    "parameters": {
        "$schema": "http://json-schema.org/draft-04/schema#",
        "type": "object",
        "properties": {
            "products": {
                "description": "Количество продуктов",
                "hint": "Предельное количество тестируемых на платформе проектов:
к продуктам прикрепляются отчёты и чек-листы, а также загружаются сборки.",
                "type": "integer",
                "minimum": 0,
                "default": 0
            },
            "groups": {
                "type": "integer",
                "description": "Количество групп пользователей",
                "hint": "Возможность разделения сотрудников заказчика на отдельные
группы с различными правами в выбранных продуктах. Инструмент позволяет структурировать
доступы и тестируемые продукты, объединяя их в группы по обозначенным критериям, что
повышает уровень контроля.",
                "minimum": 0,
                "default": 0
            },
            "members": {
                "type": "integer",
                "description": "Количество участников",
                "hint": "Количество сотрудников компании заказчика, которые могут
использовать инфраструктуру тестирования и обрабатывать отчёты от тестировщиков VK
Testers.",
                "minimum": 0,
                "default": 0
            },
            "checklists_per_product": {
                "type": "integer",
                "description": "Количество чек-листов на продукт",
                "hint": "Чек-лист – список действий, которые нужно выполнить при
тестировании продукта. Больше количество чек-листов позволяет лучше структурировать
проводимые участниками VK Testers проверки.",
                "minimum": 0,
                "default": 0
            }
        }
    }
}

```

```

    },
    "api_requests_daily_limit": {
      "type": "integer",
      "description": "Доступ к открытому API",
      "hint": "Открытое API помогает встроить платформу тестирования в  
работу существующих в вашей компании решений. Увеличенный суточный лимит на запросы  
позволяет поддерживать больше интеграций одновременно.",
      "minimum": 0,
      "default": 0
    },
    "report_notifications": {
      "type": "boolean",
      "description": "Уведомления о новых отчетах",
      "hint": "Настройка мгновенных уведомлений для сотрудников компании  
о появлении новых отчётов.",
      "const": false
    }
  }
}

```

Е. Термины и сокращения

В настоящем документе используются термины, перечисленные в таблице ниже.

Термин	Определение
А-запись	Адресная запись в DNS, указывающая соответствие между именем и IP-адресом
Бакет	Контейнер для хранения объектов S3
Брокер (Broker)	Сущность, обеспечивающая интеграцию сервиса и Магазина. Брокер управляет жизненным циклом инстанса сервиса. Брокер разрабатывается в соответствии с протоколом VK OSB
Группа безопасности (Security Group)	Наборы настраиваемых разрешающих правил, которые регулируют права сетевого доступа (вход по определенному протоколу, через конкретный порт) для определенных IP-адресов или групп безопасности
Диск (Volume)	Сетевое блочное устройство, обеспечивающее хранилище данных для VM
Закрытый ключ (private key)	Набор секретных данных, определяющий алгоритм и криптографический материал для дешифрования. Доступен только владельцу зашифрованных данных
Инстанс (Instance)	Виртуальная машина, которая запускается и работает в облачной платформе
Инстанс сервиса (Service instance)	Экземпляр сервиса, развернутый у пользователя. Для SaaS-сервиса инстанс сервиса — это арендатор/аккаунт SaaS-сервиса
Открытое пространство имён Магазина	Пространство имён, обеспечивающее доступ потребителей к опубликованным ревизиям сервисов, размещённых в Магазине
Открытый ключ (public key)	Набор секретных данных, определяющий алгоритм и криптографический материал для шифрования
Ревизия сервиса/плана	Версия сервиса/плана
Сервисная привязка (Service binding)	Сущность, представляющая собой запрос на использование инстанса сервиса
Сервисный пакет (VK Cloud product package)	Структурированный набор файлов, определенным образом описывающих image-based сервис и процесс его разворачивания на VM в облачном проекте
Система развёртывания (Deployment system)	Система, обеспечивающая разворачивание image-based сервиса на VM в облачной платформе. Управляет инфраструктурой и ПО сервиса
Тестовое пространство имён Магазина	Пространство имён, предназначенное для тестирования конфигурации сервиса перед публикацией. Доступно поставщикам, недоступно потребителям
Тип VM (Flavor)	Конфигурация инстанса, определяющая характеристики его ресурсов. Тип VM определяет количество и тип CPU, объём RAM
Узел (Node)	Отдельная VM, где развёрнуто и выполняется программное обеспечение
Image-based сервис	Продукт, который разворачивается на основе образов виртуальных машин в

Термин	Определение
	облачном проекте пользователя
SaaS-сервис	Централизованно установленный мультитенантный продукт

В настоящем документе используются сокращения, перечисленные в таблице ниже.

Сокращение	Расшифровка
БД	База данных
ВМ	Виртуальная машина
ЛК	Личный кабинет
ОС	Операционная система
ПО	Программное обеспечение
2FA	Two-factor authentication, двухфакторная аутентификация
API	Application Programming Interface, программный интерфейс приложения
AZ	Availability Zone, зона доступности
CPU	Central Processing Unit, центральный процессор
DNS	Domain Name System, система доменных имён
HCL	HashiCorp Configuration Language
HDD	Hard Disk Drive, жёсткий диск
IOPS	Input/output operations per second, количество операций ввода-вывода в секунду
IP	Internet Protocol, интернет-протокол
OSB	Open Service Broker
PID	Project ID, идентификатор проекта
RAM	Random Access Memory, оперативная память
S3	Simple Storage Service, сервис хранения объектов
SaaS	Software as a Service
SSD	Solid-State Drive, твердотельный накопитель
UUID	Universally unique identifier, универсальный уникальный идентификатор
XaaS	Everything as a Service